

BERTH AND QUAY CRANE SCHEDULING: PROBLEMS, MODELS AND SOLUTION METHODS

A Thesis
Presented to
The Academic Faculty

by

Aykagan Ak

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology
December 2008

BERTH AND QUAY CRANE SCHEDULING: PROBLEMS, MODELS AND SOLUTION METHODS

Approved by:

Professor Alan L. Erera, Advisor
School of Industrial and Systems
Engineering
Georgia Institute of Technology

Professor Ozlem Ergun
School of Industrial and Systems
Engineering
Georgia Institute of Technology

Professor Martin Savelsbergh
School of Industrial and Systems
Engineering
Georgia Institute of Technology

Professor Chelsea C. White III
School of Industrial and Systems
Engineering
Georgia Institute of Technology

Professor Prasad Tetali
School of Mathematics
Georgia Institute of Technology

Date Approved: 4 November 2008

To my lovely wife,

Ilgin,

who made this possible.

ACKNOWLEDGEMENTS

I would like to express my sincerest thanks to my advisor, Dr. Alan L. Erera. He has not only provided me his professional expertise, valuable guidance and the financial support I needed to complete my Ph.D. studies but also been a real friend and a true mentor. I feel extremely fortunate to work with him. He has always provided me the freedom to choose my research subjects and encouraged me to be involved in various types of research activities. During our academic discussions, I was very impressed by his enthusiasm and his ability in thinking extreme while keeping the real life applicability of the research high. I learned a lot from him, and I am very grateful to have him as my advisor.

I would like to thank Dr. Ozlem Ergun, Dr. Martin Savelsbergh, Dr. Chelsea C. White III, and Dr. Prasad Tetali for their time and effort in serving on my Ph.D. committee. I have had the chance to take wonderful classes from each one of them which helped me a lot in constructing the foundation of this dissertation.

I would also like to express my thanks to Dr. Glenn J. Rix, the director of the NEESR Grand Challenge project which initiated the research presented in this thesis.

I could not have achieved this without the continuous support and love of my family. Mom and dad, thank you for everything you did. Gokhan and Asli, you are the best brother and sister in the world.

Finally, and most importantly, my wife, Ilgin... There are no words that can help me to express my gratitude to you. You were always with me on rainy days and at sleepless nights. You smiled when I smiled, and you provided the shoulder when I cried. You kept me going, and you made this real. You are my pulse, my blood, my love...

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
LIST OF TABLES	ix
LIST OF FIGURES	xi
SUMMARY	xiii
I INTRODUCTION	1
1.1 Containerization and Trends in Maritime Logistics	1
1.2 An Overview of Container Terminal Operations	5
1.3 Motivation	8
1.4 Contributions of the Thesis	9
II THE BERTH ALLOCATION PROBLEM	15
2.1 Introduction	15
2.1.1 Models with Discrete Berths	16
2.1.2 Models with Continuous Berths	16
2.1.3 Comparison of Models	18
2.2 Problem Formulation	19
2.2.1 Position Assignment Formulation	22
2.2.2 Relative Position Formulation	23
2.2.3 Hardness of the Problem	26
2.3 Lower Bound Analysis	26
2.4 A Meta Heuristic: The Nested Tabu Search Algorithm	30
2.4.1 Encoding and Decoding a Solution	32
2.4.2 Initial Solution	33
2.4.3 Fundamentals of the Nested Search	34
2.4.4 Steps of the Nested Tabu Search Algorithm	39
2.4.5 Multi-start Version	41

2.5	Computational Experiments	41
III	THE MULTIPLE BERTH ALLOCATION PROBLEM	48
3.1	Introduction	48
3.2	Problem Formulation	49
3.2.1	Position Assignment Formulation	50
3.2.2	Relative Position Formulation	51
3.3	Lower Bound Analysis	52
3.4	Solution Method	56
3.4.1	Encoding and Decoding a Solution	56
3.4.2	Initial Solution	56
3.4.3	Fundamentals of the Nested Tabu Search	57
3.5	Computational Experiments	58
IV	THE QUAY CRANE SCHEDULING PROBLEM	64
4.1	Introduction	64
4.2	Problem Types and Models	70
4.2.1	Crane Blocking and Crane Shifting	70
4.2.2	Dedicated Quay Crane Scheduling	72
4.2.3	Roaming Quay Crane Scheduling	78
4.2.4	Analysis of Crane Scheduling Problems	81
4.3	A Tabu Search Algorithm	83
4.3.1	Initial Solution	83
4.3.2	Fundamentals of the Tabu Search	84
4.3.3	Steps of the Tabu Search Algorithm	87
4.4	The Crane Assignment Problem	88
4.5	Computational Experiments	92
V	THE SIMULTANEOUS BERTH AND QUAY CRANE SCHEDULING PROBLEM	96
5.1	Introduction	96

5.2	Problem Formulation	99
5.3	Lower Bound Analysis	104
5.3.1	Lower Bound 1	104
5.3.2	Lower Bound 2	104
5.4	Solution Method: A Two Phase Tabu Search Heuristic	107
5.4.1	Encoding and Decoding a Solution	108
5.4.2	Initial Solution	110
5.4.3	Single Phase Search	110
5.4.4	Two Phase Search	114
5.4.5	Additional Remarks	116
5.5	Computational Experiments	117
VI	THE VOYAGE AND BERTH SCHEDULING PROBLEM	122
6.1	Introduction	122
6.2	Problem Formulation	125
6.2.1	Cost Structure	126
6.2.2	Modeling with Network Flow	130
6.2.3	Analysis of the Network Flow Formulation	135
6.3	Hardness of the Problem	139
6.4	Further Discussion	143
6.4.1	Terminal Time Windows	144
6.4.2	Service Line Durations	144
6.4.3	Transshipments	145
6.4.4	Schedule Reliability	146
6.4.5	Fleet Restrictions	149
6.5	Computational Experiments	149
6.5.1	Instance Generation	150
6.5.2	Results	151
VII	CONCLUSIONS AND FUTURE WORK	161

REFERENCES	164
VITA	170

LIST OF TABLES

1	Statistics for test BAP instances used	42
2	Tabu search parameters used for BAP computational experiments . .	43
3	Test results for small BAP instances	44
4	Test results for large BAP instances	46
5	Average computational times (in sec.) for test BAP instances	47
6	Statistics for the small MBAP instances	59
7	Statistics for the large MBAP instances	60
8	The MBAP tabu search parameters used in the computational exper- iments	60
9	Test results for small MBAP instances	61
10	Test results for large MBAP instances	62
11	Average computational times for test MBAP instances	63
12	Computational analysis of crane scheduling strategies	82
13	The tabu search parameters used in the computational experiments .	92
14	Summary of results for small QCSP instances	93
15	Summary of results for large QCSP instances	95
16	Summary of computation times observed in seconds	95
17	A sample BQCSP instance	101
18	A feasible solution to the sample BQCSP instance	101
19	Single Phase Search and Two Phase Search parameters used in the computational experiments	117
20	Test results for small BQCSP instances	118
21	Test results for large BQCSP instances	119
22	Average computation times observed (in sec.) for test BQCSP instances	120
23	Comparison of sequential and simultaneous berth and quay crane plan- ning	121
24	Summary of results for test VBSPP instances	153
25	Effect of time windows on the optimal solution	155

26	Effect of service line duration limits on the optimal solution	156
27	Effect of transshipments on the optimal solution	157
28	Effect of voyage speed buffers on the optimal solution	158
29	Effect of berth buffers with $\varepsilon = 1$ on the optimal solution	159
30	Effect of berth buffers with $\varepsilon = 2$ on the optimal solution	159

LIST OF FIGURES

1	Six generations of containerships.	2
2	Emma Maersk. Source: http://img.dailymail.co.uk/	3
3	Container traffic at major US ports. Source: American Association of Port Authorities (AAPA).	4
4	Schematic representation of a container terminal system. Source: Steenken et al. 2004.	5
5	A container terminal at the Port of Antwerp. Photograph by Thomas Vanhaute.	7
6	Discrete berth models can be employed for some continuous berths. .	19
7	Representation of a berth schedule on a time-space diagram	23
8	Construction of lower bounding scheduling problem P for a given dynamic BAP instance	28
9	Multiple priority lists may correspond to a single berth schedule . . .	32
10	Encoding and decoding a berth schedule	33
11	Illustration of moving from one solution to another by modifying \mathcal{L} and \mathcal{B}	35
12	The importance of using the primal list in the inner search, TS_2 . . .	37
13	Schematic Representation of the Nested Tabu Search Approach . . .	39
14	Illustration of the graph constructed to solve the CAP	89
15	Representation of a BQCSP solution on the time-space diagram . . .	100
16	Construction of lower bounding scheduling problem P_2 for a given dynamic BQCSP instance	105
17	An illustration of improving a given solution by changing berth schedule and quay crane schedule	108
18	Schematic representation of the single phase search approach for the BQCSP	111
19	Schematic representation of the two phase search approach for the BQCSP	115
20	Illustration of a typical vessel crew cost function used in VBSP	126
21	Illustration of a typical fuel consumption function used in the VBSP .	128

22	Illustration of typical terminal cost functions used in the VBSP . . .	129
23	Representation of the graph used to model the VBSP	131
24	Illustration of the voyage cost function used in the model constructed for the VBSP	132
25	An example of a possible fractional optimal solution for the model . .	137
26	Existence of an optimal valid route flow on the graph	138
27	Change in voyage cost function with berth schedule buffer	148

SUMMARY

In this thesis, we focus on planning problems related to berth and quay cranes which are the most important resources in container terminals at seaports. Many researchers have concentrated on berth and quay crane planning; however, recent trends and changes in maritime logistics, like the introduction of mega-ships and increasing popularity of flexible continuous berth structures, have created gaps in the current literature. In this thesis, we contribute by filling many of these gaps and provide a comprehensive analysis of berth and quay crane planning problems.

The thesis can be divided into two parts. The first part consists of four chapters in which problems most applicable to multi-user terminals are studied. In Chapter 2, we focus on the berth allocation problem (BAP) which is the problem of assigning ships to berthing positions within a seaport to optimize some performance metric. The variant we study considers dynamic arrival of vessels and a single berth having a long continuous structure that can serve multiple vessels simultaneously. We provide a polynomially computable lower bound and develop a meta-heuristic algorithm based on tabu search which uses a novel nested neighborhood structure to solve large problem instances. In the next chapter, we introduce the multiple berth allocation problem (MBAP) which is defined for a container terminal with multiple disjoint berths that can each handle multiple vessels simultaneously. Arriving vessels can be assigned to any position at any berth, but the processing time required by vessels can change depending on the berth assignment due to the distance from yard storage area and equipment used at the berth. We modify the solution procedure introduced for the BAP in order to handle the MBAP. Chapter 4 is dedicated to the quay crane

scheduling problem (QCSP) which is the problem of assigning quay cranes to vessels moored at the berth in order to optimize container unloading and loading. The QCSP variant we introduce uses a berth schedule, which defines positions of vessels at the berth and berth usage priority, as input. Therefore, unlike most earlier studies, we assume a longer planning horizon and consider multiple vessels including those not present at the port at the time of decision making. We analyze several crane split strategies and provide a high speed tabu search algorithm. We conclude the first part of the thesis by introducing the simultaneous berth and quay crane scheduling problem (BQCSP) in Chapter 5. In practice, berth scheduling and quay crane scheduling problems are approached sequentially by terminal operators. They first determine a berth schedule using estimates of total berth time for each vessel, and then try to split cranes among the vessels planned to dock simultaneously. Terminal operators can develop a better operational plan if actual crane requirements are considered while determining berth schedules. We propose a meta-heuristic algorithm which combines the features of the algorithms designed to solve the BAP and the QCSP. Two polynomially computable lower bounds are also introduced and used in the evaluation of the proposed algorithm.

The second part of the thesis, Chapter 6, focuses on berth scheduling at dedicated terminals and its impact on vessel voyage planning. At a dedicated terminal, vessels of only one carrier are serviced, and the carrier has direct control of the schedules of all vessels visiting. For such terminals, scheduling one vessel earlier or later impacts the times that other vessels may be scheduled, and thus there are interactions between the schedules of different routes that visit common dedicated terminals. We introduce therefore the voyage and berth scheduling problem (VBSP) which is the problem of scheduling vessel voyages and determining optimal voyage speeds between port calls for multiple intersecting voyages operated by a single ocean carrier or alliance. The goal is to develop feasible visit schedules for all voyage routes at all visited ports to

minimize total operating cost, which is a function of sailing speeds and delays experienced between port calls. A mathematical model based on multi-commodity network flow is developed and solved on a series of realistic test problems. We also show how transshipments, terminal time windows, and service levels can be incorporated in the model as well as how improvements to schedule reliability can be achieved by properly modifying the instance data.

CHAPTER I

INTRODUCTION

1.1 Containerization and Trends in Maritime Logistics

In the late 1950s, the first ocean container ships were introduced, and containerized transportation changed the way of transporting general cargo drastically. Before containerization, most general cargo was handled by building pallets and loading them into the holds of vessels using cranes on the ship and on the wharf. This labor-intensive process was very slow, and goods transported were vulnerable to damage. With containers, easy and fast handling of freight became possible. Containers enabled international door-to-door transportation, with the goods packed at the shipper and not unloaded until arrival at the consignee.

Containers are metal boxes with standardized dimensions. There are five common standard lengths, 20-ft, 40-ft, 45-ft, 48-ft, and 53-ft. Containers of lengths 48-ft and 53-ft are not used for international ocean transportation, and instead are mainly used in domestic rail-truck intermodal services. Container capacity and throughput measurements are often expressed in twenty-foot equivalent units (TEU). A TEU is a measure of containerized cargo capacity equal to one standard 20-ft (length) \times 8-ft (width) \times 8-ft 6-in (height) container. Note that the TEU is an approximate measure. For instance, the 9-ft 6-in high cube and the 4-ft 3-in half height 20-ft containers are also denoted one TEU. Similarly, one 45-ft container is also commonly designated as two TEU. The most common container type in international ocean carriage is the general purpose dry 40-ft (2 TEU) container. Specialized containers are also used for various types of commodities. In addition to general purpose dry containers, there are also temperature controlled (reefer) containers for frozen items, open top bulkcontainers

for bulk minerals, ventilated containers for organic products requiring ventilation, tank containers for bulk liquids, gas containers, and many others.









		Length	Draft	TEU
First (1956-1970)	 Converted Cargo Vessel	135 m	< 9 m	500
	 Converted Tanker	200 m	< 30 ft	800
Second (1970-1980)	 Cellular Containership	215 m	10 m 33 ft	1,000 – 2,500
Third (1980-1988)	 Panamax Class	250 m	11-12 m 36-40 ft	3,000
		290 m		4,000
Fourth (1988-2000)	 Post Panamax	275 – 305 m	11-13 m 36-43 ft	4,000 – 5,000
Fifth (2000-2005)	 Post Panamax Plus	335 m	13-14 m 43-46 ft	5,000 – 8,000
Sixth (2006-)	 New Panamax	397 m	15.5 m 50 ft	11,000 – 14,500

Figure 1: Six generations of containerships.

The last decade of the 20th century witnessed significant growth in worldwide container transportation. This steady increase in international intermodal container traffic has led to changes in the business practices of ocean carriers and container terminal operators. Carriers are faced with higher and higher shipping demands, and have responded with the introduction of new routes and more frequent service on existing ones. Because of that, they invest heavily on sophisticated decision support systems and new equipment. In order to satisfy high shipping demands, larger container vessels have been built; today, construction of vessels with capacities as large as 15,000 TEUs, called mega-ships, is planned. An overview including statistics detailing increasing container vessel sizes and traffic growth is provided in [8]. In [7] and [73], the impact of larger vessels on ports is discussed. Some authors caution that vessel

sizes may soon begin to plateau; [47] and [69] provide cautious arguments supporting this notion. The former suggests that the benefits of larger vessels will likely peak due to physical restrictions in ports, while the latter argues that economies of scale likely diminish beyond capacities of 3000 TEUs and disappear over 8000 TEUs. Figure 1 adopted from [62] illustrates the evolution of container vessels, and provides specifics for each vessel generation. The largest mega-ships currently in use, Emma Maersk and her three sister ships, each have a length of 397m and a capacity of 11,000 TEUs [46]; see Figure 2 for an illustration.



Figure 2: Emma Maersk. Source: <http://img.dailymail.co.uk/>

The introduction of larger vessels into fleets has certainly reduced per container transportation costs for both carriers and shippers. However, such vessels also increase the stress on container terminals since they both require faster handling to achieve vessel dwell times similar to those of smaller vessels, and also new equipment like larger quay cranes and deeper and longer berths. Consequently, terminals are faced

with more and more containers to be handled in shorter times. Figure 3 depicts the increase in the number of TEUs handled each year at the top 5 US ports since 1996. To accommodate these increases in container traffic, container terminal operators and port authorities must both invest in new infrastructure and equipment, and also find ways to use existing resources more efficiently. In addition, due to the competition between container terminals, especially between those in close geographic proximity, terminal operators also need to reduce costs and increase reliability to provide effective service to ocean carriers. Terminal operators strive to achieve rapid container vessel unloading and loading, which corresponds to a reduction of the time in port for the vessels. A comprehensive discussion on the relationship between container terminal operators and ocean carriers is provided in [66].

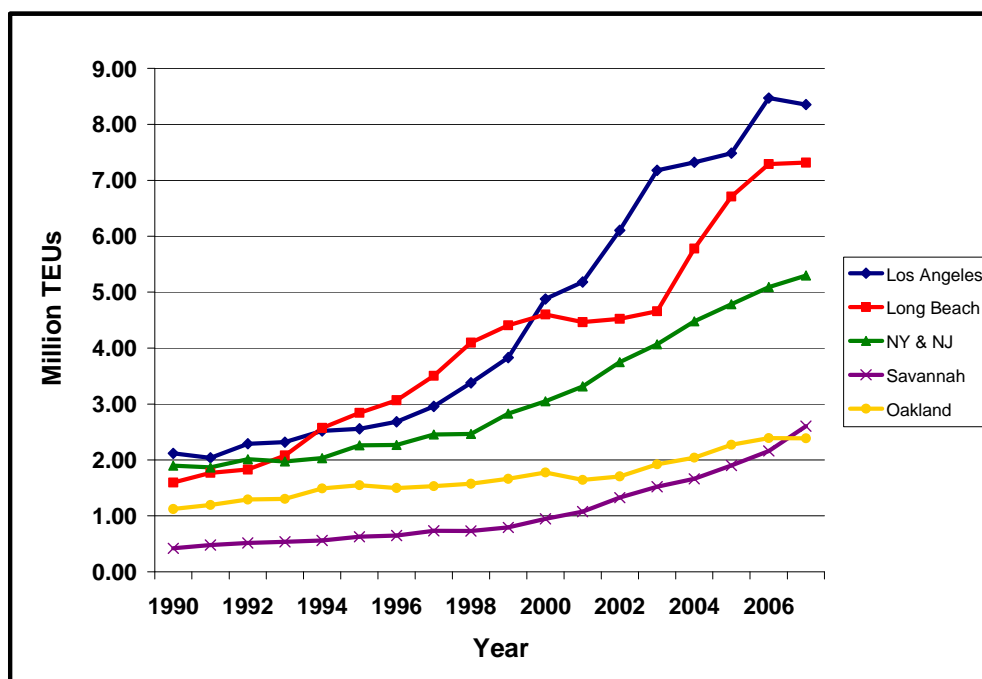


Figure 3: Container traffic at major US ports. Source: American Association of Port Authorities (AAPA).

1.2 *An Overview of Container Terminal Operations*

In general terms, container terminals can be viewed as open systems of material flow with two external interfaces. These interfaces are the quayside with loading and unloading of vessels, and the landside where containers are loaded and unloaded on and off trucks and trains. A container yard connects the quayside and landside, and provides space for container storage. Containers are stored either in stacks on the yard deck, or on truck chassis. Under a chassis storage system, each container is individually accessible providing fast transfer to landside movements. Alternatively, under a stacking system, containers are often stored in stacks several containers high, which means that not every container is directly accessible when needed. Yard cranes are utilized to access containers and reposition them within the stack. Because of increased demand and limited storage space in most modern seaports, nowadays stacking on the ground is the most commonly used storage approach.

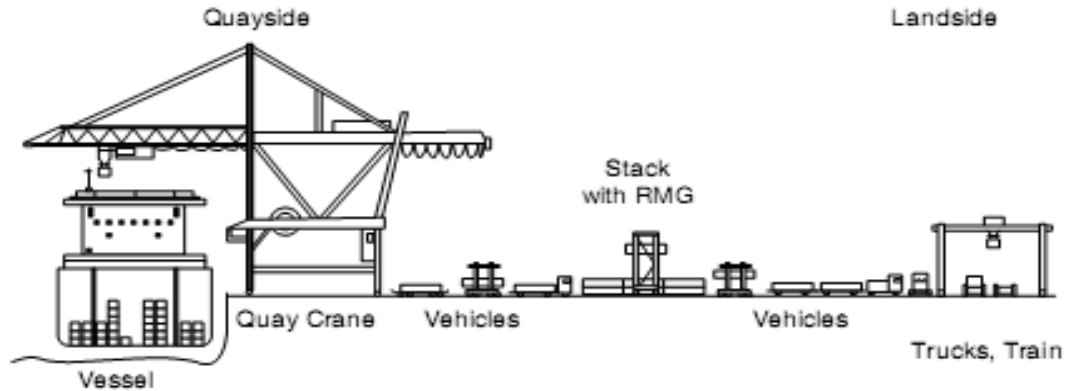


Figure 4: Schematic representation of a container terminal system. Source: Steenken et al. 2004.

When a vessel arrives in a seaport, it first has to moor for container loading and unloading. For this purpose, a number of berths are available at container terminals. Berths have very large construction costs, and therefore the number and length of berths at a container terminal is one of the most important strategic decisions that

must be made at the strategic level. Berthing decisions initiate the work within a terminal by pushing and pulling containers into and from the yard storage areas. Clearly, the utilization of berths directly affects the overall utilization of the terminal, and therefore the operational level decision of allocating berth space to vessels is crucial. Most container berths in the large ports of the United States and Japan are leased by ship operators. Under such arrangements, ocean carriers are directly responsible for the containers. Such berthing systems are called dedicated berth systems, and terminals operating with dedicated berths are called dedicated terminals. An alternative system, known as public berths, is used by many major hub ports like Hong Kong, Singapore, Rotterdam, and Hamburg. Public berth systems are used in multi-user terminals that process the vessels of different carriers, and generally have longer berths and higher berth utilization rates than dedicated terminals.

When a vessel is moored at a berth, the unloading and loading of containers begins. Quay cranes are the standard equipment designed for this task. A quay crane is a special type of gantry crane having a large steel framework, which is positioned along the wharf (or quay) alongside a berthed vessel. Quay cranes are generally classified by their lifting capacity, and the size of the container ships they can load and unload. A Panamax crane can fully load and unload containers from a container vessel capable of passing through the Panama Canal (vessels 12-13 container rows wide). A Post-Panamax crane can fully load and unload containers from larger container vessels up to about 18 container rows wide. The largest modern container cranes are classified as Super-Post Panamax, and are used for vessels up to 22 container rows wide). A modern container crane capable of lifting two 20-ft containers at one time generally has a lifting capacity of at least 40 tonnes. Some new cranes have now been built with 120 tonne load capacity enabling them to lift up to four twenty foot or two forty foot long containers. The speed of quay cranes during unloading and loading movements is also important. Modern quay cranes have hoisting speeds of 60-80 m/min when

carrying a load. Trolley speeds can exceed 140 m/min. Given these parameters, it takes about 90 seconds to load or unload a single 40-ft container with an experienced crane operator. Post-Panamax cranes weigh approximately 800-900 tonnes while the newer generation Super-PostPanamax cranes can weigh 1600-2000 tonnes. To enable large, heavy quay cranes to move along the wharf, they are mounted on rails parallel to the water. Quay cranes are the second most expensive infrastructure investment at a container terminal after berths. Since most berths have a limited set of such expensive cranes, one of the key operational bottlenecks at container terminals is the quay crane availability. By improving quay crane efficiency, ports can reduce vessel turnaround time, and improve overall terminal productivity and throughput.



Figure 5: A container terminal at the Port of Antwerp. Photograph by Thomas Vanhaute.

The number of optimization problems defined for container terminal operations is enormous as well as the amount of research conducted. A comprehensive literature review is provided by [71], [67] and [66] on problems observed at container terminals.

1.3 Motivation

The research presented in this dissertation was motivated by the NEESR Grand Challenge project conducted on the seismic risk management for port systems which is sponsored by National Science Foundation, Network for Earthquake Engineering Simulation, and National Earthquake Hazards Reduction Program [1]. Since many U.S. seaports are located in areas with significant seismic hazard, and since these seaports are critical gateways for national and international trade, the damage caused by earthquakes and associated business interruption losses can have devastating consequences for the port and broader, adverse effects on local, regional, national, and international stakeholders. The NEESR Grand Challenge project integrates geotechnical and structural earthquake engineering research with expertise in port system operations and risk and decision analysis of containerized port systems. Individual tasks in the project include:

- predicting the seismic response and resulting damage states of key port components such as container berths and cranes via large-scale experimentation and numerical simulation,
- estimating the effects of damage to these components on cargo-handling capacity and the resulting impact on port revenues, and
- mitigating possible losses via both geotechnical and structural engineering design and retrofit options.

The algorithms and solution methods provided for port planning problems in this thesis are used in construction of the high speed optimization engines which are utilized by the simulation tool designed for the project.

1.4 Contributions of the Thesis

As mentioned before, berths and quay cranes are the most crucial resources at a container terminal. They are not only the most expensive equipment, but they also initiate container flow within the terminal. Efficient and effective utilization of berths and quay cranes is essential for overall port throughput. Many researchers have focused on planning problems related to berths and quay cranes. However, recent trends and changes in maritime logistics, have created gaps in the current literature. In this thesis, we contribute by filling many of these gaps. This study provides a comprehensive analysis of berth and quay crane planning problems. Some of these problems are newly defined in the research literature, even though they exist in practice. The multiple berth allocation problem, the simultaneous berth and quay crane scheduling problem, and the voyage and berth scheduling problem presented in the later chapters of the thesis are in this group. The well known berth allocation problem and the quay crane scheduling problem are also studied. A new polynomially computable lower bound and an effective meta-heuristic solution method for the continuous berth version is provided for the former. An analysis of crane splitting methods is performed, and a very fast solution method is described for the latter. Furthermore, the significant benefit that can be obtained by simultaneously scheduling berth and quay cranes is demonstrated by this thesis.

We first focus on an *NP*-hard berth allocation problem (BAP) which is the problem of assigning ships to berthing positions within a seaport to optimize some performance metric. We consider the dynamic variant of the BAP, where we model the dynamic arrival of vessels during the planning horizon. We use a model that treats the berth as a continuous facility where a vessel can moor anywhere along the berth simultaneously with other vessels. We provide mixed integer programming (MIP) formulations, and then develop a meta-heuristic algorithm based on tabu search which employs a novel nested neighborhood structure to solve large problem instances. We

also introduce a lower bound which can be computed quickly in polynomial time, and is provably tighter than the bound generated by solving the LP relaxation of the associated MIP. Computational experiments show that the algorithm is able to provide high quality solutions in relatively short computation times. Instances with 10 to 14 vessels were solved optimally by the nested tabu search algorithm under 4 seconds, whereas, for some of these instances, it took days to solve with the MIP. More realistic instances with 20 to 30 vessels were also solved within 100 seconds. For those instances, we were able to reduce the optimality gap of a reasonable initial solution by 70% on average.

In chapter 3, we introduce an extension of the BAP, where multiple physically-disjoint berths may be used to moor arriving vessels. We call this problem the multiple berth allocation problem (MBAP). In this problem, we consider a container terminal with multiple berths, where each berth is a continuous facility that can handle multiple vessels of different lengths simultaneously. Arriving vessels can be assigned to any berth within the terminal, but the processing time required for vessels can change depending on berth assignments. Hence, the MBAP is defined as the problem of assigning vessels to berths, and determining the berthing position and time within the berth that vessels are assigned, to optimize a performance metric. We show how our BAP models can be generalized to the MBAP case. We also show that the nested tabu search algorithm designed for the BAP can be an effective tool to solve the MBAP, given appropriate modifications. A lower bound analysis is again conducted, and the results are again used in the development of polynomially-computable bounds provably stronger than that of the LP relaxation. Computational experiments show that the modified nested tabu search algorithm can solve small instances with 10-14 vessels optimally under 10 seconds. Larger instances with up to 50 vessels, for which MIP cannot even find an integer solution, are solved within 20 minutes. For those instances, we are able to reduce the optimality gap of a reasonable initial solution by

68% on average.

Chapter 4 focuses on the quay crane scheduling problem (QCSP). Quay cranes are standard equipment used to load containers onto or discharge containers from vessels at the quayside of terminals. Most earlier studies concerning quay crane scheduling focus on detailed models applicable to either one vessel or a set of vessels docked at the time of decision making. In this chapter, we consider the QCSP for a given berth schedule. A berth schedule defines where and when each vessel is planned to dock at the berth over some planning horizon. Hence, in the problem variant we study, not only are the vessels that are present at the berth at the time of decision making considered but also vessels planned to arrive later. We first define the concepts of crane blocking and crane shifting. Because quay cranes at a berth operate on a single railway, they cannot cross over each other. This restricts the range that a crane can operate on the berth, which is clearly limited by the locations of its neighbor cranes. Crane blocking occurs when the only available work for a crane is located outside of this crane's operating range for a given time period; hence no work can be assigned to that crane at this time period. Sometimes, crane blocking can be solved by shifting multiple cranes along the quay, but shifting cranes efficiently is often difficult in practice. We also introduce two crane scheduling methods; dedicated crane scheduling and roaming crane scheduling. In dedicated crane scheduling, a set of cranes is dedicated to each vessel, and those cranes cannot be used to process other vessels until the vessel that they are assigned departs. On the other hand, in roaming crane scheduling, a crane can be reallocated to other vessels if it is no longer needed by the vessel that it is initially assigned or if a better plan can be obtained by doing so. Hence, cranes can roam on the berth between vessels during operation. We develop exact optimization models for each crane scheduling method under both blocking and shifting assumptions, and analyze the methods computationally using a set of small instances. We show that roaming crane scheduling with crane shifting can provide

significantly better operational plans by increasing crane utilization, and design a tabu search algorithm to solve realistic instances under this scenario. We also introduce the crane assignment problem (CAP), which is the problem of determining individual assignment of cranes to vessel positions in order to minimize the distance traveled by cranes on the berth. We show that the CAP can be solved in polynomial time since it can be modeled as a minimum cost network flow problem. Computational experiments indicate that the tabu search algorithm designed is able to find optimal solutions almost instantaneously for small problems with 10 to 14 vessels and 6 cranes. For larger problem instances with 20 to 30 vessels and 10 cranes, the optimality gap was improved by approximately 68% within 5 to 15 seconds respectively.

In chapter 5, we focus on the simultaneous scheduling of berth and quay cranes. In practice, berth scheduling and crane scheduling problems are generally considered sequentially by port operators. They first determine a berth schedule using estimates of total berth time for each vessel, which is based on the vessel size and workload. Then, they allocate quay cranes to the vessels that are planned to dock simultaneously at the berth. In this chapter, we consider the problem of scheduling these two resources simultaneously, and denote the problem as the simultaneous berth and quay crane scheduling problem (BQCSP). We first propose a mixed integer program for the BQCSP. As expected, this MIP can only solve small instances optimally. Therefore, we propose a meta-heuristic approach, which combines the features of the algorithms described in Chapter 2 and Chapter 4. Two polynomially-computable lower bounds are introduced. The first one is generated by relaxing a constraint on berth positioning, while the second one is based on relaxing a constraint on the number of available cranes. These bounds are then used in the evaluation of the proposed algorithm. Computational experiments are designed and performed to evaluate the hybrid meta-heuristic designed for the BQCSP. The MIP can only be used to solve very small instances with 5 or 6 vessels. We generated 100 of such instances, and

found optimal solutions in an average run time of 15 minutes. Our algorithm found and optimal solution for each of these instances within 2 seconds. The algorithm was then tested on moderate size instances with 10 to 14 vessels and 6 cranes. We were able to reduce the optimality gap of a reasonable initial solution by 66% within 1 minute. The algorithm provided even better improvement for larger instances with 20 to 30 vessels and 10 cranes. We were able to obtain an average of 77% reduction in the optimality gap of the initial solution in 3 to 30 minutes. We conclude the chapter by comparing sequential berth and crane allocation with the simultaneous approach. Computational study shows that simultaneous berth and quay crane scheduling may yield savings as large as 35%.

Finally, Chapter 6 of the thesis considers resource allocation at container terminals from a different point of view. In the first four chapters, we concentrate on berth and quay crane scheduling problems in which the decision maker is the terminal operator. These models are most appropriate for multi-user terminals where the terminal operator and carriers are different parties, and parameters like vessel arrival time, target departure time, and tardiness penalties are given. For multi-user terminals, carriers generally have very limited influence on berthing positions and crane allocations, and have no influence on resource planning for vessels that are operated by other carriers. In dedicated terminals operated by a carrier, however, the carrier alternatively has direct control of the schedules of all vessels visiting the same terminal, since they all belong to the same party. In this chapter we introduce the problem of scheduling vessel voyages considering berth resource limitations. The decision maker is a liner carrier having a set of vessels operating on a predetermined set of routes. This problem is defined for large liner carriers, or alliances, which have many vessels operating on different routes that visit some busy hub ports operating with multi-user terminals, but intersect often at dedicated terminals controlled by the carrier. Such carriers want to operate their vessels on predetermined routes while

respecting capacity limitations at dedicated terminals and avoiding congested time periods at multi-user terminals. Their objective is to determine vessel arrival times at each port visited within each vessel route, which is equivalent to setting voyage speeds, in order to minimize fuel cost and the cost of delays. We name the resulting problem the voyage and berth scheduling problem (VBSP). A model based on the multi-commodity network flow is presented. We prove that when only the dedicated terminal capacities are relaxed, the model can be decomposed for each route and an optimal integer flow can be found by an improved version of the LP relaxation with the help of additional valid inequalities. We show how this result can be used to reduce the problem size, and present a compact valid and efficient formulation using this idea. We conclude this chapter by showing how transshipments, which are very important for ocean carrier networks, can be formulated as well as terminal time windows and limits on service line durations. We also discuss how reliability of schedules can be improved by properly modifying the model and the data. Computational experiments are performed and results are presented.

CHAPTER II

THE BERTH ALLOCATION PROBLEM

2.1 Introduction

The berth allocation problem (BAP) is the problem of assigning arriving vessels to berthing positions within a terminal to optimize some performance metric. Managing berth allocation to arriving vessels is especially critical in multi-user terminals, since poor choices can cause unnecessary delays in ship processing and resultant carrier dissatisfaction. Furthermore, note that the ship berthing process triggers the work of other port resources, such as the large, expensive quay cranes used to unload and load containers from and onto vessels, as well as yard cranes and jockey truck crews. Poor berth utilization may lead to under-utilization of these resources, as well, and to an overall reduction in port throughput.

Static and dynamic variants of the BAP have been studied. The static variant considers vessels available for berthing at the port at the decision time. On the other hand, the dynamic variant allows arrival of vessels at different times during the planning horizon. Both static and dynamic BAPs have been addressed using two main classes of models. Models in the first class partition the berthing area into discrete berths, where each berth is occupied by at most one vessel at a time. Alternatively, models in the second class treat the berth as a linear facility where multiple vessels can moor simultaneously. Often, a linear berth is divided into small non-overlapping sections of constant length, and a number of adjacent sections are simultaneously assigned to each berthed vessel. Compared to models with discrete berths, this modeling perspective typically provides more freedom for port operators in the decision of assigning berth positions.

2.1.1 Models with Discrete Berths

Among the first studies developed in the discrete berth case are [34] and [9]. The former compares three heuristic berth allocation rules using simulation, and proposes a first-come-first-served (FCFS) heuristic, whereas the latter formulates an integer programming model for assigning vessels to berth positions. Researchers in [29] also develop a model for a discrete berth structure, and propose a heuristic solution procedure based on a Lagrangian relaxation of the original model. They extend this work in [28] by incorporating service priorities for vessels. A genetic algorithm is proposed in [54] for a nonlinear formulation of the discrete BAP. The approach allows simultaneous service for up to two vessels assigned to the same berth area if total vessel size does not exceed the size of the berth area.

2.1.2 Models with Continuous Berths

The berthing area was first modeled as a single linear facility rather than a collection of discrete areas in [40]. The paper proposes the so-called berth planning problem (BPP) of determining a feasible berthing plan for a berth of finite length and a set of arriving vessels with known lengths, predetermined duration of stays, and fixed berthing times. A two-dimensional packing problem formulation is proposed, and a heuristic solution method is presented that yields good results on the instances considered. Among other studies using linear berthing area models, [39] considers the static problem variant with a minimum makespan objective, and provides worst-case performance analysis for a heuristic based on the first fit decreasing algorithm for the bin packing problem. The static BAP with the objective of minimizing total weighted completion time is considered in [25], and a solution method based on a multiprocessor task scheduling model is proposed. The authors assume that vessel lengths and processing times are positively correlated. They present a lower bound, and perform worst-case analysis on a heuristic solution approach. This work is extended to handle dynamic

variant in [24], two mathematical formulations along with a tree search procedure are presented, and a heuristic for the case with batch arrivals is introduced. Results indicate that instances with up to 10 vessels can be solved optimally within reasonable computation time limits.

The study in [57] also concentrates on the dynamic BAP with a different objective that penalizes both deviations from best berthing locations and also delayed vessel completion times. The paper presents a mixed integer programming formulation, which is then discretized in time and space. Computational experiments indicate that a solution method based on Lagrangian relaxation works well for instances with fewer than 7 vessels, and provides near optimal solutions for most instances with fewer than 20 vessels. In [32], a simulated annealing algorithm is proposed for the same setting and objective function. Computational results demonstrate that the method solves most instances with fewer than 7 vessels optimally; larger instances are solved, but their solution quality is not assessed. In [30], the authors extend their previous work on the discrete berth variant of the dynamic BAP to the linear berthing area case. In the proposed model, vessel processing time depends on berthing position. The paper details a heuristic adapted from their discrete case heuristic, and compares the performance of the two approaches. No optimality gaps are reported for the instances solved. Researchers in [14] propose a new model to minimize total weighted service time for a dynamic BAP with a berthing area partitioned into discrete berths. The model is based on the multiple depot vehicle routing problem with time windows, and the paper develops a tabu search heuristic for its solution. The paper additionally proposes modifications that enable application of the heuristic to the dynamic BAP with a linear berthing area. Computational results indicate that the algorithm improves FCFS rule solutions with reasonable computation times.

The papers [18] and [72] both focus on the dynamic BAP with the objective of minimizing total weighted vessel delay and penalties for deviating from best berthing

locations. Researchers in [18] formulate the problem as a rectangle packing problem and present a simulated annealing algorithm. The neighborhood structure used is defined by augmenting the sequence pair approach proposed in [27] using similar ideas to those in [50] which represents a solution by a pair of permutations of rectangles. Optimality gaps are provided for congested scenarios. For lightly loaded scenarios, where optimal solutions are known, it is shown that the algorithm generates high-quality solutions. The dynamic berth allocation problem is treated as a multi-stage decision making procedure in [72]. A stochastic beam search solution algorithm is developed. The algorithm is shown to outperform the simulated annealing algorithm proposed in [18] on test instances, but the paper does not provide measures of berth congestion for the instances considered.

2.1.3 Comparison of Models

As mentioned before, models with discrete berths have been employed widely by viewing the berthing area as a finite set of berths where each berth can accommodate only one vessel at a time. With such models, if the spatial dimension is ignored, berths can be viewed as points. As a result, these models treat the BAP as a parallel machine scheduling problem, where a vessel is treated as a job and a berth as a machine. The vessel's arrival time is the job release time. Models with continuous berthing areas, on the other hand, allow that vessels can berth anywhere along the berth. Compared to models with discrete berths, this modeling perspective typically provides more freedom for port operators in the decision of assigning berthing positions. However, solving continuous berthing area models is typically much harder.

Models that use discrete berthing areas can be used effectively by some terminals even though such terminals have single continuous berthing areas. If vessels calling are approximately the same length, the terminal operator can simply divide the berthing area into discrete segments of common vessel size. Similarly, if the berth is longer

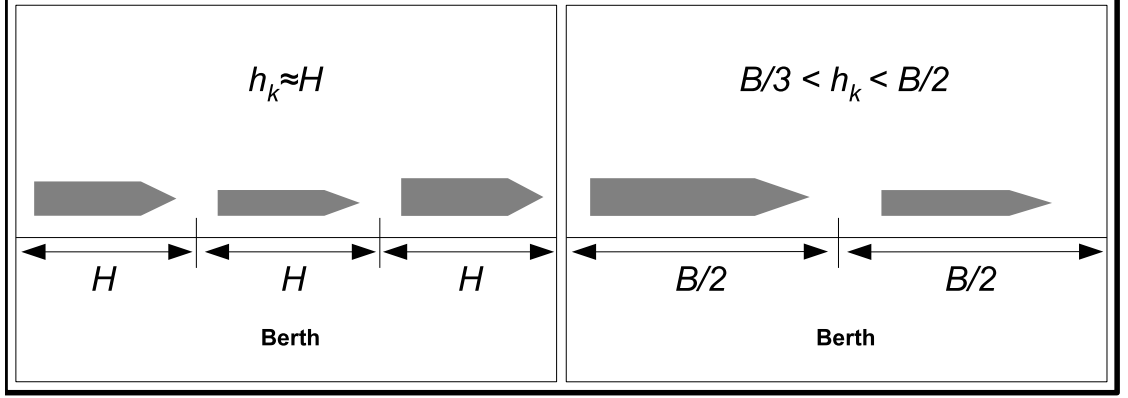


Figure 6: Discrete berth models can be employed for some continuous berths.

than twice but shorter than three times the length of any vessel calling, defining two discrete berths is enough to model all possible simultaneously berthing possibilities. Figure 6 illustrates the idea.

The growth trend in container vessel sizes has resulted in a parallel increase in the diversity of vessel sizes, since older vessels are generally not retired but instead placed into service on different routes. For terminals that serve a mix of container vessels of different lengths, dividing the berthing area into discrete segments is not easy. Using long segments will result in poor berth utilization, while using short segments may result in infeasible models for otherwise feasible problems. Because of this, flexible berth allocation planning is gaining more importance, especially in busy hub ports where vessels of various sizes are calling and long continuous berths are employed.

2.2 Problem Formulation

In this chapter, we consider a dynamic variant of the BAP where the berthing area is modeled as a linear facility that can berth multiple vessels simultaneously. We propose an objective function that balances the goals of ocean carriers (who desire minimal processing delay for arriving vessels) and the terminal operator (who desires maximum vessel and container throughput). The primary contributions of this study include:

- The introduction of a lower bound that can be found in polynomial time for any instance of the dynamic variant of BAP, where the bound is provably tighter than the linear programming relaxation bound for a standard integer programming model for the problem; and
- The development of an effective tabu search algorithm specifically designed for the continuous berth variant of the BAP algorithm that utilizes a novel solution encoding and a nested neighborhood search procedure.

We consider a problem where the berth is modeled as a single continuous linear structure having B equal size sections. We discretize time and assume an infinite planning horizon. When berthed, a vessel covers a number of adjacent sections depending on its size, and multiple vessels can moor at the berth and receive service simultaneously. We concentrate on the dynamic problem where we have a set \mathcal{V} of vessels with known arrival times, where $n = |\mathcal{V}|$. For each vessel $k \in \mathcal{V}$, we define:

- h_k : length of vessel k measured in number of required berth sections,
- p_k : processing time of vessel k ,
- a_k : arrival time of vessel k ,
- d_k : due time of vessel k (where $d_k \geq a_k + p_k$),
- f_k : lateness penalty of vessel k ,
- b_k : planned berthing position of vessel k ,
- t_k : berthing time of vessel k ,
- c_k : the earliest time that vessel k can depart (its completion time $t_k + p_k$).

A feasible solution of the BAP is called a berth schedule x . Any such x can be depicted on a time-space diagram where the horizontal axis measures time and the vertical axis represents berth sections; see Figure 7. In such a representation, a vessel

can be represented by a rectangle whose length is its processing time and height is its length. Given a known vector of arriving vessel information $\{h_k, p_k, a_k, d_k, f_k\}$, the optimization problem is then to determine berthing section b_k and berthing time period t_k for each vessel k . If we say vessel k is berthed at position b_k at time t_k , we mean berth sections $[b_k, b_k + h_k - 1]$ are occupied by vessel k for time periods $[t_k, t_k + p_k - 1]$. This also means that once a vessel is berthed, its location cannot be changed during service, and no preemption is allowed.

Our objective is to minimize $\sum_{k \in \mathcal{V}} (c_k - a_k) + \sum_{k \in \mathcal{V}} f_k (c_k - d_k)^+$. The first term is the sum of the dwell times (often called flow times in the scheduling literature), where a vessel's dwell time is measured between arrival and departure including both time waiting to be berthed and servicing time while berthed. The second term is a penalty accrued by tardy vessels, where $(c_k - d_k)^+$ measures the tardiness of vessel k using the standard notation $z^+ = \max(z, 0)$. This objective function attempts to balance the terminal operator's goal of fast vessel (and therefore container) throughput with a penalty term that attempts to treat different vessels fairly. Parameter f_k provides flexibility in prioritizing vessels for early processing.

We present two widely used mixed integer programs to formulate BAP. The first formulation considers the space covered by the vessel rectangles, and hence called Position Assignment Formulation (PAF). The PAF has been used to construct solution methods based on Lagrangian Relaxation in [24] and [57]. The other formulation considers the relative position of the vessel rectangles in the time space diagram, and hence is called Relative Position Formulation (RPF). Due to the relatively small number of binary variables, this formulation is generally used to solve small instances directly. Optimal solutions generated by RPF can then be used in the evaluation of heuristic methods generated.

2.2.1 Position Assignment Formulation

In PAF, we suppose that the time-space area is divided into $T \times B$ unit blocks where each block has the width of one time unit and height of one berth section. Let block (x, y) be the block for time unit $x \in [1, T]$, and berth section $y \in [1, B]$. For instance, in Figure 7, vessel 5 covers blocks $(15, 5), (16, 5), (17, 5), (18, 5), (15, 6), (16, 6), (17, 6)$, and $(18, 6)$. The following binary variables are needed in PAF:

$$\delta_{klt} = \begin{cases} 1 & \text{if vessel } k \text{ berths at location (berth section) } l \text{ at time } t, \\ 0 & \text{otherwise;} \end{cases}$$

$$\sigma_{kxy} = \begin{cases} 1 & \text{if vessel } k \text{ covers block } (x, y), \\ 0 & \text{otherwise.} \end{cases}$$

Then, PAF can be written as follows:

$$\text{Minimize} \quad \sum_{k=1}^n (c_k - a_k) + \sum_{k=1}^n f_k (c_k - d_k)^+ \quad (1)$$

subject to

$$\sum_{l=1}^{B-h_k+1} \sum_{t=1}^{T-p_k+1} \delta_{klt} = 1 \quad \forall k \quad (2)$$

$$\sum_{x=t}^{t+p_k-1} \sum_{y=l}^{l+h_k-1} \sigma_{kxy} - p_k h_k - (\delta_{klt} - 1)M \geq 0 \quad \forall k, l, t \geq a_i \quad (3)$$

$$\sum_{k=1}^n \sigma_{kxy} \leq 1 \quad \forall x, y \quad (4)$$

$$p_k + \sum_{l=1}^{B-h_k+1} \sum_{t=a_k}^{T-p_k+1} t \delta_{klt} \leq c_k \quad \forall k \quad (5)$$

$$\delta_{klt} \in \{0, 1\} \quad \forall k, l, t \quad (6)$$

$$\sigma_{kxy} \in \{0, 1\} \quad \forall k, x, y \quad (7)$$

In PAF, Constraint (2) ensures that each vessel must be berthed exactly once respecting vessel size and vessel arrival time. Constraint (3) forces that blocks occupied by each vessel rectangle must be adjacent, whereas Constraint (4) guarantees that a block can only be occupied by at most one vessel rectangle. Constraint (5) relates the earliest possible time that a vessel can leave to its berthing time and processing time.

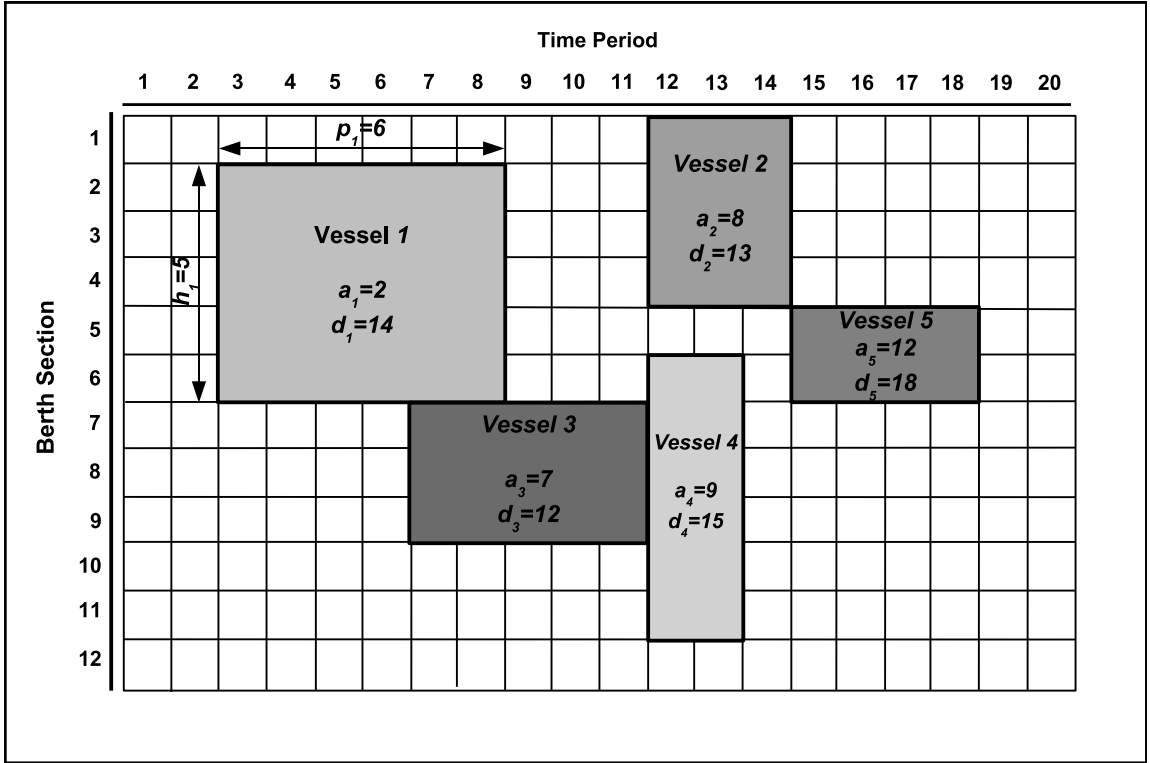


Figure 7: Representation of a berth schedule on a time-space diagram

2.2.2 Relative Position Formulation

Another standard approach to formulate an integer program for the continuous berth allocation problem is based on constraining the relative positions of vessels in the time-space representation of the problem. The primary decision variables in RPF are:

$$x_{\ell k} = \begin{cases} 1 & \text{if vessel } k \text{ berths after vessel } \ell \text{ departs,} \\ 0 & \text{otherwise;} \end{cases}$$

$$y_{\ell k} = \begin{cases} 1 & \text{if vessel } k \text{ berths completely above vessel } \ell \text{ on the time-space diagram,} \\ 0 & \text{otherwise.} \end{cases}$$

A feasible selection of these primary variables then constrain the secondary set of decisions $\{b_k, t_k, c_k\}$ for all ships k , modeled as continuous variables. The formulation is then:

$$\text{Minimize} \quad \sum_{k=1}^n (c_k - a_k) + \sum_{k=1}^n f_k (c_k - d_k)^+ \quad (8)$$

$$x_{\ell k} + x_{k\ell} + y_{\ell k} + y_{k\ell} \geq 1 \quad \forall k, \ell \in \mathcal{V} \text{ and } k < \ell \quad (9)$$

$$x_{\ell k} + x_{k\ell} \leq 1 \quad \forall k, \ell \in \mathcal{V} \text{ and } k < \ell \quad (10)$$

$$y_{\ell k} + y_{k\ell} \leq 1 \quad \forall k, \ell \in \mathcal{V} \text{ and } k < \ell \quad (11)$$

$$t_\ell \geq c_k + (x_{k\ell} - 1)M \quad \forall k, \ell \in \mathcal{V} \text{ and } k \neq \ell \quad (12)$$

$$b_\ell \geq b_k + h_k + (y_{k\ell} - 1)M \quad \forall k, \ell \in \mathcal{V} \text{ and } k \neq \ell \quad (13)$$

$$t_k \geq a_k \quad \forall k \in \mathcal{V} \quad (14)$$

$$c_k \geq t_k + p_k \quad \forall k \in \mathcal{V} \quad (15)$$

$$b_k \leq B - h_k + 1 \quad \forall k \in \mathcal{V} \quad (16)$$

$$b_k \geq 1 \quad \forall k \in \mathcal{V} \quad (17)$$

$$x_{\ell k} \in \{0, 1\}, \quad y_{\ell k} \in \{0, 1\} \quad \forall k, \ell \in \mathcal{V} \text{ and } k \neq \ell \quad (18)$$

Constraints (9) through (11) ensure that no vessel rectangles overlap. Constraints (12) and (13) ensure that the selected berthing times and berthing positions are

consistent with the definitions of $x_{\ell k}$ and $y_{\ell k}$, where M is a large positive scalar. Constraint (14) and (15) force berthing time to occur no earlier than arrival time, and departure time to occur no earlier than service completion time. Constraints (16) and (17) guarantee that all vessels fit on the berth.

We now characterize some of the features of optimal solutions to this formulation. Theorem 1 will be used in the development of our heuristic solution approach.

Definition 1 *For a given berth schedule, vessel k is said to be supported in time if $t_k = a_k$ or $t_k = t_\ell + p_\ell$ for some vessel ℓ .*

Definition 2 *For a given berth schedule, vessel k is said to be supported in space if one of the following conditions holds: $b_k = 1$, or $b_k = B - h_k + 1$, or $b_k = b_\ell + h_\ell$, or $b_k = b_\ell - h_k$ for some vessel ℓ*

Definition 3 *For a given berth schedule, vessel k is said to be packed if it is supported both in time and in space.*

Lemma 1 *In any optimal berth schedule, all vessels are supported in time.*

Proof. Any berth schedule with some vessel k which is not supported in time can be improved by moving the rectangle representing vessel k to the left on the time-space diagram. \square

Lemma 2 *There exists an optimal berth schedule where all vessels are supported in space.*

Proof. A vessel rectangle k which is not supported in space can be moved up or down on the time-space diagram without any change in the objective function until vessel k becomes supported in space. \square

Theorem 1 *There exists an optimal berth schedule where all vessels are packed.*

Proof. The proof directly follows Lemma 1 and Lemma 2. □

Refer again to Figure 7. In this example, $b_1 = 2$, $t_1 = 3$ and $c_1 = 9$. Vessel 1 is supported in space because $b_1 = b_3 - h_1$, but is not packed since $t_1 > a_1$ and there is no vessel on the berth scheduled before vessel 1. Similarly, vessel 2 is supported only in space, vessel 3 is packed, vessel 4 is supported only in time, and vessel 5 is supported neither in time nor space. Furthermore, vessel 2 and vessel 5 are tardy since $c_2 = 15 > 13 = d_2$ and $c_5 = 19 > 18 = d_5$. This solution cannot be optimal because all vessels are not supported in time.

2.2.3 Hardness of the Problem

This BAP variant is clearly *NP*-hard; instances with unit length vessels ($h_k = 1$) and no lateness penalties ($f_k = 0$) are each a machine scheduling problem with release times, no job preemption, and the objective of minimizing the sum of job completion times, known to be *NP*-hard even with a single machine ($B = 1$) [38]. The mixed integer program (9)-(17), when solved directly with off-the-shelf commercial software, can be used to find optimal solutions for small instances with up to 10 vessels in a reasonable amount of time. However, larger instances likely to be found in practice cannot be readily solved with this approach.

2.3 Lower Bound Analysis

In this section we introduce a polynomially-computable lower bound for the objective function value of the dynamic BAP formulated in the previous section. This bound is provably stronger than the linear programming relaxation bound, and is therefore of potential use in improving the computational performance of the proposed model. The relaxation used to generate the bound is somewhat similar to one used in [25] for

the static BAP with a minimum total weighted flow time objective, but our approach handles dynamic problems (with positive arrival, or release, times), and can also properly accommodate nonzero lateness penalties.

For any instance of our problem, we can construct a corresponding parallel machine scheduling problem P using the relaxation illustrated partially in Figure 8. First, each berth segment is treated as a separate parallel machine; thus, B identical machines are defined. Next, for each vessel k , we create $p_k \times h_k$ jobs, each with unit length and unit processing duration. We use a two-dimensional index (i, j) for each of these jobs, where i refers to the spatial position and j the time position. For each job (i, j) , we set its release time $r_{ij}^k = a_k + j - 1$; while such a construct does not prevent a job with time index j from being processed after a job with time index $j + 1$, it at least prevents clearly infeasible start times given the vessel arrival time. Similarly, we set a due time $d_{ij}^k = d_k - (p_k - j)$. Each job is also assigned a set \mathcal{B}_{ij}^k of feasible machines (berthing locations), limited by the berthing area boundaries: $\mathcal{B}_{ij}^k = \{b = 1, \dots, B : b \geq i, b \leq B + i - h_k\}$. Parameters $\alpha_{ij}^k = \frac{1}{p_k h_k}$ and $\beta_{ij}^k = \frac{f_k}{p_k h_k}$ are assigned to each job (i, j) for vessel k , and will be used as objective function weights. The objective is to assign each job to a feasible machine, minimizing the sum of total weighted completion time and total weighted tardiness, given by $\sum_{k,i,j} \alpha_{ij}^k c_{ij}^k + \sum_{k,i,j} \beta_{ij}^k (c_{ij}^k - d_{ij}^k)^+$, where the completion time c_{ij}^k of job (i, j) for vessel k is $t_{ij}^k + 1$ when it is processed in time period t_{ij}^k .

Theorem 2 $LB_P = \lceil C_P^* + \Phi - \Psi \rceil$ is a lower bound for BAP where C_P^* is the optimal objective function value of P constructed for BAP, $\Phi = \sum_{k=1}^n (p_k - 1)/2$, and $\Psi = \sum_{k=1}^n a_k$.

Proof. Any feasible solution of BAP generates a feasible solution for the corresponding P . In such a solution, for each job (i, j) defined for vessel k , $c_{ij}^k = c_k - p_k + j$ holds. Then, $c_{ij}^k - d_{ij}^k = c_k - p_k + j - (d_k - p_k + j) = c_k - d_k$. Hence,

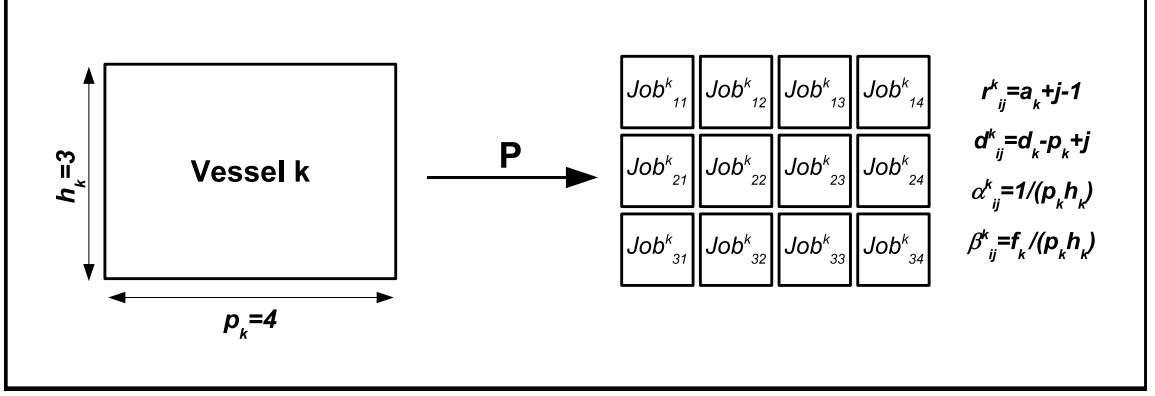


Figure 8: Construction of lower bounding scheduling problem P for a given dynamic BAP instance

$$\sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k} \alpha_{ij}^k c_{ij}^k = \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k} \frac{1}{p_k h_k} (c_k - p_k + j) = \sum_{k=1}^n \left(c_k - \sum_{j=1}^{p_k} \frac{p_k - j}{p_k} \right) \quad (19)$$

and

$$\sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k} \beta_{ij}^k (c_{ij}^k - d_{ij}^k)^+ = \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k} \frac{f_k}{p_k h_k} (c_k - d_k)^+ = \sum_{k=1}^n f_k (c_k - d_k)^+ \quad , \quad (20)$$

which implies

$$C_P = \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k} \alpha_{ij}^k c_{ij}^k + \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k} \beta_{ij}^k (c_{ij}^k - d_{ij}^k)^+ = \sum_{k=1}^n \left(c_k - \frac{\sum_{j=1}^{p_k-1} j}{p_k} \right) + \sum_{k=1}^n f_k (c_k - d_k)^+ \quad . \quad (21)$$

Since $C_P^* \leq C_P$, and $(\sum_{j=1}^{p_k-1} j)/p_k = (p_k - 1)/2$,

$$C_P^* + \sum_{k=1}^n \frac{p_k - 1}{2} - \sum_{k=1}^n a_k \leq \sum_{k=1}^n (c_k - a_k) + \sum_{k=1}^n f_k (c_k - d_k)^+ \quad (22)$$

With all integer parameters, the right hand side of Expression 22, which is the objective function of the BAP, is integer. Hence, $LB_P = \lceil C_P^* + \Phi - \Psi \rceil$ is a lower bound for the BAP. \square

Theorem 3 *P can be solved in polynomial time.*

Proof. *P* can be modeled as a minimum weight matching problem on a bipartite graph $\mathcal{G} = (\mathcal{N}_1 \cup \mathcal{N}_2, \mathcal{E})$. For each job (i, j) of vessel k , we define a unique node in \mathcal{N}_1 . We also define a unique node in \mathcal{N}_2 for each (berth, time period) pair. We then construct the edge set \mathcal{E} as follows: $\{(p, q) : p \in \mathcal{N}_1, q \in \mathcal{N}_2, b_q \in \mathcal{B}(p), r(p) \leq t_q\}$, where (b_q, t_q) is the (berth, time period) pair associated with node q , $r(p)$ is the ready time r_{ij}^k of the job associated with node p , and $\mathcal{B}(p)$ is the set of allowable berth positions \mathcal{B}_{ij}^k for that job. Edge weights c_{pq} are defined to be $c_{pq} = \alpha(p)(t_q + 1) + \beta(p)(t_q + 1 - d(p))$, where again $\alpha(p)$ and $\beta(p)$ are the appropriate constants α_{ij}^k and β_{ij}^k for the job associated with node p , and $d(p)$ is its due date d_{ij}^k . It is well known that weighted bipartite matching can be solved in polynomial time (see, for example, discussion in [4]). \square

Theorem 4 $LB_P \geq LB_{LP}$ where LB_{LP} denotes the objective function value of the optimal solution to the LP relaxation of the mixed inter program (9)-(18).

Proof. In a feasible solution of the LP relaxation, it is easy to see that berthing times t_k and positions b_k can be chosen such that any vessel rectangles may overlap one another. Therefore, in every optimal solution, $c_k = a_k + p_k \forall k \in \mathcal{V}$. Since we assume that $d_k \geq a_k + p_k$, it is clear that the LP relaxation yields the weak lower bound given by the sum of vessel processing times:

$$LB_{LP} = \sum_{k=1}^n (c_k - a_k) = \sum_{k=1}^n p_k \quad . \quad (23)$$

In P , $c_{ij}^k = t_{ij}^k + 1 \geq r_{ij}^k + 1$ and $r_{ij}^k = a_k + j - 1$ for each job j in set i of vessel k . Therefore,

$$C_P^* \geq \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k} \frac{a_k + j}{p_k h_k} , \quad (24)$$

which, by Theorem 2, implies,

$$LB_P \geq \left[\sum_{k=1}^n \sum_{j=1}^{p_k} \frac{a_k + j}{p_k} + \sum_{k=1}^n \frac{p_k - 1}{2} - \sum_{k=1}^n a_k \right] . \quad (25)$$

Rearranging terms then yields the result:

$$LB_P \geq \left[\sum_{k=1}^n \frac{p_k + 1}{2} + \sum_{k=1}^n \frac{p_k - 1}{2} \right] = \sum_{k=1}^n p_k = LB_{LP} . \quad (26)$$

□

2.4 A Meta Heuristic: The Nested Tabu Search Algorithm

The meta-heuristic term was coined in the same paper that introduced the term tabu search ([22]), and has come to be widely applied in the literature, both in the titles of comparative studies and in the titles of volumes of collected research papers (see e.g. [35], [21], [56], [10], and [5]).

A meta-heuristic refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality. The heuristics guided by such a meta-strategy may be high level procedures or may embody nothing more than a description of available moves for transforming one solution into another, together with an associated evaluation rule.

The contrast between the meta-heuristic orientation and the *local optimality* orientation is significant. The primary conception of a heuristic procedure is to envision either a clever rule of thumb or an iterative rule that terminates as soon as no solutions immediately accessible could improve the last one found. Such iterative heuristics are often referred to as descent methods, ascent methods, or local search methods. The emergence of methods that departed from this classical design constituted an important advance. Meta-heuristics in their modern forms are based on a variety of interpretations of what constitutes *intelligent* search. These interpretations lead to design choices which in turn can be used for classification purposes. The most well-known and widely used meta-heuristics are *genetic algorithms*, *simulated annealing*, and *tabu search*.

In this section, we present a meta-heuristic method based on the tabu search algorithm to solve the dynamic BAP with a linear berth structure. The well known tabu search method can be rooted to [22], and more detail can be found in [23]. Briefly, the method explores solution space by moving at each iteration to the best non-tabu neighbor of the current solution. In order to avoid cycling, solutions that were recently visited are made tabu for a number of iterations.

The heuristic we design uses a simple yet interesting method for encoding a feasible solution, and a nested search approach with two layers for exploring neighbors. The outer layer, TS_1 , searches a neighborhood defined by a berthing priority list of vessels, while the inner layer, TS_2 , searches for best vessel berth positions given a priority list. For computational efficiency, both layers of the nested search considers only packed berth schedule solution candidates at each iteration; recall that by Theorem 1, there exists an optimal solution to the problem that is packed.

2.4.1 Encoding and Decoding a Solution

We use two vectors of size n , \mathcal{B} and \mathcal{L} , to encode a feasible solution. Entries of \mathcal{B} are the berth positions of vessels. Entries of the ordered list \mathcal{L} are vessel indices, and the order determines berthing priority. The vector pair $(\mathcal{B}, \mathcal{L})$ defines a unique feasible berth schedule x as follows: beginning with the first vessel in \mathcal{L} , berth each vessel k at the berthing position given in \mathcal{B} at the later of a_k or the earliest time that the berthing time-space rectangle to be occupied by k is not occupied by any earlier berthed vessel. With appropriate data structures, this decoding operates in $O(n)$ time. Note then that all vessels are supported in time in feasible solutions defined in this way.

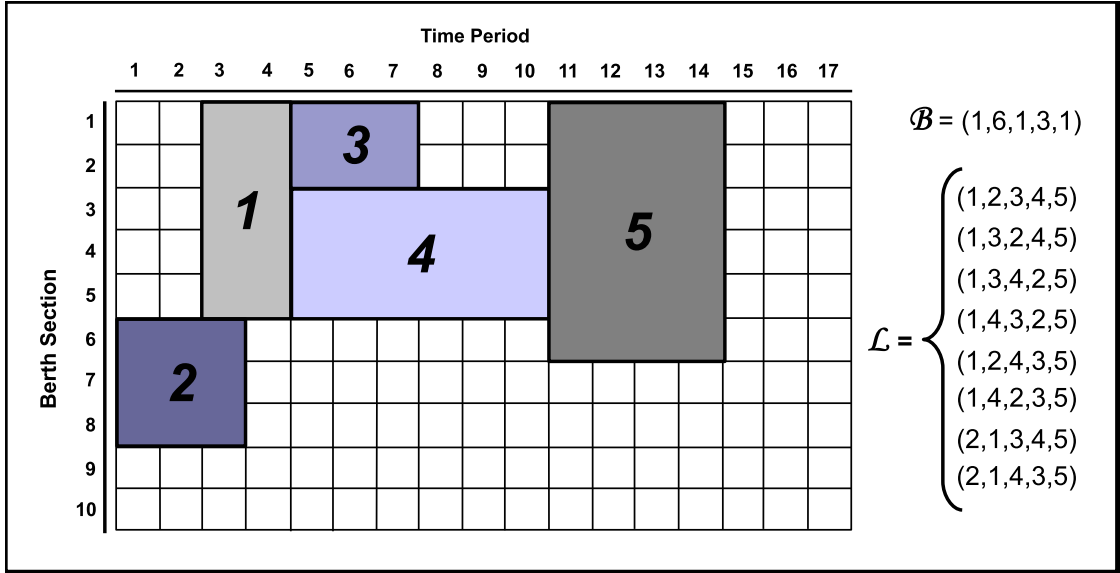


Figure 9: Multiple priority lists may correspond to a single berth schedule

Although there is a one-to-one mapping from \mathcal{B} and \mathcal{L} to a feasible berth schedule x , the reverse is not true. While a given schedule uniquely defines a set of berth positions \mathcal{B} , it may correspond to many different vessel priority lists. Therefore, for any given berth schedule x , we define a unique vessel priority list called the *primal*

$list, \mathcal{L}^P(x)$. The primal list ordering of vessels is created by sorting the vessel indices according to their berthing times t_k , breaking ties in favor of smaller berth positions b_k . Figure 9 shows an example where a single berth schedule is encoded by eight different vessel priority lists, where the primal list is $(2, 1, 3, 4, 5)$.

We will also mark each feasible berth schedule encountered during the search with a large scalar mark $M(x)$ determined by an arbitrary function $g(\{b_k\}, \{t_k\})$. Figure 10 summarizes the relationships between vessel priority lists, the berthing position vector, a feasible berth schedule, and its corresponding scalar M .

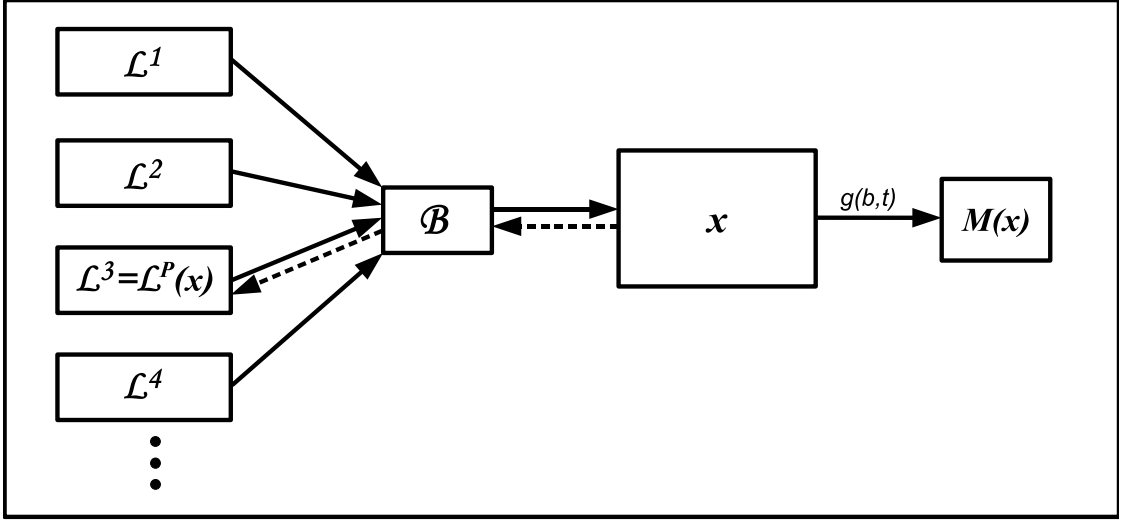


Figure 10: Encoding and decoding a berth schedule

2.4.2 Initial Solution

It is possible to generate a feasible solution x^0 using only a vessel priority list \mathcal{L} by constructing a berth position vector using a two-dimensional hierarchical first-fit (FF) rule. Such a first-fit rule places vessel rectangles k on the time space diagram one by one in the order specified by \mathcal{L} at the earliest possible berthing time (i.e., the smallest time $t \geq a_k$ with h_k available contiguous berth sections), and at the lowest index berth section at that time. Note that this first-fit rule yields a packed solution.

Thus, initial solutions can be generated using any method for creating a sorted \mathcal{L} . In our approach, we use the following initial vessel priority lists:

- First-come, first-served (FCFS): \mathcal{L}^{FCFS} is sorted by non-decreasing vessel arrival times a_k
- Earliest due date (EDD): \mathcal{L}^{EDD} is sorted by non-decreasing vessel due times d_k
- Modified earliest due date (mEDD): \mathcal{L}^{mEDD} is sorted by non-decreasing penalty-weighted due times $\frac{d_k}{f_k}$

The initial solution is chosen to be the one of the above with the smallest objective function value. Since the rules for generating the priority lists are simple but intuitive, this approach for building an initial solution mimics decision rules-of-thumb that a terminal operator might use to generate berth schedules.

2.4.3 Fundamentals of the Nested Search

Given some \mathcal{L} and \mathcal{B} , and their corresponding berth schedule x , a neighboring solution can be defined by some change to \mathcal{B} , or to \mathcal{L} , or to both vectors. We define a nested approach that explores changes to \mathcal{B} given a fixed $\mathcal{L}^P(x)$ as an inner search TS_2 , while an outer search TS_1 explores changes to \mathcal{L} . The first berth position vector \mathcal{B} created for any new \mathcal{L} in the outer search is the one defined by the hierarchical FF rule.

An example set of nested search steps is illustrated in Figure 11. In this instance, $a_1 = 1$, $a_2 = 2$, $a_3 = 4$, $a_4 = 3$, and $a_5 = 8$ and all vessel due times are 17. Part *A* of the figure depicts the berth schedule generated by the first-fit rule berth positions corresponding to $\mathcal{L} = (1, 2, 3, 4, 5)$. If we conduct a outer search TS_1 move that swaps vessels 3 and 4 in the vessel list yielding $\mathcal{L} = (1, 2, 4, 3, 5)$, we reach the better solution illustrated in Part *B*. Next, we conduct two inner search TS_2 moves to generate the schedules in Part *C* and *D*. In Part *C*, we illustrate a move where we change only the berth position of vessel 2 from 7 to 10. A second move yields the improved solution

in Part *D*, where we again change only the berth position of vessel 5 from 1 to 8. It can be shown that the schedule in Part *D* is an optimal schedule for this instance. Note that all vessels in all berth schedules given in figure 11 are packed.

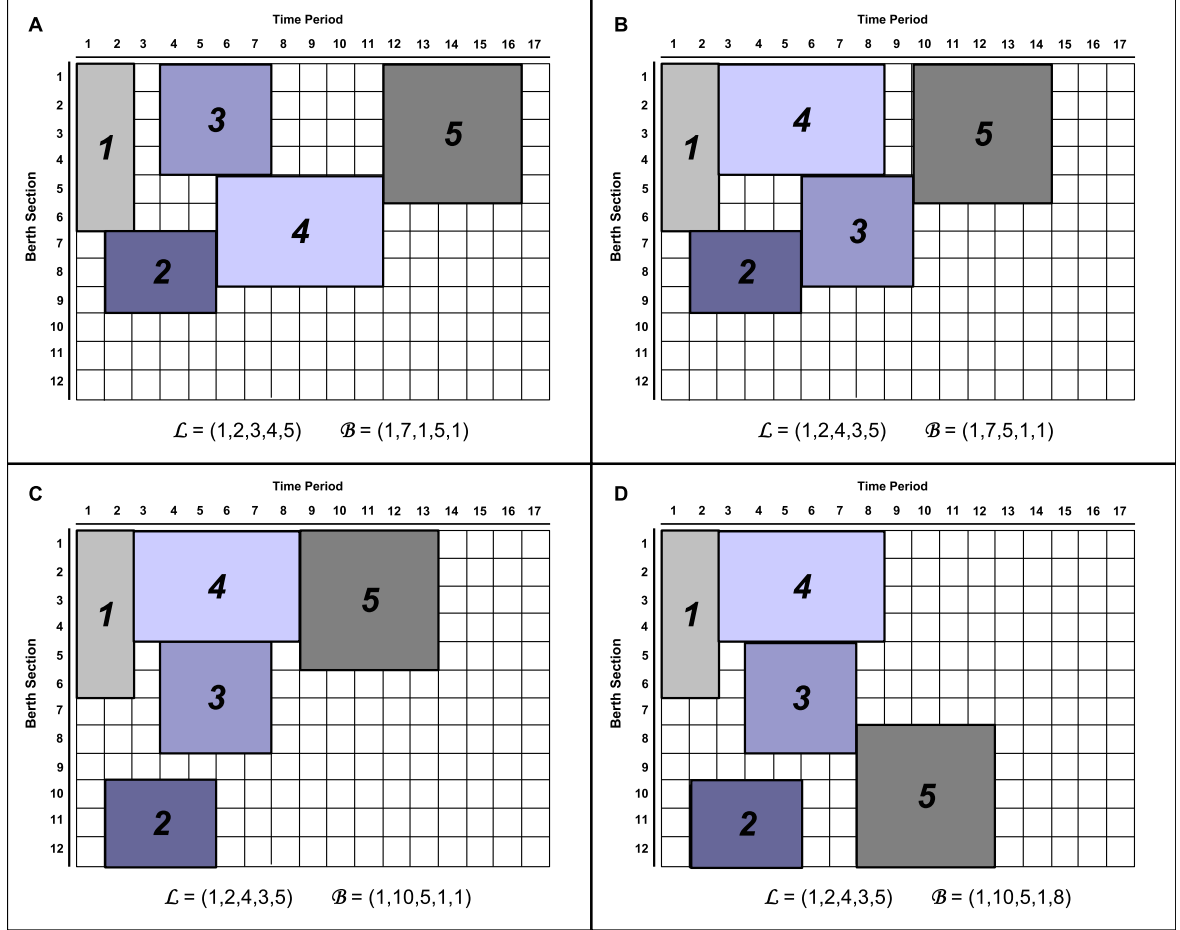


Figure 11: Illustration of moving from one solution to another by modifying \mathcal{L} and \mathcal{B}

We now outline the major steps. At each iteration of TS_1 , we modify the current \mathcal{L} using single swap moves within a randomized neighborhood. Given a list \mathcal{L} , define the (p, r, q) randomized neighborhood of lists $\mathcal{N}_{\mathcal{L}}(p, r, q, \mathcal{L})$ as all lists generated by a single swap in \mathcal{L} of the position of one of q randomly-selected vessels (denoted vessel k) with the position of one of p randomly-selected vessels from among the r closest neighbor vessels to k in \mathcal{L} . Closeness in \mathcal{L} is measured as the absolute difference in

position in the list.

Each candidate neighbor list generated in an iteration of TS_1 is initially assigned a berth position vector \mathcal{B} by the FF rule (corresponding to a packed solution). These pairs now form a set of candidate moves, each with a corresponding berth schedule x , mark $M(x)$, and primal vessel list $\mathcal{L}^P(x)$. Each non-tabu candidate move is now considered in sequence, where a tabu move is one whose mark $M(x)$ is currently in the tabu list. If the mark $M(x)$ has not yet been considered during the current TS_1 iteration, we initiate the inner search TS_2 with x as the initial solution.

In an iteration of TS_2 , we generate a set of candidate moves using changes to a single entry in the berth position vector \mathcal{B} corresponding to the current solution y while keeping the primal list $\mathcal{L}^P(y)$ fixed. Note that the initial solution used in an iteration of TS_2 is packed, and we will restrict neighbor candidates to also be packed in each iteration of TS_2 . Let \mathcal{V}_S be the set of all vessels that have adjacent unoccupied berth sections in schedule y for the entire vessel berthing time; note that these sections may be above or below the vessel in the time-space diagram, but not both. For example, in part *A* of Figure 11, $\mathcal{V}_S = \{2, 4, 5\}$. Given \mathcal{B} , we define the (s, \mathcal{V}_S) randomized neighborhood of berth position vectors $\mathcal{N}_{\mathcal{B}}(s, \mathcal{V}_S, \mathcal{B})$ as all those generated by changing the berth position of one of s randomly-selected vessels from the set \mathcal{V}_S ; for the selected vessel, the berth position is shifted up or down in the direction of the adjacent unoccupied berth sections as far as possible before encountering another berthed vessel. Note that the shifted vessel is still supported in space. For each $\mathcal{B}_j \in \mathcal{N}_{\mathcal{B}}(s, \mathcal{V}_S, \mathcal{B})$, a feasible berth schedule is decoded using the vector pair $(\mathcal{B}_j, \mathcal{L}^P(y))$; recall the discussion in Section 2.4.1. Thus, TS_2 moves from one packed solution to another.

The importance of primal vessel lists for the nested search is now established.

Lemma 3 *If $\mathcal{L}^P(y)$ is the primal list for current berth schedule y , every candidate move in an iteration of TS_2 yields a new berth schedule with an objective value smaller*

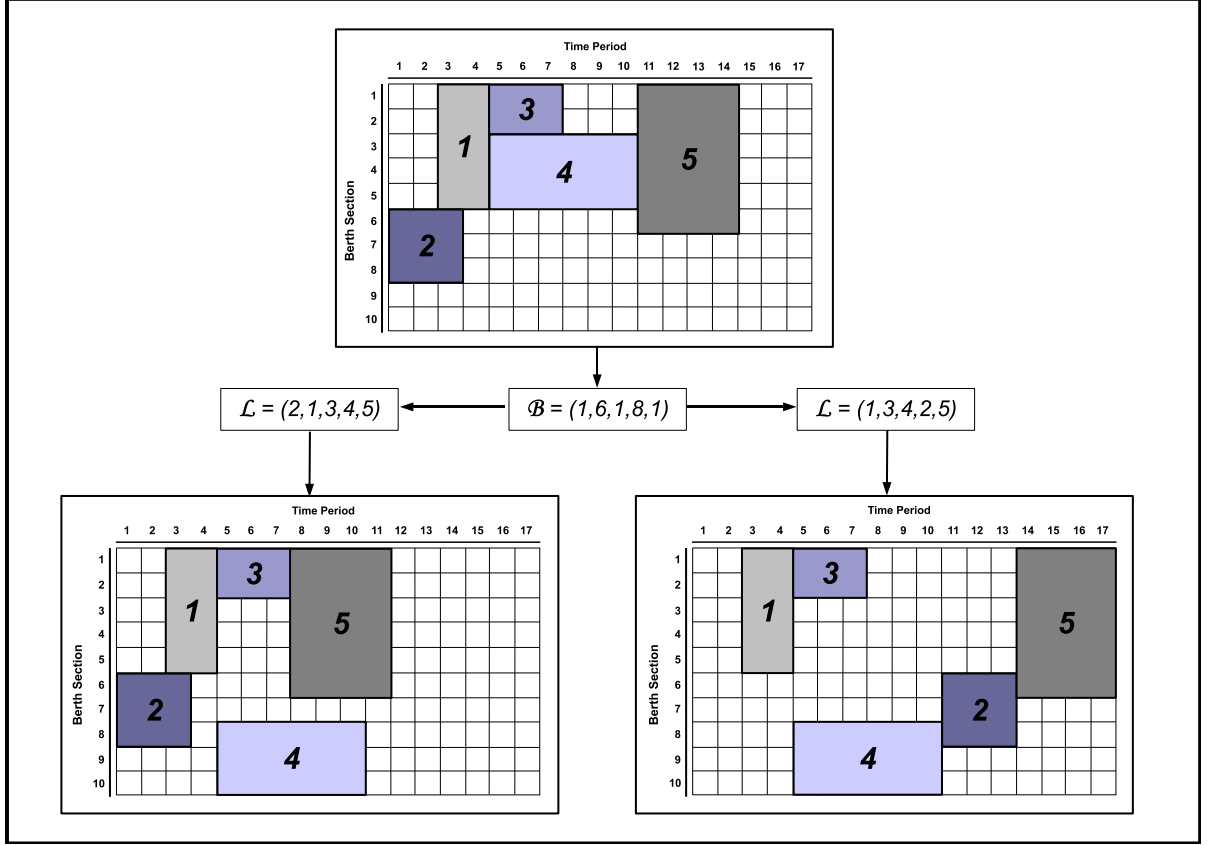


Figure 12: The importance of using the primal list in the inner search, TS_2

or equal to the current.

Proof. Let v be the vessel whose berth position is shifted in the candidate move, and let t_v be its berthing time in the current solution y . By definition, each vessel k currently berthed at some time $t_k < t_v$ precedes v in the list $\mathcal{L}^P(y)$ since it is the primal list. Therefore, these vessels will have the same berthing times in the new solution. The same is true for each vessel $\ell \neq v$ currently berthed at $t_\ell = t_v$, since the shifted berth position of v is restricted by these vessels. Vessel v and any vessel m where $t_m > t_v$ may be berthed at the same time, or an earlier time after the shift of vessel v . Clearly, vessel v can berth no later than its current berthing time t_v , since the berthing times and positions of precedent vessels in $\mathcal{L}^P(y)$ do not change. By simple induction, it can be shown that since the berthing time of a vessel m is

constrained only by its arrival time a_m and the completion time of some precedent vessel in $\mathcal{L}^P(y)$, and precedent vessel berthing times are always no later than their current berthing times, each vessel m is berthed at a time no later than its current. Thus, the lemma holds. \square

Figure 12 illustrates these ideas. Suppose vessel 1 has $a_1 = 3$, vessel 4 has $a_4 = 5$, and all other vessels arrive at time 1. In the example, if the candidate move is to shift the berth position of vessel 4 downward by setting $b_4 = 8$, we reach the schedule on the left part of the figure using the primal list $\mathcal{L}^P = (2, 1, 3, 4, 5)$. Importantly, however, if the list corresponding to the current solution were not the primal, such a candidate move may be decoded into a solution with a worse objective. For example, list $\mathcal{L} = (1, 3, 4, 2, 5)$ also corresponds to the initial solution. However, the same berth position shift in this case yields the solution on the right part of the figure, with a clearly higher objective function value.

The tabu list in TS_2 prevents reverse berth shift moves (defined as shifting a vessel back to its previous berth position) for a random number of iterations θ_2 , where θ_2 is distributed discrete uniform on $[\theta_2^{min}, \theta_2^{max}]$. At the completion of TS_2 , a best (lowest objective function value) berth position vector and corresponding best solution is identified for the current candidate move of TS_1 . After all candidate TS_1 moves have been evaluated for the current iteration, the best (lowest objective function value) move is selected as the new current solution x , with its corresponding vessel list \mathcal{L} and berth position vector \mathcal{B} . The mark $M(x)$ of this solution is placed into the TS_1 tabu list for a random number of iterations θ_1 , where θ_1 is discrete uniform on $[\theta_1^{min}, \theta_1^{max}]$.

2.4.4 Steps of the Nested Tabu Search Algorithm

In this section, we provide step-by-step detail for the proposed nested search. The overall procedure is illustrated in Figure 13. Let x^0 be the initial berth schedule solution for TS_1 , and let y^0 be the initial solution for TS_2 . Furthermore, let x , x^* and y , y^* denote the current and the best solutions found by TS_1 and TS_2 respectively. Let $C(x)$ represent the objective function value for solution x .

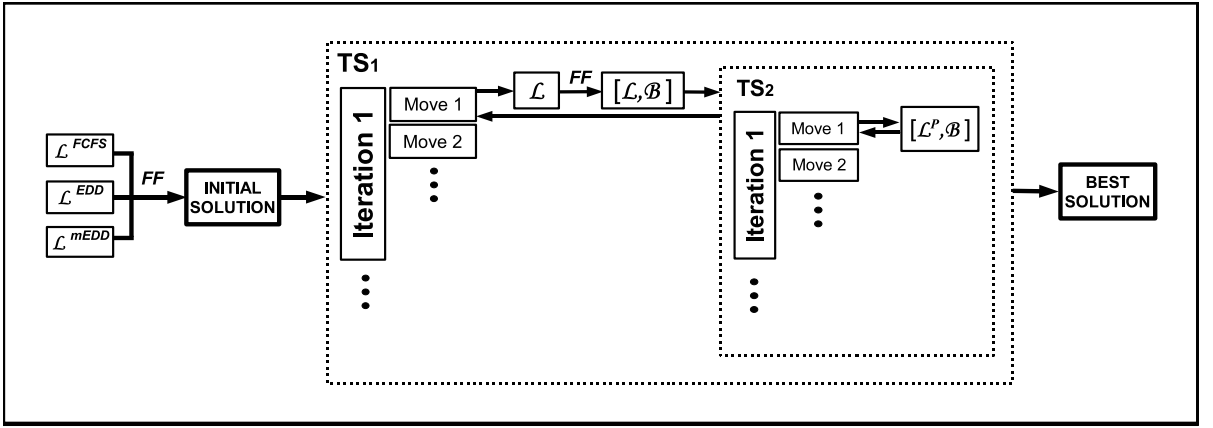


Figure 13: Schematic Representation of the Nested Tabu Search Approach

2.4.4.1 Steps of $TS_1(x^0, x^*)$:

Step 1: Initialization. Set p , r , and q . Let x^0 be the lowest-cost solution found from among the three initial vessel lists, \mathcal{L}^{FCFS} , \mathcal{L}^{EDD} , and \mathcal{L}^{mEDD} , each with a berth position vector determined by the FF rule. Let \mathcal{L} and \mathcal{B} be the vessel list and berth position vector yielding x^0 . Set $x = x^* = x^0$. Initialize counters $t_1^T = 0$ and $t_1^S = 0$, which denote the total number of iterations, and the number of successive iterations with no improvement in the best objective function value respectively. Set the value of t_1^M , which is the maximum number of iterations allowed with no improvement.

Step 2: Neighborhood Search. Set $t_1^T = t_1^T + 1$. For each candidate list $\mathcal{L}_i \in$

$\mathcal{N}_{\mathcal{L}}(p, r, q, \mathcal{L})$, construct an associated \mathcal{B}_i by the FF rule, and let x_i be the associated berth schedule solution. For each solution x_i such that $M(x_i)$ is not in the tabu list and has not yet been considered during this step, run $TS_2(x_i, x_i^*)$. Let a be the move i with the smallest value of $C(x_i^*)$.

Step 3: Solution Updates. Set $x = x_a^*$, and let $\mathcal{L} = \mathcal{L}_a$. If $C(x) < C(x^*)$, set $x^* = x$ and $t_1^S = 0$. Otherwise, set $t_1^S = t_1^S + 1$. Update the tabu list, adding $M(x)$ for the appropriate random number of iterations and removing any marks whose duration has expired. If $t_1^S < t_1^M$, go to Step 2. Else, stop. The best solution found is x^* .

2.4.4.2 Steps of $TS_2(y^0, y^*)$:

Step 1: Initialization. Set the value of s . Set $y = y^* = y^0$, and let \mathcal{B} be the berth position vector for solution y . Initialize counters $t_2^T = 0$ and $t_2^S = 0$, which denote the total number of iterations, and the number of successive iterations with no improvement the best objective function value respectively. Set the value of t_2^M , which is the maximum number of iterations allowed with no improvement.

Step 2: Neighborhood Search. Set $t_2^T = t_2^T + 1$. Construct set V_S for y . Consider each move y_j decoded from non-tabu berth position vector $\mathcal{B}_j \in \mathcal{N}_{\mathcal{B}}(s, \mathcal{V}_S, \mathcal{B})$ combined with primal list $\mathcal{L}^P(y)$, and let b be the move j with the smallest value of $C(y_j)$.

Step 3: Solution Update. Set $y = y_b$. If $C(y) < C(y^*)$, set $y^* = y$ and $t_2^S = 0$. Otherwise, set $t_2^S = t_2^S + 1$. Update the tabu list by inserting the reverse berth position move associated with solution y_b into the tabu list for the appropriate random number of iterations, and removing reverse moves whose duration has expired. If $t_2^S < t_2^M$, go to Step 2. Else, stop. The best solution found is y^* .

2.4.5 Multi-start Version

Multiple starts or related “kick” moves are widely used techniques in metaheuristic search algorithms. They attempt to direct the search to unexplored areas of the solution space when the algorithm gets trapped by a local optimum.

We develop and test a multi-start version of the nested tabu search described earlier. Instead of running the search with a large limit on successive non-improving iterations, we restart R times with a smaller limit. To generate the R initial solutions, we use a diversified search to create R vessel lists. This is done using a modified version of TS_1 that does not call TS_2 ; each candidate list is only evaluated by the berth schedule resulting from the FF rule. Furthermore, we select relatively smaller values for q and p to increase the randomization in neighborhood and hence increase diversification in the search. The R best lists found in this search are stored and used as the R initial lists in the multi-start version of the heuristic.

2.5 Computational Experiments

Computational experiments on randomly generated sets of test instances were conducted to evaluate the performance of the algorithm proposed. Six sets of test problems were used, each containing 10 different randomly generated instances. The first three of these sets include relatively small problems for a terminal with $B = 12$, and $n = 10, 12$, and 14 vessels respectively. The next three problem sets contain larger instances with $B = 20$ and with $n = 20, 25$, and 30 vessels. For both sets of instances, the vessel lengths are random and range from 2 to 6 berthing positions: $h_i \sim U^D[2, 6]$, where $U^D[a, b]$ is the discrete uniform distribution on the closed interval $[a, b]$. The vessel processing times vary depending on vessel length, and are determined as follows. For vessel k with length h_k , we generate h_k integers $p_k^1, \dots, p_k^{h_k}$ randomly where $p_k^i \sim U^D[1, 4]$ for $h_k = 2$, $p_k^i \sim U^D[1, 5]$ for $h_k \in \{3, 4\}$, and $p_k^i \sim U^D[1, 6]$ for $h_k \in \{5, 6\}$. We then let $p_k = \max_i \{p_k^i\}$. Constructing processing times in this

way models the idea that different sections of the ship may require different processing times, but the berthing duration of the ship can be no smaller than the largest processing time for any ship section. Finally, we distribute the vessel arrival times in the instances, but consider only the more difficult cases when many vessels are in port ready to be berth simultaneously; more discussion on this point follows in the next paragraph. In the first three problem sets, the vessel k arrival time is given by $a_k \sim U^D[1, 10]$, while in the second three sets, $a_k \sim U^D[1, 20]$. Vessel due times are also generated randomly, and are determined by adding a random multiple of a vessel's processing time to its arrival time: $d_k = a_k + Kp_k$ where $K \sim U^D[1, 3]$. Lateness penalties also vary by vessel, and are given by $f_k \sim U^D[3, 5]$.

Table 1: Statistics for test BAP instances used

Set	B	n	Average RU_{max}	Average RU_{avg}
1	12	10	2.175	1.236
2	12	12	2.483	1.389
3	12	14	3.008	1.635
4	20	20	1.615	0.822
5	20	25	1.800	0.997
6	20	30	2.495	1.296

Vessel arrival times in our problem context are like release times in classic machine scheduling problems. It is well-known that while release times generally increase theoretical problem complexity, they can often lead to simple-to-solve practical instances. In the dynamic BAP, suppose for example that the k^{th} arriving vessel has $a_k \geq a_{k-1} + p_{k-1}$. Clearly, serving the vessels in first-come, first-served order is always optimal. Therefore, to measure the practical difficulty of the instances used in this computational study, we use the concept of *required utilization* (RU). Suppose that every vessel were berthed at its arrival time, ignoring available berth positions. Then, at every time period t , one can determine the minimum number of required berth sections for the vessels by adding up the lengths of vessels berthed at that time; note

that this includes all vessels k for which $t \in [a_k, a_k + p_k - 1]$. RU_t is then the ratio of this minimum to B . Let RU_{max} for an instance be the maximum value of RU_t , and let RU_{avg} be the average RU_t from time 1 until time $\max_{k \in \mathcal{V}} \{a_k + p_k - 1\}$. If $RU_{max} > 1$, then that at least one vessel in the instance cannot be berthed at its arrival time. The higher the value of RU_{avg} , the more likely that many vessels will wait before berthing and may even experience completion time delay past their due times. We report the average RU_{max} and RU_{avg} for the instances in each set of problems in Table 1. For sets of instances with higher values of these statistics, we expect the problems to be more difficult to solve.

The parameter values for each layer of the nested tabu search were determined by experimentation, and are presented in Table 2 for both the single-start (TS) and the multi-start (mTS) versions. Note that the parameters controlling the outer search neighborhood $\mathcal{N}_{\mathcal{L}}(p, r, q, \mathcal{L})$ indicate that each of $q = 5$ randomly-selected vessels will be swapped in the list with $p = 5$ nearby vessels in each iteration, but the TS neighborhood with $r = 10$ results in more search diversification than the mTS neighborhood which only considers the 5 nearest neighbors. Similarly, the outer search tabu duration parameters lead to longer tabu durations for TS , also increasing diversification. In the inner layer, the same parameter values are used for both versions.

Table 2: Tabu search parameters used for BAP computational experiments

Parameter	TS	mTS
R	-	10
p	5	5
r	10	5
q	5	5
t_1^M	$50n$	$5n$
θ_1^{min}	$n/2$	$n/4$
θ_1^{max}	n	$n/2$
s	$\max\{2, V /3\}$	$\max\{2, V /3\}$
t_2^M	$n/2$	$n/2$
θ_2^{min}	2	2
θ_2^{max}	5	5

Instances will be numbered in the tables using two digit numbers xy , where x corresponds to the set and y indicates the individual instance number. The cost of the initial solution for each of the test instances, and the initial list used to construct it, can be found in Table 3 and 4.

Table 3: Test results for small BAP instances

Instance	IS List	IS	LB_{LP}	LB_P	LB_{MIP}	UB_{MIP}	TS	mTS	Imp. (%)
10	EDD	108	45	67	90	90	90	90	100
11	EDD	62	41	49	54	54	54	54	100
12	FCFS	200	46	110	159	159	159	159	100
13	EDD	96	44	53	74	74	74	74	100
14	FCFS	59	40	45	54	54	54	54	100
15	mEDD	105	46	61	81	81	81	81	100
16	EDD	71	40	47	60	60	60	60	100
17	EDD	139	51	108	138	138	138	138	100
18	mEDD	197	49	89	131	131	131	131	100
19	EDD	123	45	69	115	115	115	115	100
20	mEDD	173	54	87	131	131	131	131	100
21	mEDD	240	50	103	118	170	170	170	57
22	mEDD	166	53	85	123	123	123	123	100
23	FCFS	193	54	84	124	124	124	124	100
24	EDD	93	47	66	79	79	79	79	100
25	FCFS	144	55	84	121	121	121	121	100
26	FCFS	155	58	93	120	120	120	120	100
27	FCFS	275	56	107	164	164	164	164	100
28	EDD	137	57	91	117	117	117	117	100
29	FCFS	160	52	82	119	119	119	119	100
30	mEDD	295	60	112	108	156	156	156	76
31	FCFS	359	61	194	157	249	247	247	68
32	FCFS	206	61	95	133	133	133	133	100
33	mEDD	410	61	190	136	254	250	250	73
34	FCFS	252	62	115	87	170	166	166	63
35	FCFS	410	52	150	126	219	213	213	76
36	EDD	198	59	101	100	142	142	142	58
37	FCFS	185	60	93	130	130	130	130	100
38	mEDD	446	65	234	141	334	332	332	54
39	EDD	529	65	253	142	352	329	329	72

Table 3 and Table 4 also present the two polynomially-computable lower bounds, LB_{LP} and LB_P . Theorem 4 proves that $LB_P \geq LB_{LP}$, and the empirical results show that LB_P generally improves LB_{LP} substantially. LB_P is on average 54%, 64%, and 152% higher the LB_{LP} for small instances (1y, 2y, and 3y) respectively. For large instances (4y, 5y, and 6y), the average improvement is 14%, 44%, and 115% respectively.

The RPF was applied to all of the instances in sets 1, 2, and 3 using CPLEX with default parameters on an Intel 2.4 GHz Pentium processor running Linux with 2 GB memory. CPLEX solved optimally all instances in set 1 in an average of 77 seconds, and 9 out of 10 instances in set 2 in an average of 1.7 hours. However, only 2 of the instances in set 3 could be solved optimally, and the average run time for these instances was approximately 4 hours. For unsolved instances, CPLEX ran for approximately 13 hours on average before stopping due to insufficient memory. The best lower bound (LB_{MIP}) and the objective function value for the best integer solution found (UB_{MIP}) by the MIP for problems in sets 1, 2, and 3 are also presented in Table 3.

The objective function values for the solutions found by TS and mTS are also presented in Table 3 and Table 4. Note that for the small instances in Table 3, TS and mTS always found berth schedules with the same cost. Furthermore, they always found the optimal solutions for instances where the optimal solution was obtained by MIP. For the remaining 9 instances where optimality cannot be proved, the algorithms found a better solution than the MIP in 6 instances, and found an equal cost solution for the remaining 3 instances. The last column in Table 3 measures the percentage improvement (reduction) in the optimality gap computed between the best lower bound (maximum of LB_P and LB_{MIP}) and the initial solution versus the gap computed for the tabu search solution. Note that LB_P provides better bounds for problems that cannot be solved optimally for all problems in set 3. It appears, therefore, that as problems size increase, the lower bounding method described in Section 2.3 becomes quite valuable. For cases where the optimal solution is found, the gap improvement percentage is 100%, which indicates that the gap is completely removed. The average gap improvement generated by the tabu search is 91% over all instances in sets 1, 2, and 3, and 69% for the ones not solved to provable optimality by CPLEX. Recall that the initial solution is the best found by simple, practical

scheduling rules.

Table 4: Test results for large BAP instances

Instance	IS List	IS	LB_{LP}	LB_P	TS	mTS	TS Imp. (%)	mTS Imp. (%)
40	FCFS	157	82	102	112	110	82	85
41	EDD	199	92	108	124	121	82	86
42	EDD	168	87	112	134	134	61	61
43	EDD	129	87	102	119	119	37	37
44	FCFS	77	70	71	76	76	17	17
45	EDD	149	91	97	111	111	73	73
46	FCFS	197	90	106	132	132	71	71
47	FCFS	185	93	106	129	129	71	71
48	FCFS	103	89	90	96	96	54	54
49	FCFS	150	86	93	112	112	67	67
50	FCFS	455	119	200	248	245	81	82
51	EDD	143	93	95	112	112	65	65
52	FCFS	427	108	175	226	224	80	81
53	EDD	338	116	156	184	184	85	85
54	mEDD	543	114	280	366	366	67	67
55	FCFS	254	114	145	173	172	74	75
56	FCFS	221	116	135	153	153	79	79
57	EDD	88	78	80	82	82	75	75
58	FCFS	293	116	148	175	175	81	81
59	mEDD	358	115	171	223	223	72	72
60	EDD	523	123	256	331	333	72	71
61	EDD	771	134	377	470	470	76	76
62	mEDD	1262	149	715	866	859	72	74
63	FCFS	630	139	288	413	408	63	65
64	EDD	466	126	198	228	225	89	90
65	FCFS	539	126	245	336	336	69	69
66	mEDD	673	145	278	392	392	71	71
67	FCFS	579	144	260	370	370	66	66
68	EDD	250	128	146	174	174	73	73
69	FCFS	304	123	161	208	206	67	69

Test results for the larger instances in sets 4, 5, and 6 are presented in Table 4. TS and mTS found a solution with the same cost for 20 of these instances. For 9 of the remaining, mTS found a better solution, and for only one instance the berth schedule found by TS had a lower cost. Since these instances are too large to run on CPLEX, only LB_P is presented, and is used to calculate gaps. For all instances, significant gap improvements were generated by the tabu search heuristics. The average improvement was calculated as 70%, similar to the same statistic for small instances not solved to provable optimality by CPLEX. Furthermore, the amount of improvement seems fairly consistent from the set 4 instances to the set 6 instances.

Average improvement is 62%, 76%, and 72% for sets 4,5, and 6, respectively for both TS and mTS . This indicates that the performance of the tabu search algorithms proposed do not appear to decline as problem size increases.

Table 5: Average computational times (in sec.) for test BAP instances

Set	n	MIP	Best TS	Total TS	Best mTS	Total mTS
1x	10	77	<1	2	1	2
2x	12	10235	<1	4	1	3
3x	14	40472	1	6	1	4
4x	20	-	3	16	6	13
5x	25	-	14	37	12	35
6x	30	-	56	105	39	99

Finally, Table 5 presents the computational times in seconds observed during tests. With the parameters used, it appears that mTS finds slightly better solutions in slightly less amount of total run time compared to TS . For instances with 10 to 14 vessels, optimal solutions are generated within 1 second while it takes hours with the MIP. Best solutions for large problems with 30 vessels are found within 1 minute. We also observe an exponential increase in average run time as the problem size increases. This is a direct result of the nested approach. However, we believe that the largest problems solved in these test instances represent the largest problems that might need to be solved in practice, and are likely to actually be more difficult.

CHAPTER III

THE MULTIPLE BERTH ALLOCATION PROBLEM

3.1 Introduction

In many terminals, there may exist more than one continuous linear berth structure with the capability to handle multiple vessels of different sizes simultaneously. If a terminal has such a berth structure and if some of the vessels calling can dock any one of these berths, then the multiple berth allocation problem arises. This problem arises, for example, in terminals with a long berthing area divided into two or more sections because of kinks or physical obstacles which restrict quay crane or yard operations. As in the BAP, we assume that each vessel requires a predetermined time for unloading and loading of containers. However, these processing times may depend on the berth selected due to the variation in the speed of equipment used at each berth and the distance to the yard stack. The MBAP is defined as the problem of assigning vessels to berths and determining the berthing position and time within the assigned berth to optimize some performance metric. Both dynamic and static variants of the MBAP can be defined depending on the presence of vessels at the time of decision making. In this chapter, we study the dynamic variant of MBAP where arrival of vessels at different times during the planning horizon is allowed.

It is easy to see that the MBAP is an NP -hard problem, since the BAP is already NP -hard. To the best of our knowledge, no other study on the MBAP exists in the literature even though it is a direct extension of the BAP.

3.2 Problem Formulation

In the MBAP, we consider a number of berths and a set of container vessels that need to be moored at one of these berths. Berth set is denoted by \mathcal{M} where $m = |\mathcal{M}|$ and \mathcal{V} represents the vessel set where $n = |\mathcal{V}|$ is the total number of vessels. We discretize berth q into B_q equal size sections. Section sizes are equal for all berths. Hence, the length of a berth can be represented by the number of sections it has. We also discretize time and assume an infinite planning horizon. For each vessel $k \in \mathcal{V}$, we define:

h_k : length of vessel k measured in number of required berth sections,

p_k^q : processing time of vessel k at berth q ,

p_k^* : minimum processing time required by vessel k ($p_k^* = \min_q \{p_k^q\}$),

a_k : arrival time of vessel k ,

d_k : due time of vessel k ($d_k \geq a_k + p_k^*$),

f_k : lateness penalty for vessel k ,

q_k : berth number that vessel k is assigned,

b_k : berthing position of vessel k ,

t_k : berthing time of vessel k ,

c_k : the earliest time that vessel k can depart.

Similar to the BAP, a feasible solution for the MBAP can be represented by a set of time-space diagrams, one for each berth, where each horizontal axis measures time and each vertical axis represents berth sections. In such a representation, a vessel can be represented by a rectangle on one of the diagrams whose length is its processing time depending on berth assignment and height is its length. Given known $\{h_k, p_k^1, \dots, p_k^m, a_k, d_k, f_k\}$, the optimization problem is then to determine berth assignment q_k , berthing position b_k and berthing time period t_k for each vessel k . If we say vessel k is berthed at section b_k of berth q_k at time t_k , we mean berth sections

$[b_k, b_k + h_k - 1]$ of berth q_k are occupied by vessel k for time periods $[t_k, t_k + p_k^{q_k} - 1]$. This also means that once a vessel is moored, its berth and position cannot be changed during service, and no preemption is allowed.

We use the same objective function as in the BAP introduced before, which to minimize $\sum_{k \in N} (c_k - a_k) + \sum_{k \in N} f_k (c_k - d_k)^+$. Then, the PAF and the RPF constructed for the BAP can be modified for the MBAP as described in the following sections.

3.2.1 Position Assignment Formulation

In the PAF formulation of the MBAP, the following binary variables are used:

$$\delta_{qkt}^k = \begin{cases} 1 & \text{if vessel } k \text{ berths at location (berth section) } l \text{ of berth } q \text{ at time } t, \\ 0 & \text{otherwise;} \end{cases}$$

$$\sigma_{qxy}^k = \begin{cases} 1 & \text{if vessel } k \text{ covers block } (x, y) \text{ of berth } q, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the PAF for the MBAP can be written as:

$$\text{Minimize} \quad \sum_{k=1}^n (c_k - a_k) + \sum_{k=1}^n f_k (c_k - d_k)^+ \quad (27)$$

subject to

$$\sum_{q=1}^m \sum_{l=1}^{B_q-h_k+1} \sum_{t=1}^{T-p_k^q+1} \delta_{qlt}^k = 1 \quad \forall k \quad (28)$$

$$\sum_{x=t}^{t+p_k^q-1} \sum_{y=l}^{l+h_k-1} \sigma_{qxy}^k - p_k^q h_k - (\delta_{qlt}^k - 1)M \geq 0 \quad \forall k, q, l, t \geq a_k \quad (29)$$

$$\sum_{k=1}^n \sigma_{qxy}^k \leq 1 \quad \forall q, x, y \quad (30)$$

$$\sum_{q=1}^m \sum_{l=1}^{B-h_k+1} \sum_{t=a_k}^{T-p_k^q+1} (t + p_k^q) \delta_{qlt}^k \leq c_k \quad \forall k \quad (31)$$

$$\delta_{qlt}^k \in \{0, 1\} \quad \forall k, q, l, t \quad (32)$$

$$\sigma_{qxy}^k \in \{0, 1\} \quad \forall k, q, x, y \quad (33)$$

Note that the PAF for the MBAP defines blocks for each berth separately. Namely, we construct block (q, x, y) for each berth q , time period x and berth section y , and prevent mooring vessels in multiple berths using the summation limits in Constraint (28) and adjacency restriction imposed by Constraint (29).

3.2.2 Relative Position Formulation

In addition to the binary variables used in the RPF of the BAP to prevent vessel overlapping, the following variable set is defined for the MBAP to keep track of berth assignments:

$$w_k^q = \begin{cases} 1 & \text{if vessel } k \text{ moors at berth } q, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the problem can be formulated as follows:

$$\text{Minimize} \quad \sum_{k=1}^n (c_k - a_k) + \sum_{k=1}^n f_k(c_k - d_k)^+ \quad (34)$$

subject to

$$x_{\ell k} + x_{k\ell} + y_{\ell k} + y_{k\ell} \geq w_k^q + w_\ell^q - 1 \quad \forall k, \ell, q \text{ where } k < \ell \quad (35)$$

$$\sum_{q=1}^m w_k^q = 1 \quad \forall k \quad (36)$$

$$t_\ell \geq c_k + (x_{k\ell} - 1)M \quad \forall k, \ell \text{ where } k \neq \ell \quad (37)$$

$$b_\ell \geq b_k + h_k + (y_{k\ell} - 1)M \quad \forall k, \ell \text{ where } k \neq \ell \quad (38)$$

$$t_k \geq a_k \quad \forall k \quad (39)$$

$$c_k \geq t_k + \sum_{q=1}^m p_k^q w_k^q \quad \forall k \quad (40)$$

$$b_k \leq \sum_{q=1}^m B_q w_k^q - h_k + 1 \quad \forall k \quad (41)$$

$$b_k \geq 1 \quad \forall k \quad (42)$$

$$x_{\ell k} \in \{0, 1\}, \quad y_{\ell k} \in \{0, 1\} \quad \forall k, \ell \text{ where } k \neq \ell \quad (43)$$

$$w_k^i \in \{0, 1\} \quad \forall k, i \quad (44)$$

Constraint (35) says that vessel rectangles cannot overlap if corresponding vessels are assigned to the same berth. Constraint (36) ensures that each vessel should be assigned to a berth. In Constraint (40), the expression $\sum_{q=1}^m p_k^q w_k^q$ defines the duration of stay for vessel k depending on berth assignment. Similarly, in Constraint (41), the expression $\sum_{q=1}^m B_q w_k^q$ gives the length of the berth that vessel k is assigned. With the help of these expressions, Constraints (37) - (42) work in the same way as Constraints (12) - (17) do in the RPF of BAP.

3.3 Lower Bound Analysis

In this section we introduce a polynomially-computable lower bound for the objective function value of the dynamic MBAP using a similar argument introduced for the BAP lower bound.

For any instance of the MBAP, we construct a weighted bipartite matching problem P_M on a bipartite graph $G = (\mathcal{N}_1 \cup \mathcal{N}_2, \mathcal{E})$. For each vessel k , we create $p_k^* \times h_k$

nodes in \mathcal{N}_1 . We use a two dimensional index (i, j) for each of these nodes defined for vessel k , where i refers to the spatial position and j the time position. Hence, each node in \mathcal{N}_1 is represented by a triplet (k, i, j) . Similarly, we define a unique node in \mathcal{N}_2 for each (berth section, time period) pair of each berth. Hence, each node in \mathcal{N}_2 is represented by a triplet (q, b, t) where q, b , and t are the corresponding berth, berth section, and time period respectively. For each node (k, i, j) in \mathcal{N}_1 , we define parameters r_{ij}^k , d_{ij}^k , α_{ij}^k , and β_{ij}^k as follows.

$$\begin{aligned} r_{ij}^k &= a_k + j - 1 \\ d_{ij}^k &= d_k - p_k^* + j \\ \alpha_{ij}^k &= \frac{1}{p_k^* h_k} \\ \beta_{ij}^k &= \frac{f_k}{p_k^* h_k} \end{aligned}$$

We construct the edge set \mathcal{E} by creating an edge between node (k, i, j) in \mathcal{N}_1 and node (q, b, t) in \mathcal{N}_2 if $r_{ij}^k \leq t$ and $i \leq b \leq B_q - h_k + i$. For each edge $\{(k, i, j), (q, b, t)\}$ created, we assign a weight $c_{\{(k, i, j), (q, b, t)\}}$ defined as follows.

$$c_{\{(k, i, j), (q, b, t)\}} = \alpha_{ij}^k (t + 1 + p_k^q - p_k^*) + \beta_{ij}^k (t + 1 + p_k^q - p_k^* - d_{ij}^k)^+$$

We know that the minimum weight bipartite matching problem can be solved in polynomial time. Thus, P_M can be solved in polynomial time.

Theorem 5 $LB_{P_M} = \lceil C_{P_M}^* + \Phi - \Psi \rceil$ is a lower bound for MBAP where $C_{P_M}^*$ is the optimum objective function value of P_M constructed for MBAP, $\Phi = \sum_{k=1}^n (p_k^* - 1)/2$, and $\Psi = \sum_{k=1}^n a_k$.

Proof. Let q_{ij}^k, b_{ij}^k , and t_{ij}^k denote respectively the q, b , and t values of node (q, b, t) that corresponding (k, i, j) node is matched in P_M . First, it is easy to see that any feasible solution of MBAP yields a feasible solution for the corresponding P_M . In

such a solution, for each node (k, i, j) defined for vessel k , $q_{ij}^k = q_k$, $b_{ij}^k = b_k + i - 1$, and $t_{ij}^k = t_k + j - 1$ hold. Furthermore $c_k = t_k + p_k^{q_k}$ is true for each vessel. Then the corresponding P_M solution has the objective function value:

$$C_{P_M} = \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^*} (\alpha_{ij}^k (t_{ij}^k + 1 + p_k^{q_k} - p_k^*) + \beta_{ij}^k (t_{ij}^k + 1 + p_k^{q_k} - p_k^* - d_{ij}^k)^+) \quad (45)$$

Using a little algebra, the first part of the objective function can be rewritten as

$$\begin{aligned} & \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^*} \alpha_{ij}^k (t_{ij}^k + 1 + p_k^{q_k} - p_k^*) = \\ &= \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^*} \frac{1}{p_k^* h_k} (t_k + j - 1 + 1 + p_k^{q_k} - p_k^*) = \\ &= \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^*} \frac{1}{p_k^* h_k} (c_k - (p_k^* - j)) = \sum_{k=1}^n (c_k - \sum_{j=1}^{p_k^*} \frac{(p_k^* - j)}{p_k^*}) = \\ &= \sum_{k=1}^n (c_k - \sum_{j=1}^{p_k^* - 1} \frac{(j)}{p_k^*}) = \sum_{k=1}^n (c_k - \frac{(p_k^* - 1)p_k^*}{2p_k^*}) = \sum_{k=1}^n c_k - \sum_{k=1}^n \frac{(p_k^* - 1)}{2} \end{aligned}$$

Similarly, the second part of the objective function can be rearranged as

$$\begin{aligned} & \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^*} \beta_{ij}^k (t_{ij}^k + 1 + p_k^{q_k} - p_k^* - d_{ij}^k)^+ = \\ &= \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^*} \frac{f_k}{p_k^* h_k} (t_k + j - 1 + 1 + p_k^{q_k} - p_k^* - d_k + p_k^* - j)^+ = \\ &= \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^*} \frac{f_k}{p_k^* h_k} (t_k + p_k^{q_k} - d_k)^+ = \sum_{k=1}^n f_k (c_k - d_k)^+ \end{aligned}$$

which yields

$$C_{P_M} = \sum_{k=1}^n c_k - \sum_{k=1}^n \frac{(p_k^* - 1)}{2} + \sum_{k=1}^n f_k (c_k - d_k)^+ \quad (46)$$

because $C_{P_M^*} \leq C_{P_M}$,

$$C_{P_M^*} + \sum_{k=1}^n \frac{p_k^* - 1}{2} - \sum_{k=1}^n a_k \leq \sum_{k=1}^n (c_k - a_k) + \sum_{k=1}^n f_k (c_k - d_k)^+ \quad (47)$$

With parameters being all integer, the right hand side of expression (47), which is the objective function of MBAP, is integer. Hence, $LB_{P_M} = \lceil C_{P_M}^* + \Phi - \Psi \rceil$ is a lower bound for MBAP. \square

Theorem 6 $LB_{P_M} \geq LB_{LP}$ where LB_{LP} denotes the objective function value of the optimal solution to the LP relaxation of the RPF.

Proof. In a feasible solution to the LP relaxation, it is easy to see that $w_k^{q_k} = 1$ for $q_k = \{q | p_k^* = p_k^q\}$ can be chosen and berthing times and positions can be set in such a way that any vessel rectangles may overlap one another. Hence, in any optimal solution, $\sum_{q=1}^m p_k^q w_k^q = p_k^*$ and $c_k = a_k + p_k^*$ hold $\forall k \in N$. Because we assume $d_k \geq a_k + p_k^*$,

$$LB_{LP} = \sum_{k=1}^n (c_k - a_k) = \sum_{k=1}^n p_k^* \quad (48)$$

In any feasible P_M solution, $t_{ij}^k \geq r_{ij}^k$ and $r_{ij}^k = a_k + j - 1$ for each (k, i, j) triplet. Then,

$$C_P^* \geq \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^*} \alpha_{ij}^k (t_{ij}^k + 1) \geq \sum_{k=1}^n \sum_{j=1}^{p_k^*} \frac{(a_k + j)}{p_k^*} \quad (49)$$

which, by Theorem 5, implies,

$$LB_{P_M} \geq \left\lceil \sum_{k=1}^n \sum_{j=1}^{p_k^*} \frac{a_k + j}{p_k^*} + \sum_{k=1}^n \frac{p_k^* - 1}{2} - \sum_{k=1}^n a_k \right\rceil. \quad (50)$$

Rearranging terms then yields the result:

$$LB_{P_M} \geq \left\lceil \sum_{k=1}^n \frac{p_k^* + 1 + p_k^* - 1}{2} \right\rceil = \sum_{k=1}^n p_k^* = LB_{LP} \quad (51)$$

□

3.4 *Solution Method*

In this section, we present how the nested tabu search algorithm designed for the BAP can be modified to solve the MBAP efficiently. The presence of multiple berths changes the way we encode a feasible solution and also changes the neighborhood structure used in TS_1 designed for the BAP.

3.4.1 Encoding and Decoding a Solution

We use a vector of size n , \mathcal{B} , and a set of lists containing a total of n elements to encode a feasible solution. Entries of \mathcal{B} are the berth positions of vessels. $\mathcal{L} = \{\mathcal{L}_1, \dots, \mathcal{L}_m\}$ is a collection of vessel priority lists defined for each berth. The n_q entries of the ordered list \mathcal{L}_q are vessel indices, and the order determines berthing priority for berth q . Here, n_q denotes the number of vessels assigned to berth q , and $\sum_{q=1}^m n_q = n$. Any \mathcal{B} and \mathcal{L} pair defines a unique feasible berth schedule x as follows: for each berth q , beginning with the first vessel in \mathcal{L}_q , berth each vessel k at the berthing position given in \mathcal{B} at the later of a_k or the earliest time that the berthing time-space rectangle to be occupied by k is not occupied by any earlier berthed vessel. With appropriate data structures, this decoding operates in $O(n)$ time. To encode a berth schedule x , the vessel berth positions uniquely define \mathcal{B} , and we use the primal list, $\mathcal{L}_q^P(x)$, constructed for each berth q as described in the previous chapter. With such an encoding note that the schedule for each berth can be decoded independent of each other.

3.4.2 Initial Solution

An initial berth schedule, x^0 , can be generated by using a single aggregate vessel list \mathcal{L}^A and placing each vessel in the list one by one on the berth which results in the

minimum completion time for the vessel considered; note that the berthing position for each vessel that minimizes the completion time on a given berth mimics the two-dimensional first-fit rule used in the prior chapter for assigning berth positions in the case of a single berth. The minimum cost solution provided by the lists \mathcal{L}^{FCFS} , \mathcal{L}^{EDD} , and \mathcal{L}^{mEDD} is chosen. This greedy procedure provides a feasible packed berth schedule which mimics decision rules-of-thumb that a terminal operator might use, and can be used as an initial solution.

3.4.3 Fundamentals of the Nested Tabu Search

For a given \mathcal{L} and \mathcal{B} , and their corresponding berth schedule x , a neighboring solution can be defined by some change to \mathcal{B} , or to \mathcal{L} , or both. The nested approach described in the previous chapter is utilized. It explores changes to \mathcal{B} given a fixed $\mathcal{L}^P(x) = \{\mathcal{L}_1^P(x), \dots, \mathcal{L}_m^P(x)\}$ as an inner search TS_2 , while an outer search TS_1 explores changes to \mathcal{L} .

At each iteration of TS_1 , we modify the current \mathcal{L} using swap and insertion moves within a randomized neighborhood. We define the (p, r, q) randomized neighborhood of list sets $\mathcal{N}_{\mathcal{L}}(p, r, q, \mathcal{L})$ as follows. We first select q vessels randomly. Then, for each vessel k selected, we generate a candidate vessel set \mathcal{C}_k which includes all vessels that are berthed at time periods between $t_k - r$ and $c_k + r$ in the corresponding berth schedule x , which is generated by the FF decoding each berth vessel priority list $\mathcal{L}_q \in \mathcal{L}$. We consider a random subset $\hat{\mathcal{C}}_k \subseteq \mathcal{C}_k$ where $|\hat{\mathcal{C}}_k| = r$. For each $\ell \in \hat{\mathcal{C}}_k$, if $q_k = q_\ell$ in berth schedule x , we create a move by swapping the positions of k and ℓ in \mathcal{L}_{q_k} . Otherwise, if $q_k \neq q_\ell$, we create both a swap move which places vessel k in \mathcal{L}_{q_ℓ} at the position of vessel ℓ and places vessel ℓ in \mathcal{L}_{q_k} at the position of vessel k , and also an insertion move which removes vessel k from \mathcal{L}_{q_k} , and places it after vessel ℓ in \mathcal{L}_{q_ℓ} .

Entries of \mathcal{B} that correspond to vessels in each list \mathcal{L}_q which is modified by a

candidate move are updated using the FF rule. The resulting \mathcal{B} and \mathcal{L} correspond to berth schedule x with marks $M_q(x)$ and primal lists $\mathcal{L}_q^P(x)$ for each berth q . Each non-tabu candidate move is considered in sequence, where a tabu move is one whose mark vector $M(x) = \{M_1(x), \dots, M_m(x)\}$ is currently in the tabu list. If the mark $M(x)$ has not yet been considered during the current TS_1 iteration, we initiate the inner search TS_2 with x as the initial solution.

TS_2 is only run for berth q if the upper layer move changes \mathcal{L}_q . The details of the inner search is exactly the same as described for the BAP in the previous chapter. At the completion of the TS_2 runs for the berths specified, a best (lowest objective function value) berth position vector and corresponding best solution is identified for the current candidate move of TS_1 . After all candidate TS_1 moves have been evaluated for the current iteration, the best move is selected as the new current solution x , with its corresponding \mathcal{B} and \mathcal{L} . The mark vector $M(x)$ of this solution is placed into the TS_1 tabu list for a random number of iterations Θ_1 , where Θ_1 is discrete uniform on $[\Theta_1^{min}, \Theta_1^{max}]$. Furthermore, if the move selected changes the berth assignment of some vessels, assigning these vessels back to their previous berths is defined tabu for Υ_1 iterations where Υ_1 is discrete uniform on $[\Upsilon_1^{min}, \Upsilon_1^{max}]$.

Similar to the BAP, a multi-start variant of the nested tabu search algorithm can be utilized for the MBAP with initial solutions found as described in the previous chapter.

3.5 Computational Experiments

Computational experiments on randomly generated test instances were conducted to evaluate the performance of the nested tabu search algorithm designed for the MBAP. Nine sets of test instances were used, each containing 5 different randomly generated instances. The first four of these sets include relatively small problem instances for a terminal with $m = 2$, $B_1 = 9$, and $B_2 = 6$. They have 10, 12, 14, and

16 vessels respectively. For these instance sets, we assume $a_k \sim U^D[1, 10]$ for vessel k , where $\sim U^D[a, b]$ represents a randomly generated integer between a and b . The remaining five sets represent larger instances with respectively 30, 35, 40, 45, and 50 vessels visiting a terminal with $m = 4$, $B_1 = 20$, $B_2 = 15$, $B_3 = 10$, and $B_4 = 6$. $a_k \sim U^D[1, 15]$ for vessels in these instances. Note that these terminals get busier from set 1 to set 4, and set 5 to set 9. For all instances, $h_k \sim U^D[2, 6]$, and each instance has at least one vessel of each size. For vessel k with length h_k , we generate h_k integers $p_k^1, \dots, p_k^{h_k}$ randomly where $p_k^i \sim U^D[1, 4]$ for $h_k = 2$, $p_k^i \sim U^D[1, 5]$ for $h_k \in \{3, 4\}$, and $p_k^i \sim U^D[1, 6]$ for $h_k \in \{5, 6\}$. We set $\bar{p}_k = \max_i \{p_k^i\}$. Then, we select one of the berths randomly for each vessel and declare it as the vessel's favorite berth q_k^* where $p_k^{q_k^*} = \bar{p}_k$. For any other berth $q \neq q_k^*$ we set $p_k^q = \lfloor L\bar{p}_k \rfloor$ where $L \sim U^D[1.25, 1.75]$ for each vessel k . Finally, for all vessels, $d_k = a_k + Kp_k^{q_k^*}$ where $K \sim U^D[1, 3]$, and $f_k \sim U^D[3, 5]$.

Table 6: Statistics for the small MBAP instances

Set	m	B_1	B_2	n	RU_{max}	RU_{avg}
1	2	9	6	10	1.950	1.079
2	2	9	6	12	2.392	1.294
3	2	9	6	14	2.493	1.410
4	2	9	6	16	2.492	1.479

Similar to the computational experiments in Chapter 2, we use the term *required utilization* (RU) to measure the congestion at the terminal for a given problem instance. RU_t^q denotes the ratio of berth q sections required to berth q sections available at time t to moor arriving vessels having berth q as their favorite berth exactly at their arrival times. RU_{max} and RU_{avg} are computed over all berths and reported in Table 6 and Table 7 along with the size of the instances for small and large instance sets respectively.

Instances are numbered in the tables using two digit numbers xy , where x corresponds to the set and y indicates the individual instance number.

Table 8 summarizes the tabu search parameters used for both single-start (TS)

Table 7: Statistics for the large MBAP instances

Set	m	B_1	B_2	B_3	B_4	n	RU_{max}	RU_{avg}
5	4	20	15	10	6	30	1.541	0.748
6	4	20	15	10	6	35	1.671	0.860
7	4	20	15	10	6	40	1.949	1.011
8	4	20	15	10	6	45	2.031	1.080
9	4	20	15	10	6	50	2.592	1.318

and multi-start (mTS) versions.

Table 8: The MBAP tabu search parameters used in the computational experiments

Parameter	TS	MTS
R	-	$\max\{10, n/2\}$
p	5	5
r	10	5
q	5	5
t_1^2	$50n$	$5n$
θ_1^{min}	$n/2$	$n/4$
θ_1^{max}	n	$n/2$
Υ_1^{min}	$n/2$	$n/4$
Υ_1^{max}	n	$n/2$
s	$\max\{2, V /3\}$	$\max\{2, V /3\}$
t_2^2	$n/2$	$n/2$
θ_2^{min}	2	2
θ_2^{max}	5	5

The cost of the initial solution is provided in Table 9 and 10 for small and large instances respectively. Table 9 and 10 also present the two polynomially computable lower bounds LB_{LP} and LB_{P_M} . Theorem 6 proves that $LB_{P_M} \geq LB_{LP}$, and empirical results show that LB_{P_M} improves LB_{LP} by 20%, 45%, 58%, and 68% for small instances 1y, 2y, 3y, and 4y) respectively. The improvement is observed as 11%, 15%, 20%, 31%, and 55% for large instances 5y, 6y, 7y, 8y, and 9y respectively.

The mixed integer program presented above was applied on small instances using CPLEX on an Intel 2.4 GHz Pentium processor running Linux with 2GB memory. We let CPLEX run for at most 2 hours, and it was able to solve all 5 instances in sets 1 and 2, and 3 out of 5 instances in set 3 optimally. None of the set 4 instances could be solved optimally within two hours. The lower bound (LB_{MIP}) and the best

Table 9: Test results for small MBAP instances

Instance	IS	LB_{LP}	LB_{PM}	LB_{MIP}	UB_{MIP}	TS	mTS	Imp. (%)
10	81	43	48	59	59	59	59	100
11	81	47	55	70	70	70	70	100
12	108	49	59	85	85	85	85	100
13	79	40	55	71	71	71	71	100
14	107	45	50	63	63	63	63	100
20	191	60	89	127	127	127	127	100
21	160	56	90	141	141	141	141	100
22	97	50	64	83	83	83	83	100
23	132	51	70	100	100	100	100	100
24	124	52	77	90	90	90	90	100
30	272	67	123	166	230	230	230	40
31	289	65	113	168	168	168	168	100
32	104	57	74	91	91	91	91	100
33	200	64	103	146	150	150	150	93
34	157	64	90	117	117	117	117	100
40	334	72	121	134	163	154	154	90
41	339	74	128	192	217	217	217	83
42	150	67	89	102	114	114	114	75
43	165	67	105	115	151	145	145	40
44	427	72	149	195	321	318	318	47

integer solution found (UB_{MIP}) are provided in Table 9.

Best solutions found by TS and mTS are presented in Table 9 and 10. Note that for small instances presented in Table 9, TS and mTS provide berth schedules with the same cost. Furthermore, they found the optimal solutions for instances where the optimal solution was obtained by MIP. For the remaining 7 instances where optimality was not proved, when compared to MIP solution, our algorithms found an equivalent solution in 4 cases, and came up with better solutions for the rest. The last column in Table 9 presents the improvement in optimality gap obtained by the tabu search algorithms over the initial solution. Here, LB_{MIP} is used to compute gaps. For cases where the optimality is proved, the gap is reported as 100%, which means the gap is completely removed. The average gap improvement realized over the initial solution by tabu search is 88% over all small instances, and 67% for the ones where the optimality could not be proved.

Table 10 presents the results for large instances. TS and mTS found equivalent solutions for all set 5 and 3 of set 6 instances. For the rest, mTS found better solutions

Table 10: Test results for large MBAP instances

Instance	IS	LB_{LP}	LB_{P_M}	TS	mTS	TS Imp (%)	mTS Imp (%)
50	165	136	147	154	154	61	61
51	212	144	158	178	178	63	63
52	169	132	143	156	156	50	50
53	208	146	163	180	180	62	62
54	229	141	165	191	191	59	59
60	269	164	186	225	225	53	53
61	219	158	171	192	192	56	56
62	396	180	228	281	280	68	69
63	368	174	197	250	248	69	70
64	261	160	182	213	213	61	61
70	285	170	186	215	213	71	73
71	394	192	277	328	321	56	62
72	457	184	221	275	273	77	78
73	361	188	216	258	255	71	73
74	227	164	177	200	199	54	56
80	552	223	316	410	399	60	65
81	388	195	228	281	274	67	71
82	488	205	246	303	296	76	79
83	483	206	265	310	305	79	82
84	697	227	329	444	437	69	71
90	656	235	328	415	413	73	74
91	1775	268	575	995	993	65	65
92	835	237	324	420	419	81	81
93	706	230	338	438	420	73	78
94	674	228	305	385	381	78	79

compared to TS . Since these instances are too large to run on CPLEX, LB_{P_M} is used to calculate gaps. The average gap improvement is 66% for TS , and 68% for mTS . Furthermore, the amount of improvement seems consistently to increase as we go from set 5 instances to set 9 instances for both tabu search versions. They are computed as 59%, 61%, 66%, 70%, and 74% on average for TS , and 59%, 62%, 68%, 74%, and 76% on average for mTS for instances in sets 5, 6, 7, 8, and 9 respectively.

Table 11 presents computational times in seconds observed during tests. With the parameters used, it appears that for large instances the single-start tabu search TS finds slightly worse solutions than the multiple-start mTS , but finds these solutions slightly faster. For small instances, optimal solutions are generated by both methods under 10 seconds where MIP may spend hours. The largest instance takes less than 12 minutes to solve by the tabu search algorithms proposed.

Table 11: Average computational times for test MBAP instances

Set	MIP	TS	mTS
1x	5	3	4
2x	50	5	7
3x	4328	7	8
4x	7858	10	12
5x	-	79	106
6x	-	142	189
7x	-	240	318
8x	-	235	498
9x	-	485	693

CHAPTER IV

THE QUAY CRANE SCHEDULING PROBLEM

4.1 Introduction

Quay cranes are very important resources at container terminals. They are used to load containers onto and discharge containers from vessels at the quayside of terminals. Quay cranes along the same berth are mounted and operate on a common set of rails. This prevents quay cranes from passing each other at any time. In this chapter, we focus on the problem of scheduling quay cranes at container terminals for a given berth schedule, namely the Quay Crane Scheduling Problem (QCSP).

The problem of scheduling quay cranes has been studied before in many different settings. Reference [16] considers both the static problem of assigning cranes to a set of vessels present at port at the time of decision making to minimize total weighted vessel completion time, as well as a dynamic extension in which vessels arrive over time. Each vessel consists of a number of holds where each hold can be processed by only one crane at a time; this model captures the distribution of unloading/loading work along the length of each vessel and properly accounts for the physical space occupied by cranes. However, the length of the berth is not modeled, and thus it is assumed that the berth is long enough to handle all vessels simultaneously. Furthermore, the paper also assumes that cranes can pass each other. The paper develops a mixed integer program for the static problem, and provides scheduling principles that are then used to produce heuristic solutions to both the static and dynamic problems. Researchers also study the static quay crane scheduling problem with no berth length limitation in [59]. The developed model uses an objective of minimizing weighted tardiness, and is solved via a branch and bound method. Instead of

considering individual vessel data, an aggregate study using probability distributions describing vessel arrivals and workloads is performed in [17] to analyze the effect of quay crane allocation strategies on long-run maximum terminal throughput and average vessel delay.

A branch and bound algorithm and a greedy randomized adaptive search procedure (GRASP) is proposed in [33] to solve a quay crane scheduling and load sequencing problem restricted to a single vessel. Compared to the approach in [59], the proposed model captures more details like individual locations of containers to be moved onto or off the vessel and properly restricts cranes so that they may not pass each other during operation, which we denote as non-crossing constraints. The objective used is to minimize a weighted sum of the makespan of the vessel and the total completion time of all quay cranes. The computational complexity of the problem is not discussed.

Quay crane scheduling with non-crossing constraints is also the focus of [43] and [75] (see also [41], [42] and [74]). In [43], the problem is modeled as a maximum weight bipartite graph matching problem. Jobs represent container unloading/loading work, and arise at specific locations along the quayside. The collection of jobs may represent work for a single ship, or multiple ships. Vertices define jobs and cranes, and the container throughput realized when a crane is assigned to a job is assigned to be the weight of each connecting edge. Since spatial constraints that restrict cranes passing each other and prevent job processing due to the physical size of cranes complicate the matching problem, the paper develops a dynamic programming algorithm to solve problems with such constraints. Furthermore, the model is enhanced to consider more complex spatial constraints that they call job separation constraints which limits processing of certain jobs simultaneously. The enhanced model is an *NP*-hard optimization problem, and the authors propose solution via a probabilistic tabu search and a squeaky wheel optimization with local search. Since real-world

terminals focus on completing all vessel unloading/loading jobs as earlier as possible, this model is enhanced in [75] to consider the real-world practice of attempting to minimize the completion time of the latest job.

There are several other recent studies on quay crane scheduling also applicable to single vessel case. Among these, a heuristic is proposed in [53] to solve the scheduling problem of identical quay cranes moving on a linear rail. The approach attempts to minimize a given ship's stay time in port by finding an appropriate work schedule for each quay crane. Only one crane can operate on individual ship bay at a single time. The problem is formulated using an integer programming model, and the model is decomposed by partitioning the ship into non-overlapping zones. The authors of [48] formulate the quay crane problem as a VRP with side constraints including precedence relationships between vertices. Their formulation strengthens the model given in [33]. The objective is to minimize the weighted sum of the completion time of a single vessel and the idle times of cranes which originate from interferences between cranes since cranes roll on the same rails and a minimum safety distance must be maintained between them. In [37], the quay scheduling with non-interference constraints is studied. The study is motivated by the problems studied in [33]. The objective is the minimization of the makespan for a single container vessel, and the paper develops a genetic algorithm approach that obtains near optimal solutions for the proposed MIP model.

Quay crane scheduling for a given berth schedule is first mentioned in [44]. The objective of the problem considered in that reference is to minimize the maximum relative tardiness of vessel departures. The authors propose a heuristic decomposition into a vessel level model and a berth level model. The vessel level model provides the optimal processing time for any given number of quay cranes assigned to each individual vessel. The berth level model considers the entire set of vessels. Quay cranes are assigned among the vessels using the results from the given berth allocation

and from the vessel level model. Since a quay crane assigned to a vessel cannot be re-deployed to another vessel until all cranes assigned to the first vessel have completed their work, this model reduces to a simplified problem that we call dedicated crane allocation in this thesis. In practice, better solutions can be found that do not impose this restriction.

In this chapter, we focus on a quay crane scheduling problem (QCSP) defined for a given berth schedule. A berth schedule provides berthing positions and berthing priorities for a given set of vessels. Therefore, the crane scheduling problem variant we study is defined for multiple vessels that are planned to dock at a berth over some finite planning horizon. The primary contributions of this chapter include:

- The classification and analysis of different crane scheduling methods;
- The development of an effective tabu search algorithm designed for the most realistic crane scheduling model that allows crane roaming and shifting; and
- The introduction of the crane assignment problem and a polynomial time method for its solution.

The QCSP variants that we consider in this thesis use the output of the berth allocation problem (BAP) as input. The result of BAP can be expressed on a time-space diagram similar to that in Figure 7. In practice, many terminal operators use a hierarchical approach for scheduling berths and quay cranes. They first determine a good berth schedule using estimates on total processing (berthing) time for each vessel by solving the BAP. The resultant berth schedule specifies the berthing position and an estimated berthing time for vessels considered in the planning horizon. Once a reasonable berth schedule is determined, port operators attempt to allocate available quay cranes to vessels that are planned to berth simultaneously. Once cranes are assigned, the actual berthing and completion times of vessels are determined with more precision.

Because we consider a variant of the problem with multiple vessels, the number of containers discharged from and loaded onto vessels is expected to be large. Since a limited number of quay cranes can simultaneously work a ship due to the size of the cranes and guidelines for safe crane separations, we develop a reasonable model of the distribution of container unload/load workload along the length of each vessel. We assume that each vessel is divided along its length into holds, where each hold can be visualized as a number of container rows. We assume that the aggregate workload, measured in units of processing time, of each hold can be reasonably estimated; for example, this might be computed as the product of the number of containers to be handled in the hold and the average processing time per container. The problem setting we use is very similar to those used in [16] and [59]. In these studies, vessels are also partitioned into holds to model the work distribution into account.

Defining holds as 3 or 4 container rows seems to be reasonable, given average crane widths and guidelines for safe crane separation during operation. Note that such a hold definition also allows us to measure the length of each vessel in terms of the number of holds they have, and the berth length in terms of the total number of holds it can accommodate. Of course, such a discretization is likely to be imperfect, and may result in wasted berthing space and/or less-than-optimal crane utilization. Furthermore, we recognize that we assume that cranes simultaneously processing adjacent holds on single vessels may need to be scheduled carefully so that no safety distances are compromised while they work on nearby container within their assigned holds. We also assume that when work starts on a hold it continues until completion without interruption, and a vessel can leave the port only after every hold on that vessel is processed.

It is expected that the container processing time of a vessel decreases when more quay cranes are assigned to it as long as there is no interference between the assigned quay cranes and other relevant resources are allocated accordingly.

In our model, we discretize time and berth space. The number of containers in a hold structure as we define it can be close to 1000 for modern mega-ships. Hence, the processing time required for such a hold can be as large as a full day. Therefore, we choose to discretize time using relatively larger time periods of 3 or 4 hours. Berth is discretized into sections so that the length of each berth section is equal to the length of a hold. We have Q identical quay cranes which operate on a single set of rails. Vessel set is denoted by \mathcal{V} where $n = |\mathcal{V}|$. For each vessel $k \in \mathcal{V}$, we define:

h_k : number of holds of vessel k ,

p_k^i : processing time of hold i of vessel k ,

p_k^{max} : maximum hold processing time for vessel k ($p_k^{max} = \max_i \{p_k^i\}$),

a_k : arrival time of vessel k ,

d_k : due time of vessel k (where $d_k \geq a_k + p_k^{max}$),

f_k : lateness penalty for vessel k ,

b_k : berthing position of vessel k ,

t_k : berthing time of vessel k ,

c_k : the earliest time that vessel k can depart.

Additionally, for each vessel k , we define a set, $\Omega(k)$, of vessels such that vessel ℓ is in $\Omega(k)$ if vessel ℓ is planned to dock after vessel k in the BAP and requires at least one of the berth sections occupied by vessel k .

Given a known vector of arriving vessel information and berth allocation plan represented together by $\{h_k, p_k^1, \dots, p_k^{h_k}, a_k, d_k, f_k, b_k\}$ for each vessel k , the optimization problem is then to find actual berthing time period t_k , and a crane allocation plan which specifies the duration of stay for vessel k and hence determines c_k . We consider the same objective function used in previous chapters. We minimize $\sum_{k \in N} (c_k - a_k) + \sum_{k \in N} f_k (c_k - d_k)^+$ which is the sum (or average) of the dwell times and the total penalty accrued due to tardy vessels.

4.2 *Problem Types and Models*

In this section we describe two problem types that arise depending on the method used for assigning cranes to vessels. In the first method, the terminal operator assigns a number of cranes to each vessel and these cranes can only work on that specific vessel until it leaves port. This method may be chosen for its simplicity in practice or there may be specific restrictions in the agreement between the terminal operator and the ocean carrier. We refer to this approach as dedicated crane assignment and call the resulting problem the dedicated quay crane scheduling problem. In the alternative approach, cranes assigned to a vessel may leave that vessel if needed and can be allocated to other vessels simultaneously docked. Hence, cranes can roam on the berth between vessels during operation. We refer to this method as roaming crane assignment and call the resulting problem the roaming quay crane scheduling problem.

Before presenting models for dedicated and roaming crane scheduling, we also discuss the important issue of crane blocking, and how crane shifting can be used to overcome problems related to crane blocking. The decision of using crane shifting or not creates two variants for both dedicated and roaming crane scheduling.

4.2.1 **Crane Blocking and Crane Shifting**

As noted, quay cranes operate on rails mounted on the berth. This configuration prevents cranes from passing each other during operation. This restriction can create a situation that we call *crane blocking* in this thesis. We say a crane is blocked if we cannot find any job (hold) that the crane can work within the region it can presently reach, which may be restricted by one or two adjacent cranes that are working. If in a time period, the adjacent crane or cranes are working on holds such that there exists no other hold requiring work in between, then this crane is blocked for that time period.

An assumption that we already mentioned is that when work starts on a hold it continues until completion without interruption. However, this assumption does not prevent a change of cranes during the processing of a hold, as long as the hold processing is not interrupted. We call this change of cranes during the processing of a hold crane shifting. Quay cranes move relatively slowly, with speeds around 40-60 m/min. Given our assumptions, the time required to move a quay crane from the position of one hold to the position of an adjacent hold should be on the order of a few minutes. This is a very small amount of time, given that time is discretized into 3-4 buckets. Hence, relatively speaking, we can say that quay cranes move quickly on the berth, compared to the time it takes to process one hold.

In practice, terminal operators may assign one crane to each hold and avoid crane shifting for operational simplicity. This decision, however, increases crane blocking and may lead to under-utilization of these valuable resources. With crane shifting performed at the beginning of each time period, crane blocking can always be prevented. In order to benefit from crane shifting, terminal operators should know when and which cranes to shift. However, getting this information is tricky unless the problem is modeled and solved appropriately. In the following sections, we introduce models with and without crane shifting for both dedicated and roaming crane scheduling. In each model, we omit the time required to move cranes on the berth since, as mentioned above, this time will be very small compared to the length of a time period considered. Models with no crane shifting provide solutions where the hold assignment of each individual crane is determined specifically, namely which crane works on which hold in what time periods is explicitly specified. On the other hand, models that allow crane shifting only determine the time periods during which each hold is processed. The only resource constraint in such models is a limit on the total number of cranes used in any time period. In order to generate a complete solution and an operational plan for cranes in this case, individual crane assignments should

be specified. For this reason, we define the Crane Assignment Problem (CAP), which will be described later, to determine work plans (routes) for each quay crane with the objective of minimizing total distance traveled by cranes during shifting moves along the berth over the planning horizon.

4.2.2 Dedicated Quay Crane Scheduling

In dedicated quay crane scheduling, the terminal operator first decides on the number of cranes to be allocated to each vessel. Then, he assigns those cranes to jobs (holds) within each vessel. Since the cranes assigned to each vessel can only operate on that specific vessel while it remains berthed, crane blocking may occur. However, the terminal operator can also decide whether or not to allow crane shifting within each vessel. Hence, two problems can be defined; one with crane blocking and no intra-ship crane shifting, and another with crane shifting within each vessel.

Both of these problems can be decomposed into subproblems, and can be solved to optimality in two stages. In the first stage, we define a vessel level problem for each vessel considered in the planning horizon. The vessel level problem is to minimize the processing time for a vessel for a given number of assigned quay cranes. By solving the vessel level problem for all possible numbers of cranes that can be assigned to each ship, we generate data elements $p_k(q)$, the total processing time required for a ship k given that q cranes are assigned. In the second stage (master problem), we determine the most appropriate number of quay cranes and specifically which cranes to assign to each vessel to minimize the objective value.

While we need to solve many vessel level problems, the vessel level problems are each of small sizes and easy to solve in both crane blocking and crane shifting (within vessel) case. Because of the decomposition, the master problem also is relatively small.

4.2.2.1 Vessel Level Problem Formulation with Crane Blocking

The vessel level model with crane blocking and no intra-vessel crane shifting finds the minimum processing time of a vessel for a given number of assigned quay cranes. As a reminder, we assume that jobs cannot be interrupted, and therefore a crane assigned to a hold of the vessel must process that hold until all container operations are completed. Note that this problem allows a crane to move from one hold of a ship to another, but only after all work at the initially assigned hold is completed. In the problem formulation, we keep track of the locations of individual cranes. Hence, we define:

L_m^t : location of crane m at time period t .

A vessel level model is defined for a particular vessel. For conciseness of exposition, we drop the subscript of the vessel in the presentation and hence define:

H : number of holds,

p^i : processing time of hold i ,

Furthermore, we use the following binary variable set:

$$z_{mi}^t = \begin{cases} 1 & \text{if crane } m \text{ starts working on hold } i \text{ at time } t, \\ 0 & \text{otherwise;} \end{cases}$$

Then, the problem can be formulated as a mixed integer program as follows where T represents the number of time periods in the planning horizon which should be set large enough to model the problem feasibly but no larger than $\sum p^i$.

$$\text{Minimize } p(Q) \quad (52)$$

subject to

$$\sum_{m=1}^Q \sum_{t=1}^T z_{mi}^t = 1 \quad \forall i \quad (53)$$

$$p(Q) \geq tz_{mi}^t + p^i - 1 \quad \forall m, i, t \quad (54)$$

$$L_m^t \leq L_{m+1}^t - 1 \quad \forall m, t \quad (55)$$

$$L_m^{t+\bar{t}} \geq (z_{mi}^t - 1)H + i \quad \forall m, i, t, \bar{t} \in \{0, \dots, p^i - 1\} \quad (56)$$

$$L_m^{t+\bar{t}} \leq (1 - z_{mi}^t)H + i \quad \forall m, i, t, \bar{t} \in \{0, \dots, p^i - 1\} \quad (57)$$

$$1 \leq L_m^t \leq H \quad \forall m, t \quad (58)$$

$$z_{mi}^t \in \{0, 1\} \quad \forall m, i, t \quad (59)$$

Constraint (53) ensures that a crane is assigned to each hold. $p(Q)$ defines the total stay of the vessel given Q quay cranes. Hence, it should be no smaller than the completion times of all holds, which is guaranteed by Constraint (54). Constraint (55) ensures that cranes cannot cross over each other. Constraints (56) and (57) together ensures that if a crane is assigned to a hold, it should stay at that hold until hold is completely processed. Constraint (58) says that cranes cannot leave vessel until completion.

4.2.2.2 Vessel Level Problem Formulation with Crane Shifting

In the vessel level model with crane shifting, we need not keep track of individual crane locations. It is sufficient instead to find processing start times for each hold while respecting the work continuity constraint, without exceeding a limit on the total number of cranes working during each time period. Hence, we can use the following set of binary variables:

$$z_i^t = \begin{cases} 1 & \text{if work on hold } i \text{ starts at time } t, \\ 0 & \text{otherwise;} \end{cases}$$

Then, the problem can be formulated as a mixed integer program follows:

$$\text{Minimize } p(Q) \tag{60}$$

subject to

$$\sum_{t=1}^T z_i^t = 1 \quad \forall i \tag{61}$$

$$\sum_{i=1}^H \sum_{t=\bar{t}-p^i+1}^{\bar{t}} z_i^t \leq Q \quad \forall i, \bar{t} \in \{p^i, \dots, T\} \tag{62}$$

$$p(Q) \geq tz_i^t + p^i - 1 \quad \forall i, t \tag{63}$$

$$z_i^t \in \{0, 1\} \quad \forall i, t \tag{64}$$

Constraint (61) ensures that work is initiated for each hold. Constraint (62) limits the total number of cranes that we can use at each time period to Q , the number of cranes considered for this problem. $p(Q)$ is the total processing time of the vessel, which must be no less than the completion times of all holds, which is guaranteed by Constraint (63).

4.2.2.3 Master Problem Formulation

The vessel level models determine the minimum total vessel processing times required by each vessel for different number of quay cranes assigned. Recall that we let $p_k(q)$ represent this minimum processing time when using q dedicated cranes to work ship k . Using these results, we now formulate the master problem to determine the number of quay cranes (and which cranes) to be assigned to each vessel to minimize the objective function. Naturally, for each vessel, there is a range for the number of quay cranes

that can be assigned. The lower limit of the range can be set to 1, while the upper limit can be set to h_k , since only one crane can work on a hold at a time.

The following binary variables are used in the model:

$$\begin{aligned}
u_k^q &= \begin{cases} 1 & \text{if } q \text{ cranes are assigned to vessel } k, \\ 0 & \text{otherwise;} \end{cases} \\
v_k^m &= \begin{cases} 1 & \text{if crane } m \text{ is assigned to vessel } k, \\ 0 & \text{otherwise.} \end{cases} \\
\alpha_k^t &= \begin{cases} 1 & \text{if work on vessel } k \text{ starts after time } t, \\ 0 & \text{otherwise.} \end{cases} \\
\beta_k^t &= \begin{cases} 1 & \text{if work on vessel } k \text{ stops before time } t, \\ 0 & \text{otherwise.} \end{cases} \\
z_k^t &= \begin{cases} 1 & \text{if work is in progress at time } t \text{ on vessel } k, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

Then, the problem can be formulated as a mixed integer program:

$$\text{Minimize} \quad \sum_{k=1}^n (c_k - a_k) + \sum_{k=1}^n f_k (c_k - d_k)^+ \tag{65}$$

subject to

$$1 \leq \sum_{m=1}^Q v_k^m \leq h_k \quad \forall k \quad (66)$$

$$\sum_{m=1}^Q v_k^m = \sum_{q=1}^{h_k} q u_k^q \quad \forall k \quad (67)$$

$$t_\ell \geq c_k \quad \forall k, \ell \in \Omega(k) \quad (68)$$

$$c_k \geq t_k + \sum_{q=1}^{h_k} p_k(q) u_k^q \quad \forall k \quad (69)$$

$$c_k \leq t \beta_k^t + (1 - \beta_k^t) T \quad \forall k, t \quad (70)$$

$$t_k \geq t \alpha_k^t + 1 \quad \forall k, t \quad (71)$$

$$\alpha_k^t \geq \alpha_k^{t+1} \quad \forall k, t \quad (72)$$

$$\beta_k^t \leq \beta_k^{t+1} \quad \forall k, t \quad (73)$$

$$\alpha_k^t \geq 1 \quad \forall k, t < a_k \quad (74)$$

$$\beta_k^t \leq 0 \quad \forall k, t < a_k + p_k^{max} \quad (75)$$

$$\alpha_k^t + \beta_k^t + z_k^t = 1 \quad \forall k, t \quad (76)$$

$$0 \leq z_k^t \leq 1 \quad \forall k, t \quad (77)$$

$$L_m^t \geq (z_k^t + v_k^m - 2)M + b_k \quad \forall m, k, t \quad (78)$$

$$L_m^t \leq (2 - z_k^t - v_k^m)M + b_k + h_k - 1 \quad \forall m, k, t \quad (79)$$

$$L_m^t \leq L_{m+1}^t - 1 \quad \forall m, t \quad (80)$$

$$1 \leq L_m^t \leq B \quad \forall m, t \quad (81)$$

$$u_k^q \in \{0, 1\}, \quad v_k^m \in \{0, 1\}, \quad \alpha_k^t \in \{0, 1\}, \quad \beta_k^t \in \{0, 1\} \quad \forall m, k, t, q \quad (82)$$

Constraint (66) sets the range for the number of quay cranes that can be assigned to each vessel where Constraint (67) calculates the number of cranes assigned to each vessel. Overlapping of vessel rectangles is prevented by Constraint (68). The expression $\sum_{q=1}^{h_k} p_k(q) u_k^q$ gives the total berth time of vessel k . Hence, Constraint (69) sets a lower bound on the earliest time that each vessel can leave. Constraints (70), (71), and (76) force the values of the α , β , and z variables to take on correct

values where the z variables are equal to one while a crane is working at a vessel; these z variables are then used in constraints (78) - (79) to track the locations of the Q individual cranes. Constraints (72) - (75) are valid inequalities on binary variables α and β . Note that as a result of Constraint (76) and (77), z variables will always have a value either 0 or 1 and we can define them as continuous variables. Constraints (78) and (79) together keep cranes grouped together on the vessel to which they are assigned; note that the details of what cranes do while assigned to a vessel k cannot be determined by this formulation, but are already known from the vessel-level problem given the assignment of q cranes. Constraint (80) is the non-crossing constraint for cranes, and Constraint (81) keeps each crane on the berth.

4.2.3 Roaming Quay Crane Scheduling

In roaming quay crane scheduling, the terminal operator can move cranes from one vessel to another during processing. Since the number of cranes working a vessel may thus change over time, this problem cannot be easily decomposed like the dedicated scheduling problems. However, depending on whether or not crane shifting is allowed, the following two mixed integer models can be constructed.

4.2.3.1 Problem Formulation with Crane Blocking

In the case with crane blocking, only one crane can work on a hold of a vessel during total loading/unloading period of the hold. To keep track of individual crane assignment and crane location we use the following binary variable set:

$$z_{mki}^t = \begin{cases} 1 & \text{if crane } m \text{ starts working on hold } i \text{ of vessel } k \text{ at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the problem can be formulated as a mixed integer program as follows:

$$\text{Minimize } \sum_{k=1}^n (c_k - a_k) + \sum_{k=1}^n f_k (c_k - d_k)^+ \quad (83)$$

subject to

$$t_\ell \geq c_k \quad \forall k \text{ and } \ell \in \Omega(k) \quad (84)$$

$$\sum_{m=1}^Q \sum_{t=1}^T z_{mki}^t = 1 \quad \forall k, i \quad (85)$$

$$z_{mki}^t \leq 0 \quad \forall m, k, i \quad \forall t < a_k \quad (86)$$

$$c_k \geq tz_{mki}^t + p_k^i \quad \forall m, k, i, t \quad (87)$$

$$t_k \leq tz_{mki}^t + (1 - z_{mki}^t)T \quad \forall m, k, i, t \quad (88)$$

$$c_k \geq t_k + p_k^{max} \quad \forall k, t \quad (89)$$

$$L_m^t \leq L_{m+1}^t - 1 \quad \forall m, t \quad (90)$$

$$L_m^{t+\bar{t}} \geq (z_{mki}^t - 1)B + b_k + i - 1 \quad \forall m, k, i, t, \bar{t} \in \{0, \dots, p_k^i - 1\} \quad (91)$$

$$L_m^{t+\bar{t}} \leq (1 - z_{mki}^t)B + b_k + i - 1 \quad \forall m, k, i, t, \bar{t} \in \{0, \dots, p_k^i - 1\} \quad (92)$$

$$1 \leq L_m^t \leq B \quad \forall m, t \quad (93)$$

$$z_{mki}^t \in \{0, 1\} \quad \forall m, k, i, t \quad (94)$$

In the formulation, overlapping of vessel rectangles is prevented by Constraint (84). Constraint (85) ensures that a crane should be assigned to each hold of each vessel. Arrival time restrictions are handled by Constraint (86). Constraint (87) guarantees that a vessel can depart only after every hold of the vessel is processed, and Constraint (88) ensures that the vessel should dock before any processing work. Constraint (89) is a valid inequality that says each vessel should stay at least the number of time periods required to process the hold with the maximum workload. Constraint (90) is the non-crossing constraint for cranes. Constraints (91) and (92) together keep cranes at the position of the hold that they are assigned until its completion, while Constraint (93) keeps cranes on the berth.

4.2.3.2 Problem Formulation with Crane Shifting

In the roaming crane scheduling model with crane shifting, we need not keep track of individual crane assignments. Instead, we find work start times for each hold without exceeding the limit on the total number of cranes working at each time period. Hence, we can use the following set of binary variables:

$$z_{ki}^t = \begin{cases} 1 & \text{if work starts on hold } i \text{ of vessel } k \text{ at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the problem can be formulated as a mixed integer program as follows:

$$\text{Minimize} \quad \sum_{k=1}^n (c_k - a_k) + \sum_{k=1}^n f_k (c_k - d_k)^+ \quad (95)$$

subject to

$$t_\ell \geq c_k \quad \forall k \text{ and } \ell \in \Omega(k) \quad (96)$$

$$\sum_{t=1}^T z_{ki}^t = 1 \quad \forall k, i \quad (97)$$

$$\sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{t=\bar{t}-p_k^i+1}^{\bar{t}} z_{ki}^t \leq Q \quad \forall \bar{t} \in \{p_k^i, \dots, T\} \quad (98)$$

$$z_{ki}^t \leq 0 \quad \forall k, i \quad \forall t < a_k \quad (99)$$

$$c_k \geq tz_{ki}^t + p_k^i \quad \forall k, i, t \quad (100)$$

$$t_k \leq tz_{ki}^t + (1 - z_{ki}^t)T \quad \forall k, i, t \quad (101)$$

$$c_k \geq t_k + p_k^{max} \quad \forall k, t \quad (102)$$

$$z_{ki}^t \in \{0, 1\} \quad \forall k, i, t \quad (103)$$

Overlapping of vessel rectangles is prevented by Constraint (96). Constraint (97) ensures that work starts on each hold of each vessel. Constraint (98) ensures that we cannot use more than Q cranes at any time. Arrival time restrictions are handled by Constraint (99). Constraint (100) guarantees that a vessel can depart only after every

hold of the vessel is processed, and Constraint (101) ensures that a vessel should dock before any processing work begins. Constraint (102) is a valid inequality that says each vessel should stay at least the number of time periods required to process the hold with the maximum workload.

4.2.4 Analysis of Crane Scheduling Problems

Let C_{Db}^* and C_{Ds}^* denote the optimum objective function value for the dedicated quay crane scheduling problem with crane blocking and crane shifting respectively. Similarly, let C_{Rb}^* and C_{Rs}^* denote the optimum objective function value for the roaming quay crane scheduling problem with crane blocking and crane shifting respectively. Then, we can make the following observation immediately.

Lemma 4 $C_{Rs}^* \leq C_{Rb}^* \leq C_{Db}^*$ and $C_{Rs}^* \leq C_{Ds}^* \leq C_{Db}^*$.

It is easy to see the correctness of Lemma 4 because any feasible solution with dedicated crane assignment is also a feasible solution for roaming crane assignment, and any feasible solution with crane blocking is also a feasible solution with crane shifting. This being said, the question is how much we can improve the terminal operator objective function by allowing roaming and shifting. In order to investigate the magnitude of this gain, we present a computational experiment.

We generated 20 small instances with 6 vessels. Each vessel has 2 to 4 holds, and each instance has at least one vessel of each size. Berth length is set to 7 holds meaning that two 2-hold and one 3-hold vessel can be moored simultaneously. Processing times assigned to each hold of each vessel are related to vessel size. For a 2-hold vessel $p_k^i \sim U^D[1, 4]$, for a 3-hold vessel $p_k^i \sim U^D[1, 5]$, and for a 4-hold vessel $p_k^i \sim U^D[1, 6]$ for the i^{th} hold of vessel k . Furthermore, $a_k \sim U^D[1, 8]$, $d_k = a_k + Kp_k$ where $K \sim U^D[1, 3]$, and $f_k \sim U^D[3, 5]$ for each vessel k .

We first solve the BAP for all instances generated with $p_k = p_k^{max}$. The optimal solution of the BAP not only provides a berth schedule but also generates a lower

bound for the QCSP. This is because only one crane can be assigned to each hold at a time, and hence the handling time of each vessel cannot be smaller than the processing time required by the maximum hold of the vessel. We denote this lower bound as LB_{BS} . The results are summarized in Table 12.

Table 12: Computational analysis of crane scheduling strategies

Instance	LB_{BS}	C_{Rs}^*	C_{Rb}^*	C_{Ds}^*	C_{Db}^*
1	31	31	40	40	41
2	37	37	46	52	52
3	31	33	45	45	45
4	56	56	56	61	61
5	22	28	28	28	28
6	29	29	38	36	45
7	48	48	60	60	60
8	45	63	74	74	74
9	39	41	51	62	70
10	43	43	64	64	66
11	44	50	57	75	75
12	26	28	29	52	52
13	51	52	65	61	75
14	39	42	44	47	47
15	23	23	29	28	29
16	53	53	63	77	77
17	110	110	112	110	115
18	23	31	34	36	36
19	31	39	48	50	50
20	39	39	44	57	62
Base: LB	1.000	1.085	1.285	1.407	1.461
Base: C_{Rs}^*	0.932	1.000	1.188	1.302	1.356

As depicted in Table 12, when roaming and shifting is used, a crane schedule only 8.5% above the lower bound is found on average. When roaming is allowed but shifting is not performed, the optimum objective function value increases by 18.8% compared to the shifting case. With dedicated crane scheduling method, the best solution found is 30.2% worse than the roaming case even when shifting is used, and 35.6% worse when crane blocking is allowed within each vessel.

Because roaming and shifting provides significant efficiency in crane usage, terminal operators would like to utilize these crane scheduling techniques as effectively as

possible. Therefore, in the remainder of the chapter we focus on the roaming quay crane scheduling with shifting, and provide a fast solution method that can handle realistic instances for this version of the QCSP.

4.3 A Tabu Search Algorithm

The mixed integer program (96)-(103) can only be solved for small instances in a reasonable amount of time. Thus, in this section we present a tabu search algorithm which enables us to solve QCSP instances of larger size approximately, but very quickly. We first describe how a good initial solution can be constructed quickly. Then we present the fundamentals of the tabu search algorithm proposed.

4.3.1 Initial Solution

In order to find an initial solution, a greedy heuristic for the QCSP is designed using the following crane scheduling principles provided by [16].

Principle 1: A crane should not be idle if there are any holds on ships that are berthed, but have not yet begun processing.

Principle 2: If any cranes are working on a ship, at least one of them should be assigned to the *maximum hold*. The maximum hold of a ship corresponds to the unprocessed hold that requires the most crane time to finish processing. If cranes are available to be assigned to a vessel, it is sensible to begin work on the maximum hold as soon as possible since no other hold will take as long to complete processing.

These principles, coupled with the given berth schedule, suggest the following heuristic scheduling strategy:

1. **Construct a vessel list \mathcal{L}_v :** We let \mathcal{L}_v be the primal vessel list of the preliminary berth schedule determined by solving the BAP assuming that the ship

processing time is given by the processing time of the maximum hold. As described in Chapter 2, the primal vessel list of schedule x , which is denoted by $\mathcal{L}_v^P(x)$ in this chapter, sorts vessels in the increasing order of their berthing times while breaking ties in favor of the minimum indexed berthing position.

2. Construct a hold list \mathcal{L}_h^k for each vessel k : We construct a hold list \mathcal{L}_h^k for vessel k by following the maximum-hold-first rule; thus, the list contains holds sorted in non-increasing order of hold processing times with ties broken arbitrarily.

3. Construct a global hold list \mathcal{L}_h : The global hold list is constructed by concatenating the individual hold lists \mathcal{L}_h^k for each vessel k in the vessel order given by \mathcal{L}_v .

4. Berth vessels and allocate cranes using \mathcal{L}_h : We select the first unprocessed hold in the list (let k be the vessel index and i be the hold index within the vessel). If this is the first hold of vessel k considered, we find the first time period $t_k \geq a_k$ where berth sections $\{b_k, \dots, b_k + h_k - 1\}$ are unoccupied, and mark them occupied by vessel k for time periods $\{t_k, \dots, T\}$. Then, we find the first time period t_k^i where at least one crane is available in time periods $\{t_k^i, \dots, t_k^i + p_k^i - 1\}$. We reduce the number of available cranes at those time periods by 1, and mark the i^{th} hold of vessel k as processed. If this is the last hold of vessel k considered, we set $c_k = \max_i \{t_k^i + p_k^i\}$ and mark berth sections $\{b_k, \dots, b_k + h_k - 1\}$ as unoccupied for time periods $\{c_k, \dots, T\}$. We continue until all holds in \mathcal{L}_h are processed.

4.3.2 Fundamentals of the Tabu Search

Any feasible solution to the QCSP with crane roaming and shifting can be encoded by a global hold list \mathcal{L}_h , which serves as a priority list, assuming that this encoding is decoded using the method given in Step 4 of the method for developing an initial solution; from now on, we will call this the first-fit hold list decoding. However, as in the BAP, there may exist multiple hold lists that decode to the same crane schedule.

Therefore, for any given crane schedule z , we define a unique priority list called the *primal hold list*, $\mathcal{L}_h^P(z)$. The primal list ordering of holds is created by sorting holds in non-decreasing order of work start times, breaking ties in favor of the hold positioned in the minimum index berth section. Note that the first-fit hold decoding can be performed in $O(\hat{n})$ time, where \hat{n} is the total number of holds in the problem.

Unlike the case with the vessel list encodings used in the tabu search that we develop for the BAP, not every hold list can be decoded without ambiguity into a feasible solution. Consider two vessels that occupy some of the same berth sections in the berth schedule. In order for the hold list to be used to decode unambiguously into a feasible solution, all holds of the vessel scheduled to berth earlier should be processed before all holds of the vessel scheduled to berth later. Therefore, every hold of the first vessel should be listed before any hold of the second vessel in the list. The initial hold list generated by the heuristic algorithm always yields a feasible decoding since holds of each vessel are grouped together in the list.

At each iteration of the tabu search algorithm, we modify the current \mathcal{L}_h using single swap moves within a randomized iteration-dependent neighborhood. The iteration-dependent neighborhood changes depending on the iteration count of the search. In implementation, we divide the search iterations into n distinct stages, where n is the number of vessels. At stage $s \in \{1, \dots, n\}$, given a global hold list \mathcal{L}_h , we define (p, r, q) randomized neighborhood of lists $\mathcal{N}_{\mathcal{L}}^s(p, r, q, \mathcal{L}_h)$ as all lists generated by a single swap in \mathcal{L}_h of the position of one of q selected holds (denoted as hold i) with the position of one of p randomly-selected holds among the r closest neighbor holds to hold i in \mathcal{L}_h . Closeness in \mathcal{L}_h is measured as the absolute difference in position in the list. The set of q holds are determined by first selecting a vessel and then randomly picking a hold within the selected vessel. Here, the probability of selecting a hold from vessel k at stage s is denoted by Π_k^s and calculated as follows.

$$\Pi_k^s = \frac{\pi_k^s}{\sum_{k=1}^n \pi_k^s} \quad (104)$$

where

$$\pi_{k(j)}^s = (n - |j - s|)^\alpha \quad \forall j, s \in \{1, \dots, n\}. \quad (105)$$

In this expression, $k(j)$ is the index of the j^{th} vessel in the vessel primal list $\mathcal{L}_v^P(x)$ of the current schedule x , and $\alpha \geq 0$ is a guiding parameter set by the user. If $\alpha = 0$, holds of all vessels are selected with equal probability to generate swap moves. On the other hand, if $\alpha > 0$, holds of vessels planned to berth earlier are selected with higher probability in the beginning of the search, and as search progresses the probability of selecting holds of vessels planned to berth later increases. The intuition is that any improvement obtained at an iteration by changing the latter portions of the schedule can be lost if an earlier portion of the schedule is modified in the subsequent iterations, whereas the reverse is not true. Computational experiments are performed for both $\alpha = 1$ and $\alpha = 0$ cases to test the effectiveness of the guiding mechanism.

As mentioned earlier, some hold priority lists visited during the search may not feasibly decode into a crane schedule. The infeasibility of a list, however, is easily detected. A list created by a move that swaps positions of a hold of vessel ℓ and a hold of vessel k can be determined using the following procedure.

1. If $k = \ell$, the move is feasible; stop. Else, go to 2.
2. If $k \in \Omega(\ell)$ or $\ell \in \Omega(k)$, the move is infeasible; stop. Else, go to 3.
3. Let u_k and u_ℓ be the list positions of the hold of vessel k and the hold of vessel ℓ selected by the corresponding move respectively where $u_k < u_\ell$. Let $v(j)$ denotes the vessel of the j^{th} hold in the current \mathcal{L}_h . If $v(j) \in \Omega(k)$ or $\ell \in \Omega(v(j))$ for any $u_k < j < u_\ell$, the move is infeasible; stop. Else, the move is feasible; stop.

At the beginning of the search, setting L_h to the primal hold list reduces the probability of creating many infeasible moves since holds that can be processed simultaneously are also positioned close to each other in the list. However, as the search proceeds, this list structure can deteriorate and we may generate more infeasible moves. Therefore, at the end of each stage, we update the current solution to be the best solution found so far, and also set the current hold list to the primal hold list of the best solution.

At the end of each iteration, we also place the reverse of the move selected into a tabu list for θ iterations, where θ is a positive integer distributed discrete random uniform on $[\theta^{min}, \theta^{max}]$. A reverse move is defined as swapping back the positions of the same two holds in \mathcal{L}_h .

4.3.3 Steps of the Tabu Search Algorithm

In this section, we provide step-by-step detail for the proposed search algorithm. Let z^0 be the initial solution, and let z and z^* be the current and the best solution found by the tabu search respectively. Let $C(z)$ represent the objective function value for solution z .

Step 1: Initialization. Set $s = 1$. Compute Π_k^s . Set p , r , and q . Let z^0 be the initial solution found with the primal hold list $\mathcal{L}_h^P(z^0)$. Let $\mathcal{L}_h = \mathcal{L}_h^P(z^0)$, $z = z^* = z^0$. Initialize the counter $t^T = 0$ which denotes the total number of iterations at current stage. Set the value of t^M , which is the maximum number of iterations allowed at each stage.

Step 2: Neighborhood Search. Set $t^T = t^T + 1$. For each candidate list $\mathcal{L}_h^i \in \mathcal{N}_{\mathcal{L}}^s(p, r, q, \mathcal{L}_h)$, first check feasibility. If \mathcal{L}_i is feasible, decode it into crane schedule z_i . Construct a list \mathcal{U} by ranking the feasible moves in nondecreasing order of C . Select the first move i in \mathcal{U} . If the move is not tabu or if $C(z_i) < C(z^*)$,

then let a be the move i . If the move is tabu or does not improve the best solution, consider the next move in \mathcal{U} and repeat the check.

Step 3: Solution Update. Set $z = z_a$. If $C(z) < C(z^*)$, set $z^* = z$. Update the tabu list by inserting the reverse swap move associated with solution z_a into the tabu list for the appropriate random number of iterations, and removing reverse moves whose duration has expired. If $t^T < t^M$, update $t^T = t^T + 1$ and go to Step 2. Else if $s < n$, set $t^T = 0$, update $s = s + 1$, recompute Π_k^s , set $z = z^*$ and $\mathcal{L}_h = \mathcal{L}_h^P(z^*)$ and go to Step 2. Else, stop. The best solution found is z^* .

4.4 The Crane Assignment Problem

In this section we introduce the problem of individual crane assignment (CAP). It is the problem of assigning specific quay cranes to optimum work locations determined by the QCSP with the objective to minimize the total distance traveled by cranes on the berth over the planning horizon. This problem arises when crane shifting is used while scheduling cranes. In the QCSP with crane shifting, we do not keep track of individual crane positions because any blocking can be resolved by shifting cranes appropriately. Hence, we only determine the time periods that we need to work on each hold. Since the location of vessels are given by the berth schedule, the output of the QCSP provides the locations of work at each time period that we require cranes. If the total number of quay cranes required at a given time period is equal to the number of available cranes, then the assignment is both trivial and unique. We should assign them according to their relative positions on the rail since they cannot pass each other. However, if the total number of cranes required at any time period is less than the total number of available cranes, the decision of assigning cranes to work positions at that time period affects the total distance traveled by cranes during the planning horizon.

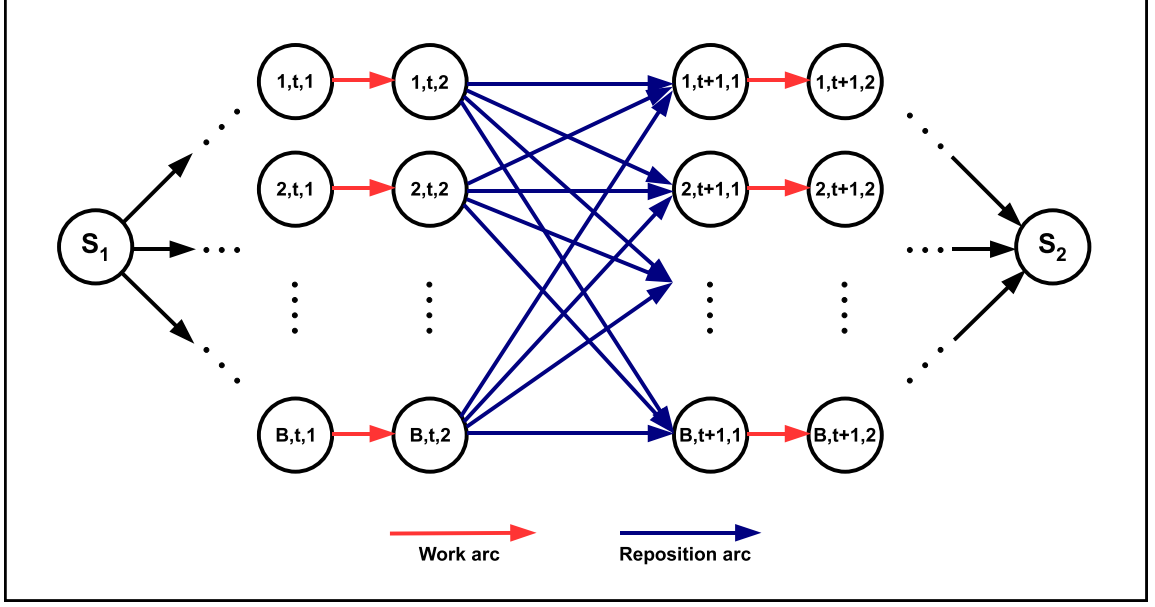


Figure 14: Illustration of the graph constructed to solve the CAP

To solve the CAP, we construct the graph $G = (\mathcal{N} \cup \mathcal{S}_1 \cup \mathcal{S}_2, \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4)$. The node set \mathcal{N} is generated by defining two nodes $(i, t, 1)$ and $(i, t, 2)$ for each berth section - time period pair (i, t) where $1 \leq i \leq B$ and $1 \leq t \leq T$. Here, B is the size of the berth in number of berth sections, and T is the completion time of the last vessel in the planning horizon. The arc set consists of four disjoint subsets \mathcal{A}_1 , \mathcal{A}_2 , \mathcal{A}_3 , and \mathcal{A}_4 . \mathcal{A}_1 includes the arcs from $(i, t, 1)$ to $(i, t, 2)$ for all (i, t) pairs. We call these arcs work arcs. \mathcal{A}_2 is the set of arcs defined from $(i, t, 2)$ to $(j, t+1, 1)$ for all $1 \leq i, j \leq B$ and $1 \leq t < T$. We call these arcs reposition arcs. The flow cost associated with each reposition arc is defined as $c_{(i,t,2)}^{(j,t+1,1)} = |i - j|$, which is the number of berth sections between i and j . No flow cost is assigned to work arcs. We also define source and sink nodes, \mathcal{S}_1 and \mathcal{S}_2 . \mathcal{A}_3 denotes the set of arcs emanating from \mathcal{S}_1 to nodes $(i, 1, 1)$ whereas \mathcal{A}_4 is the set of arcs defined from $(i, T, 2)$ to \mathcal{S}_2 for all i . No flow cost is also associated with these arcs.

When cross-over constraints are relaxed, the CAP problem can be solved as a minimum cost flow problem on the graph constructed. To do this, we place an inflow

and outflow of Q (total number of quay cranes) units at nodes S_1 and S_2 respectively. Furthermore, the upper bound on the flow on each arc is set 1 where the lower bound on work arcs between $(i, t, 1)$ and $(i, t, 2)$ is also set equal to 1 if a crane is needed at berth section i at time t . Thus, a feasible solution determines Q disjoint paths between \mathcal{S}_1 and \mathcal{S}_2 that cover the work required on the berth. The resulting linear program is provided below, where $f_{(i,t,k)}^{(j,p,l)}$ denotes the flow on arc defined from (i, t, k) to (j, p, l) where $1 \leq i, j \leq B$, $1 \leq t, p \leq T$, and $1 \leq k, l \leq 2$.

$$\text{Minimize} \quad \sum_{i=1}^B \sum_{j=1}^B \sum_{t=1}^{T-1} c_{(i,t,2)}^{(j,t+1,1)} f_{(i,t,2)}^{(j,t+1,1)} \quad (106)$$

subject to

$$\sum_{j=1}^B f_{(j,t-1,2)}^{(i,t,1)} - f_{(i,t,1)}^{(i,t,2)} = 0 \quad \forall i, t \quad (107)$$

$$\sum_{j=1}^B f_{(i,t,2)}^{(j,t+1,1)} - f_{(i,t,1)}^{(i,t,2)} = 0 \quad \forall i, t \quad (108)$$

$$\sum_{j=1}^B f_{S_1}^{(j,1,1)} = Q \quad (109)$$

$$\sum_{j=1}^B f_{(j,T,2)}^{S_2} = Q \quad (110)$$

$$0 \leq f_{(i,t,1)}^{(i,t,2)} \leq 1 \quad \forall i, t \quad (111)$$

$$0 \leq f_{(i,t,2)}^{(j,t+1,1)} \leq 1 \quad \forall i, j, t \quad (112)$$

$$0 \leq f_{S_1}^{(j,1,1)} \leq 1 \quad \forall j \quad (113)$$

$$0 \leq f_{(j,T,2)}^{S_2} \leq 1 \quad \forall j \quad (114)$$

$$f_{(i,t,1)}^{(i,t,2)} = 1 \quad \forall (i, t) \text{ requiring work} \quad (115)$$

The linear program (107)-(115) has a totally unimodular coefficient matrix and provides an integer optimal solution. Therefore, when cross-over constraints are relaxed, it determines the optimal quay crane paths (movements) in polynomial time.

Lemma 5 *Crane paths can cross over in the optimal solution found by the linear program (107)-(115), but an alternate optimal where crane paths do not cross over always exists.*

Proof. Consider a feasible solution to the linear model (107)-(115), where paths of two cranes cross over at some time period. Let us assume that the first crane moves from berth section l_1 to l_2 , the second one moves from berth section l_3 to l_4 , and $l_1 < l_3$. Since the paths cross over, we know that $l_4 < l_2$. The cost of such a repositioning is $|l_2 - l_1| + |l_4 - l_3|$, and one of the six cases below should hold.

1. $l_1 < l_3 \leq l_4 < l_2$
2. $l_1 < l_3 \leq l_4 < l_2$
3. $l_1 \leq l_4 < l_2 \leq l_3$
4. $l_4 \leq l_1 \leq l_2 \leq l_3$
5. $l_4 \leq l_1 < l_3 \leq l_2$
6. $l_1 \leq l_4 \leq l_3 \leq l_2$

If case 1 or 2 holds, $|l_2 - l_1| + |l_4 - l_3| = |l_2 - l_3| + |l_4 - l_1|$. If case 3, 4, 5, or 6 holds, $|l_2 - l_1| + |l_4 - l_3| = l_2 - l_1 + l_3 - l_4 > l_3 - l_2 + l_4 - l_1 = |l_4 - l_1| + |l_2 - l_3|$. This means that there exists another feasible solution where the first crane moves from l_1 to l_4 , and the second crane moves from l_3 to l_2 . Since $l_1 < l_3$ and $l_4 < l_2$, cranes do not cross over, and the objective function is not higher. \square

Theorem 7 *The CAP can be solved in polynomial time.*

Proof. Lemma 5 indicates that if crane paths cross over in the optimal solution, either case 1 or 2 should hold, and an alternate optimal solution can be constructed as described above where no cross over exists. Such an alternate solution can be defined by assigning cranes to the work arcs carrying unit flow in the order of crane index, because there exist exactly Q work arcs carrying unit flow at each time period.

□

4.5 Computational Experiments

The test instances generated for the BAP were also used in computational experiments conducted for the QCSP. We again use the six sets of instances where the first three sets include relatively small problems for a terminal with $B = 12$ and 10 to 14 vessels, and the second set contains larger instances with $B = 20$ and 20 to 30 vessels. We set $Q = 6$ for small instances and $Q = 10$ for large ones. For each vessel k , we generated h_k integers $p_k^1, \dots, p_k^{h_k}$ randomly and selected the maximum of these as the processing time for vessel k in the BAP. For the QCSP, the integer p_k^i now more specifically represents the processing time required by hold i of vessel k . Furthermore, since $\max_i\{p_k^i\}$ was used as the processing time of vessel k in the BAP, the best solutions found for the BAP problem defined on these instances has not only generated a berth schedule required by the QCSP but also provided a lower bound on the cost of that solution, LB_{BS} . This lower bound essentially is one that assumes that an unlimited supply of cranes is available at any time period.

Table 13: The tabu search parameters used in the computational experiments

Parameter	Value
p	5
r	10
q	5
t^M	25
θ^{min}	5
θ^{max}	10

The parameter values used in the computational experiments are provided in Table 13. The parameter t^M is the number of iterations completed during each stage. Since we have n stages, the total number of iterations performed is $t^M n$.

Table 14 and Table 15 summarize results for small and large instances respectively. As indicated before, the best BAP solutions for these instances provide the

Table 14: Summary of results for small QCSP instances

Instance	IS	LB_{BS}	LB_{MIP}	UB_{MIP}	$\alpha = 1$	$\alpha = 0$	$\alpha = 1$ (%)	$\alpha = 0$ (%)
10	111	90	106	106	106	106	24	24
11	74	54	65	65	65	65	45	45
12	176	159	159	159	159	159	100	100
13	149	74	100	100	100	101	65	64
14	93	54	67	67	67	67	67	67
15	111	81	83	83	83	83	93	93
16	92	60	86	86	86	86	19	19
17	165	138	160	160	160	160	19	19
18	181	131	161	161	161	167	40	28
19	147	115	135	135	135	135	38	38
20	209	131	164	164	164	164	58	58
21	198	170	182	182	182	182	57	57
22	185	123	158	158	158	159	44	42
23	157	124	134	134	134	134	70	70
24	126	79	96	96	96	96	64	64
25	203	121	184	184	184	184	23	23
26	158	120	154	154	154	154	11	11
27	181	164	164	164	164	164	100	100
28	200	117	175	175	175	175	30	30
29	197	119	171	171	171	171	33	33
30	229	156	195	195	195	195	47	47
31	383	247	333	333	333	333	37	37
32	160	133	145	145	145	149	56	41
33	400	250	297	336	323	323	51	51
34	219	166	196	196	200	200	36	36
35	298	213	254	254	254	256	52	49
36	184	142	161	161	161	161	55	55
37	170	130	134	134	134	134	90	90
38	478	332	441	441	441	441	25	25
39	397	329	356	356	356	357	60	59

berth schedule input for the QCSP. LB_{BS} denotes the objective function value of the corresponding berth schedules and yields a lower bound for the QCSP.

The mixed integer program presented in Section 4.2.3.2 was applied to all instances using CPLEX with default parameters on an Intel 2.4 GHz Pentium processor running Linux with 2 GB memory. MIP runs were limited to 2 hours. For 29 of the small instances and for 18 of the large instances, the optimal solution was found by CPLEX. For the remaining, the lower bound LB_{MIP} and the upper bound UB_{MIP} are provided. In instance 33, where MIP failed to prove optimality, the tabu search with $\alpha = 1$ provided a better solution, and in instance 34, where the MIP found the optimal solution, the tabu search found a worse solution. In the remaining 28

small instances, the tabu search finds the optimal solution when $\alpha = 1$. Similarly, in 24 of the large instances, the search algorithm found solutions with a better or same objective function value, and in only 6 cases it found slightly worse solutions compared to the MIP.

The last two columns present the improvement in the lower bound gap provided by the search algorithm in the $\alpha = 1$ and $\alpha = 0$ cases respectively. The lower bound gap is calculated with respect to LB_{BS} , and improvement is calculated as (initial gap - final gap)/initial gap. When the two cases are compared, we see that in 6 of the small instances, the search with $\alpha = 1$ provided better solutions. In the remaining, equivalent solution are found by both searches. For large instances, setting $\alpha = 1$ provided better solutions in 21 instances, and keeping $\alpha = 0$ found better crane schedules in only 2 cases. In the remaining 7 instances, solutions with the same objective function value were found by both approach. This indicates that the additional guiding mechanism generally helps to find better solutions, and it becomes more effective as problem size increases.

Finally, Table 16 presents the computational times in seconds observed during tests. We see that MIP provides the optimal solution for small instances in set 1 and 2 fairly quickly, while the run time limit is reached for many of the larger instances. On the other side, with the parameters used, the tabu search algorithm found optimal or near optimal solutions almost instantaneously for small instances and in 13 seconds for large ones. Since the number of iterations performed are equal for $\alpha = 1$ and $\alpha = 0$ cases, total run times are similar. However, we observe that in the $\alpha = 1$ case, the best solution was found later while tabu search got stuck at local optima in the earlier stages when $\alpha = 0$. Therefore, as expected, the guiding mechanism designed provides a more patient but consistent search.

Table 15: Summary of results for large QCSP instances

Instance	IS	LB_{BS}	LB_{MIP}	UB_{MIP}	$\alpha = 1$	$\alpha = 0$	$\alpha = 1$ (%)	$\alpha = 0$ (%)
40	121	112	118	118	118	121	33	0
41	167	124	128	128	128	129	91	88
42	308	134	162	236	236	243	41	37
43	163	119	124	124	125	125	86	86
44	81	76	78	78	78	80	60	20
45	136	111	111	111	111	111	100	100
46	225	132	137	175	171	179	58	49
47	186	129	144	144	144	158	74	49
48	102	96	96	96	96	96	100	100
49	137	112	112	112	112	113	100	96
50	357	248	268	286	286	294	65	58
51	156	112	124	124	124	124	73	73
52	459	226	244	390	344	348	49	48
53	331	184	221	282	274	275	39	38
54	614	366	425	425	427	432	75	73
55	269	173	180	180	180	180	93	93
56	271	153	183	242	221	229	42	36
57	124	82	89	94	95	100	69	57
58	352	175	196	319	300	306	29	26
59	295	223	243	243	246	251	68	61
60	431	331	351	351	351	364	80	67
61	787	470	530	686	656	653	41	42
62	1157	866	899	1092	1039	1039	41	41
63	586	413	475	475	479	476	62	64
64	293	228	239	239	240	245	82	74
65	517	336	383	383	383	384	74	73
66	713	392	448	548	535	572	55	44
67	522	370	417	475	465	465	38	38
68	274	174	203	203	203	204	71	70
69	458	208	288	288	288	313	68	58

Table 16: Summary of computation times observed in seconds

Set	MIP	Best Solution Time		Total Solution Time	
	Time	$\alpha = 1$	$\alpha = 0$	$\alpha = 1$	$\alpha = 0$
1x	13.3	0.1	0.2	1.4	1.2
2x	61.0	0.3	0.2	1.3	1.5
3x	1334.7	0.6	0.6	1.9	2.3
4x	1458.4	2.2	1.0	4.1	4.5
5x	4667.5	6.3	2.2	8.2	9.0
6x	5048.0	9.1	2.8	12.3	12.7

CHAPTER V

THE SIMULTANEOUS BERTH AND QUAY CRANE SCHEDULING PROBLEM

5.1 Introduction

In this chapter, we focus on the simultaneous scheduling of berth and quay cranes. In practice, berth scheduling and quay crane scheduling problems are generally considered sequentially by terminal operators. They first determine a berth schedule using estimates of the duration of berthing for each vessel. Then, they try to split quay cranes efficiently between the vessels that are planned to moor simultaneously at the same berth. Such sequential planning can be achieved using the models developed in this thesis: first solve the BAP introduced in Chapter 2, and use its output as an input to solve the QCSP mentioned in Chapter 4. In this chapter, we develop models and methods for scheduling these two resources simultaneously, namely the simultaneous berth and quay crane scheduling problem (BQCSP).

The number of cranes that a terminal operator can assign to a vessel depends on the number of cranes used by other vessels simultaneously moored at the berth. Assigning a crane to a vessel is equivalent to reducing the number of cranes that can be assigned to other vessels. Naturally, the total number of cranes that a terminal operator can utilize at each time period cannot exceed the total number of cranes on hand. This limitation can cause an inferior crane allocation for some vessels compared to what is projected in the berth scheduling phase, and thus vessels may be forced to stay longer than expected at berth. Such delays then may affect vessels to be moored later. A terminal operator may be able to determine a better and more accurate operational plan if actual crane requirements are considered while

determining berth schedules, which is the motivation behind simultaneous berth and quay crane scheduling.

Reference [16] introduced the problem of quay crane scheduling. Although the paper assumes a berth of infinite length, it discusses the impact of berth limitations on quay crane schedules. However, most studies on quay crane scheduling also assume infinite length berths or simply focus on crane planning for a single vessel (see Chapter 4). Similarly, most papers published on the berth allocation problem assume a known ship processing duration, and do not consider the impact of crane availability (see Chapter 2).

The problem of simultaneously scheduling berth and quay cranes was first introduced in [58]. The authors formulate an integer programming model which determines the berthing position on a continuous berth and the berthing time of each vessel, as well as the number of cranes assigned to each vessel at each time period. However, the paper assumes that vessel processing times decrease linearly with the number of quay cranes assigned to a vessel. The authors agree that this assumption is not realistic, and does not agree with the models used by other studies on crane scheduling (including the simple one used in [16]). This is because of the fact that the work required on a vessel is generally not evenly distributed on the vessel.

Researchers in [11] also draw attention to the relationship between berth allocation and quay crane scheduling. The problem they define considers a discrete berth structure where each berth can serve one vessel at a time and a number of quay cranes which can be moved from one berth to another when required. Vessel handling time is dependent on the berth where it is assigned; however, the number of cranes required by each vessel is known, and hence the impact of crane assignment on handling time is not considered. In a simplification, the paper assumes that vessel handling does not begin until the predetermined number of cranes are available at the corresponding berth. The objective of the problem considered is to minimize the total time spent by

vessels at the port. They present a mixed integer program, and introduce a solution method based on a genetic algorithm.

The problem variant introduced in this chapter is defined for a long continuous berth which can handle multiple vessels simultaneously. On this berth, a number of identical quay cranes operates on a single set of rails for container loading and unloading. We consider the case where vessel processing time depends on the number of quay cranes assigned to the vessel, where a ship is considered processed once cranes have completed the work of a set of holds identical to those introduced in Chapter 4. In this model, each vessel is divided along its length into holds of 3 or 4 container rows where at most one quay crane can work on a hold in a time period. We also assume that once started, work needs to continue on a hold until completion. Note that these assumptions do not prevent assigning more than one crane to a vessel, and cranes can be shifted from hold to hold both within ships and between ships, as long as cranes are not allowed to pass one another. Consequently, the model we develop determines the berthing position and berthing time of each vessel while simultaneously determining when to assign a quay crane to each hold of each vessel for container loading and unloading. The primary contributions of this study include:

- The introduction of a BQCSP variant which considers a continuous berth structure and detailed crane assignments to vessel holds;
- The introduction of a lower bound that can be found in polynomial time for any instance of the BQCSP, where the bound is provably tighter than the linear programming relaxation bound for a standard integer programming model for the problem; and
- The development of an effective tabu search algorithm which utilizes a nested neighborhood search procedure.

5.2 Problem Formulation

We consider a berth modeled as a continuous structure having B equal size sections and Q identical quay cranes operating on a single set of rails. We discretize time and assume an infinite planning horizon (T time periods where T is large). For modeling purposes, each vessel is divided into equal size sections that we call holds. Each hold consists of 3 or 4 container rows. Therefore, the length of a vessel can be represented by the number of holds it has. Furthermore, the length of a berth section is also set equal to the length of a hold, and it is assumed that the berth is B holds in total length. When berthed, a vessel covers a number of adjacent berth sections depending on its length. Multiple vessels can moor at the berth and receive service simultaneously. We assume that only one crane can work on a hold at a given time period. Note that this does not prevent more than one crane working on the same vessel if vessel has multiple holds that need service. Each hold requires a known processing time, and we require that when started, work continues without interruption until the container loading and unloading is done for the hold even though the crane working on the hold may change at each time period. Therefore, we allow crane roaming and shifting which is described in detail in Chapter 4. A vessel can leave the port only after container loading and unloading is completed on every hold.

We concentrate on the dynamic version of the problem where we have a set \mathcal{V} of vessels with known arrival times, where $n = |\mathcal{V}|$. For each vessel $k \in \mathcal{V}$, we define:

h_k : number of holds of vessel k ,

p_k^i : processing time of hold i of vessel k ,

p_k^{max} : maximum hold processing time for vessel k ($p_k^{max} = \max_i \{p_k^i\}$),

a_k : arrival time of vessel k ,

d_k : due time of vessel k (where $d_k \geq a_k + p_k^*$),

f_k : lateness penalty for vessel k ,

b_k : berthing position of vessel k ,

t_k : berthing time of vessel k ,

c_k : the earliest time that vessel k can depart.

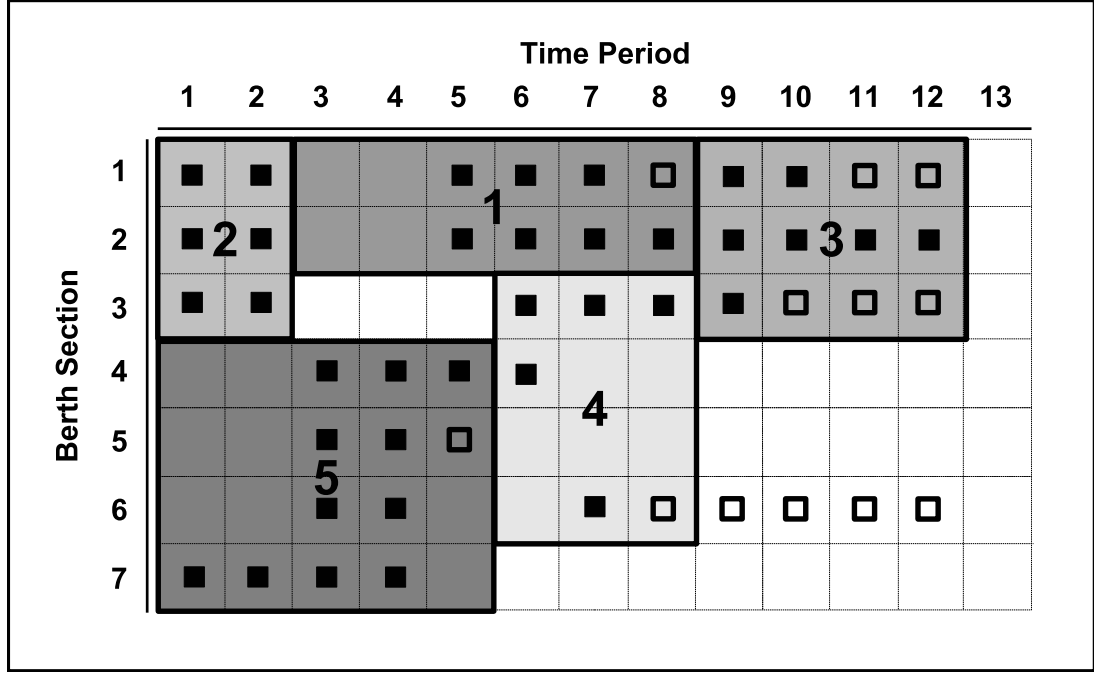


Figure 15: Representation of a BQCSP solution on the time-space diagram

A feasible solution x of the BQCSP consists of a berth schedule and a crane schedule. Any such feasible solution x can be represented by a time-space diagram where the horizontal axis measures time and the vertical axis represents berth sections; see Figure 15. In such a representation, a vessel can be represented by a rectangle whose length is its duration of stay at berth and height is its length (number of holds). Furthermore, crane locations are also depicted using solid and empty squares, where solid and empty squares represent busy and idle cranes respectively. Given a known vector of arriving vessel information $\{h_k, p_k^1, \dots, p_k^{h_k}, a_k, d_k, f_k\}$, the optimization problem is then to find berthing section b_k , berthing time period t_k , and a crane allocation plan which determines the duration of stay for each vessel k and hence c_k . If we say vessel k is berthed at position b_k at time t_k , we mean berth sections $[b_k, b_k + h_k - 1]$ are occupied by vessel k for time periods $[t_k, c_k - 1]$. This also means that once a vessel

is berthed, its location cannot be changed during service.

Our objective is the same as in the BAP and the QCSP which is to minimize $\sum_{k \in \mathcal{V}} (c_k - a_k) + \sum_{k \in \mathcal{V}} f_k (c_k - d_k)^+$ representing the sum of the dwell times and the total penalty accrued by tardy vessels.

Table 17: A sample BQCSP instance

k	1	2	3	4	5
a_k	2	1	3	2	1
d_k	8	4	11	5	5
f_k	3	4	3	3	4
h_k	2	3	3	4	4
p_k^1	3	2	2	3	3
p_k^2	4	2	4	1	2
p_k^3	-	2	1	0	2
p_k^4	-	-	-	1	4

An example instance of the problem is provided in Table 17, where $B = 7$ and $Q = 4$. A feasible solution for this instance is provided in Table 18 where z_k^i denotes the work start time of hold i of vessel k . The solution is also schematized in Figure 15.

Table 18: A feasible solution to the sample BQCSP instance

k	1	2	3	4	5
b_k	1	1	1	3	4
t_k	3	1	9	6	1
c_k	9	3	13	9	6
z_k^1	5	1	9	6	3
z_k^2	5	1	9	6	3
z_k^3	-	1	9	-	3
z_k^4	-	-	-	7	1

The problem can be formulated as a mixed integer program. The model provided below combines the RPF provided for the BAP, and the model presented for the roaming QCSP with crane shifting. The primary decision variables are:

$$x_{\ell k} = \begin{cases} 1 & \text{if vessel } k \text{ berths after vessel } \ell \text{ departs,} \\ 0 & \text{otherwise;} \end{cases}$$

$$y_{\ell k} = \begin{cases} 1 & \text{if vessel } k \text{ berths completely above vessel } \ell \text{ on the time-space diagram,} \\ 0 & \text{otherwise.} \end{cases}$$

$$z_{ki}^t = \begin{cases} 1 & \text{if work starts on hold } i \text{ of vessel } k \text{ at time } t, \\ 0 & \text{otherwise;} \end{cases}$$

A feasible selection of these primary variables then constrain the secondary set of decisions $\{b_k, t_k, c_k\}$ for all ships k , modeled as continuous variables. The formulation is:

$$\text{Minimize} \quad \sum_{k=1}^n (c_k - a_k) + \sum_{k=1}^n f_k (c_k - d_k)^+ \quad (116)$$

$$x_{\ell k} + x_{k\ell} + y_{\ell k} + y_{k\ell} \geq 1 \quad \forall k, \ell \in \mathcal{V} \text{ and } k < \ell \quad (117)$$

$$x_{\ell k} + x_{k\ell} \leq 1 \quad \forall k, \ell \in \mathcal{V} \text{ and } k < \ell \quad (118)$$

$$y_{\ell k} + y_{k\ell} \leq 1 \quad \forall k, \ell \in \mathcal{V} \text{ and } k < \ell \quad (119)$$

$$t_\ell \geq c_k + (x_{k\ell} - 1)M \quad \forall k, \ell \in \mathcal{V} \text{ and } k \neq \ell \quad (120)$$

$$b_\ell \geq b_k + h_k + (y_{k\ell} - 1)M \quad \forall k, \ell \in \mathcal{V} \text{ and } k \neq \ell \quad (121)$$

$$t_k \geq a_k \quad \forall k \in \mathcal{V} \quad (122)$$

$$t_k \leq tz_{ki}^t + (1 - z_{ki}^t)T \quad \forall k \in \mathcal{V} \forall i \in \{1, \dots, h_k\} \forall t \in \{1, \dots, T\} \quad (123)$$

$$c_k \geq tz_{ki}^t + p_k^i \quad \forall k \in \mathcal{V} \forall i \in \{1, \dots, h_k\} \forall t \in \{1, \dots, T\} \quad (124)$$

$$c_k \geq t_k + p_k^{max} \quad \forall k \in \mathcal{V} \quad (125)$$

$$\sum_{t=1}^T z_{ki}^t = 1 \quad \forall k \in \mathcal{V} \forall i \in \{1, \dots, h_k\} \quad (126)$$

$$\sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{t=\bar{t}-p_k^i+1}^{\bar{t}} z_{ki}^t \leq Q \quad \forall \bar{t} \in \{p_k^i, \dots, T\} \quad (127)$$

$$b_k \leq B - h_k + 1 \quad \forall k \in \mathcal{V} \quad (128)$$

$$b_k \geq 1 \quad \forall k \in \mathcal{V} \quad (129)$$

$$x_{\ell k} \in \{0, 1\}, \quad y_{\ell k} \in \{0, 1\} \quad \forall k, \ell \in \mathcal{V} \text{ and } k \neq \ell \quad (130)$$

$$z_{ki}^t \in \{0, 1\} \quad \forall k \in \mathcal{V} \forall i \in \{1, \dots, h_k\} \forall t \in \{1, \dots, T\} \quad (131)$$

Constraints (117) through (119) guarantee that no vessel rectangles overlap. Constraints (120) and (121) ensure that the selected berthing times and berthing positions are consistent with the definitions of $x_{\ell k}$ and $y_{\ell k}$, where M is a large positive scalar. Constraint (122) forces berthing to occur no earlier than arrival time, and Constraint (123) forces hold processing to start after berthing. Constraint (124) ensures that vessels depart only after all holds are processed, and Constraint (125) is a valid inequality that provides a lower bound on c_k given t_k . Constraint (126) ensures that work starts on each hold of each vessel and Constraint (127) ensures that no more

than Q quay cranes are used at any time period. Constraints (128) and (129) guarantee that all vessels fit on the berth.

5.3 Lower Bound Analysis

In this section, we introduce two polynomially-computable lower bounds for the objective function value of the dynamic BQCSP.

5.3.1 Lower Bound 1

The first lower bound is obtained by relaxing the crane capacity constraint. If we assume that $Q = B$, a quay crane can be positioned at each berth section. In such a case, each hold of any given vessel can be assigned a crane immediately after the vessel is berthed. This means that vessel k is always processed in p_k^{max} time periods when berthed. Therefore, the resulting problem is equivalent to the dynamic BAP introduced in Chapter 2 with $p_k = p_k^{max}$ for vessel $k \in \mathcal{V}$. This implies that the polynomially-computable lower bound introduced for the BAP in Chapter 2 is also valid for the BQCSP. We call this lower bound LB_1 in this chapter.

Theorem 8 $LB_1 \geq LB_{LP}$ where LB_{LP} denotes the objective function value of an optimal solution to the LP relaxation of the mixed integer program (117)-(131).

Proof. In a feasible solution of the LP relaxation, it is easy to see that berthing times t_k and positions b_k can be chosen such that any vessel rectangles may overlap one another. Furthermore, z variables can be selected in such a way that Constraint (125) becomes binding, and $c_k = a_k + p_k^{max}$. The rest of the proof follows the discussion presented in the proof of Theorem 4 in Chapter 2.

5.3.2 Lower Bound 2

The second lower bound is obtained by relaxing the berth length limitation. If we assume that $B = \sum_{k=1}^n h_k$, a vessel can be berthed immediately upon arrival. Then,

using a similar idea utilized in construction of LB_1 in Chapter 2, we can construct a corresponding parallel machine scheduling problem P_2 using the relaxation illustrated partially in Figure 16. First, each quay crane is treated as a separate parallel machine; thus, Q identical machines are defined. Next, for hold i of vessel k , we create p_k^i jobs, each with unit length and processing duration. We use a two-dimensional index (i, j) for each of these jobs, where i refers to the spatial position and j the time position. For each job (i, j) , we set its release time $r_{ij}^k = a_k + j - 1$ and due time $d_{ij}^k = d_k - (p_k^i - j)$. Parameters $\alpha_{ij}^k = \frac{1}{\sum p_k}$ and $\beta_{ij}^k = \frac{f_k}{\sum p_k}$ are assigned to each job (i, j) defined for vessel k , and will be used as objective function weights. The objective is to assign each job to a feasible machine, minimizing the sum of total weighted completion time and total weighted tardiness, given by $\sum_{k,i,j} \alpha_{ij}^k + \sum_{k,i,j} \beta_{ij}^k (c_{ij}^k - d_{ij}^k)^+$, where the completion time c_{ij}^k of job (i, j) for vessel k is $t_{ij}^k + 1$ when it is processed in time period t_{ij}^k .

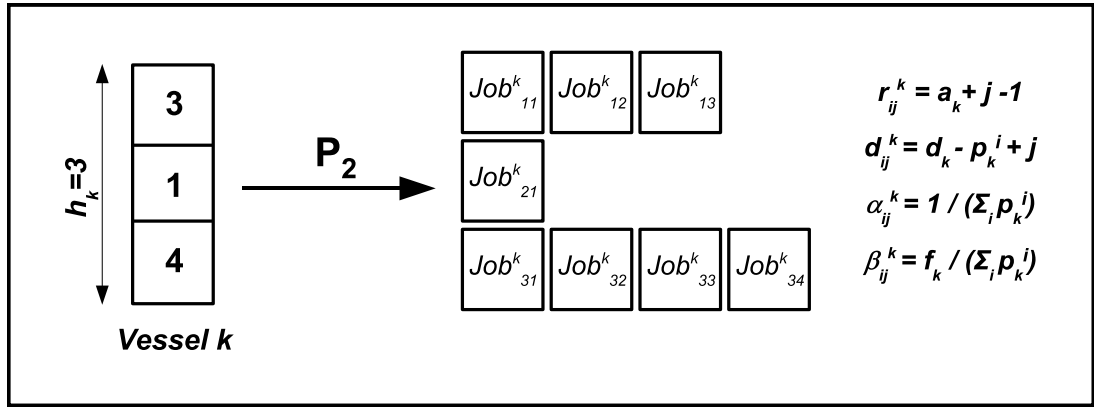


Figure 16: Construction of lower bounding scheduling problem P_2 for a given dynamic BQCSP instance

Theorem 9 $LB_2 = \lceil C_{P_2}^* + \Phi - \Psi \rceil$ is a lower bound for the BQCSP where $C_{P_2}^*$ is the optimal objective function value of P_2 constructed for the BQCSP, $\Phi = \frac{\sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^i-1} j}{\sum_{i=1}^{h_k} p_k^i}$, and $\Psi = \sum_{k=1}^n a_k$.

Proof. Any feasible solution of the BQCSP generates a feasible solution for the corresponding P_2 . In such a solution, for each job (i, j) defined for vessel k , $c_{ij}^k \leq c_k - p_k^i + j$ holds. Then, $c_{ij}^k - d_{ij}^k \leq c_k - p_k^i + j - (d_k - p_k^i + j) = c_k - d_k$. Hence,

$$\sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^i} \alpha_{ij}^k c_{ij}^k \leq \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^i} \frac{1}{\sum_{i=1}^{h_k} p_k^i} (c_k - p_k^i + j) = \sum_{k=1}^n \left(c_k - \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^i} \frac{p_k^i - j}{\sum_{i=1}^{h_k} p_k^i} \right) \quad (132)$$

and

$$\sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^i} \beta_{ij}^k (c_{ij}^k - d_{ij}^k)^+ = \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^i} \frac{f_k}{\sum_{i=1}^{h_k} p_k^i} (c_k - d_k)^+ = \sum_{k=1}^n f_k (c_k - d_k)^+ \quad , \quad (133)$$

which implies

$$C_{P_2} = \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^i} \alpha_{ij}^k c_{ij}^k + \sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^i} \beta_{ij}^k (c_{ij}^k - d_{ij}^k)^+ = \sum_{k=1}^n c_k - \frac{\sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^i-1} j}{\sum_{i=1}^{h_k} p_k^i} + \sum_{k=1}^n f_k (c_k - d_k)^+. \quad (134)$$

Since $C_{P_2}^* \leq C_{P_2}$,

$$C_{P_2}^* + \frac{\sum_{k=1}^n \sum_{i=1}^{h_k} \sum_{j=1}^{p_k^i-1} j}{\sum_{i=1}^{h_k} p_k^i} - \sum_{k=1}^n a_k \leq \sum_{k=1}^n (c_k - a_k) + \sum_{k=1}^n f_k (c_k - d_k)^+. \quad (135)$$

With all parameters integer, the right hand side of Expression 135, which is the objective function of the BQCSP, is integer. Hence, $LB_2 = \lceil C_{P_2}^* + \Phi - \Psi \rceil$ is a lower bound for the BQCSP. \square

Theorem 10 P_2 can be solved in polynomial time.

Proof. P_2 can be modeled as a minimum weight matching problem on a bipartite graph constructed similarly as in the proof of Theorem 3 in Chapter 2. \square

Consequently, the lower bound $LB = \max\{LB_1, LB_2\}$ is a polynomially computable lower bound for the BQCSP, and by Theorem 8, it is tighter than the lower bound obtained by the LP relaxation of the MIP presented.

5.4 Solution Method: A Two Phase Tabu Search Heuristic

In order to obtain a good solution, the decision maker should consider allocating two different resources simultaneously: berth area and quay cranes. Furthermore, an inferior solution can be improved by changing the berthing position of a vessel, or modifying the crane allocation plan, or doing both. For example, in Figure 17 schedule *A* can be improved by changing the berth location of vessel 5 as in schedule *B*. Further improvements can be observed by changing the crane schedule using the berth allocation plan of *B*. Schedule *C* represents the case where work is started on hold 1 of vessel 1 before hold 4 of vessel 4. This shifts the crane idleness in time period 4 in schedule *B* to time period 8 in schedule *C*. In schedule *D*, we start work on hold 2 of vessel 5 before hold 1 of the same vessel. This makes it feasible to start work on vessel 3 on time period earlier.

In the following sections, we provide search methods that combine the ideas in the tabu search algorithms proposed for the BAP and the QCSP in Chapter 2 and Chapter 4. We first show how these search procedures can be combined in a straightforward manner using the idea of nested neighborhoods. Then, we discuss the computational limitations to this approach, and present how the solution procedure can be decomposed into two phases for a more efficient and effective search.

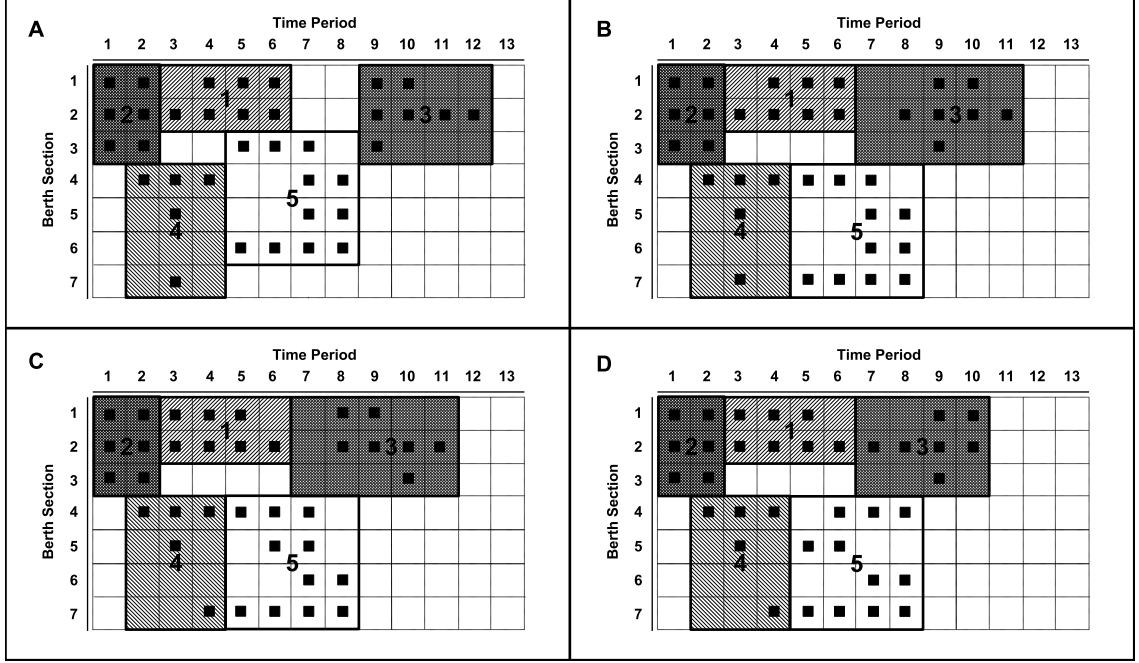


Figure 17: An illustration of improving a given solution by changing berth schedule and quay crane schedule

5.4.1 Encoding and Decoding a Solution

We associate a feasible solution of the BQCSP with three attributes; a berth position vector \mathcal{B} , a vessel list \mathcal{L}_v , and a global hold list \mathcal{L}_h . Among these, the pair $(\mathcal{B}, \mathcal{L}_h)$ is sufficient to define a unique berth and quay crane schedule x as follows. Begin with the first unprocessed hold (k, i) in \mathcal{L}_h where k is the vessel index and i is the hold index within the vessel. If this is the first hold of vessel k considered, we find the first time period $t_k \geq a_k$ where berth sections $\{b_k, \dots, b_k + h_k - 1\}$ are not occupied, and mark them occupied by vessel k for time periods $\{t_k, \dots, T\}$. Then, we find the first time period t_k^i where at least one crane is available in time periods $\{t_k^i, \dots, t_k^i + p_k^i - 1\}$. We reduce the number of available cranes at those time periods by 1, and mark hold i of vessel k as processed. If this is the last hold of vessel k considered, we set $c_k = \max_i \{t_k^i + p_k^i\}$ and mark berth sections unoccupied for time periods $\{c_k, \dots, T\}$. We continue until all holds in \mathcal{L}_h are processed. With appropriate data structures,

this decoding operates in $O(\hat{n})$ time where \hat{n} is the total number of holds. Recall the discussion in Chapter 4 that describes how this encoding/decoding approach may not always identify a feasible solution. If there exist two vessels occupying a number of overlapping berth sections, in order to have a feasible decoding all holds of the first vessel should be listed before all holds of the second vessel in \mathcal{L}_h .

This limitation of the hold list decoding becomes more important for the BQCSP, because the vessel priority list (which vessels should be berthed before which others) has not been predetermined like the case for the QCSP. Note that the hold list limitation implies that if simple swap or insertion moves are used to generate neighbor solutions by changing \mathcal{L}_h , it is impossible to change the relative positions of vessels sharing overlapping berth sections without encountering infeasible hold lists. For example, changing the relative positions of vessel 1 and vessel 2 in the berth schedules given in Figure 17 by modifying the corresponding \mathcal{L}_h requires repositioning of all holds of vessel 1 before all holds of vessel 2 in \mathcal{L}_h . This is only possible if either multiple hold swaps or reinsertions are performed in an iteration, or infeasible solutions are allowed within the search. Therefore, we associate each solution with a vessel list \mathcal{L}_v where the order of the list determines berthing priority. As will be described later, this list will help us to change the positions of blocks of holds in \mathcal{L}_h , and therefore it will create a simple mechanism for changing the relative positions of vessel rectangles on the time-space diagram.

Parallel to the discussions provided in Chapter 2 and Chapter 4, for a given berth and quay crane schedule x , there always exists an encoding where the berth schedule component is encoded by \mathcal{B} and \mathcal{L}_v , and the quay crane schedule component is encoded by \mathcal{L}_h . As mentioned earlier, while a given schedule uniquely defines a set of berth positions \mathcal{B} , it may correspond to many different \mathcal{L}_v . Similarly, a given quay crane schedule may correspond to multiple \mathcal{L}_h . Therefore, we again employ a unique *primal vessel list* $\mathcal{L}_v^P(x)$, and a unique *primal hold list* $\mathcal{L}_h^P(x)$ when encoding

a berth and quay crane schedule x to utilize their benefits described in Chapter 2 and Chapter 4 within the search algorithms designed. Before presenting the details of these algorithms, we first describe how an initial solution is constructed.

5.4.2 Initial Solution

We generate an initial feasible solution x^0 using only a vessel priority list \mathcal{L}_v by constructing a berth position vector using a two-dimensional first-fit (FF) rule and by constructing a hold processing order by utilizing the maximum-hold-first (MHF) rule. The FF rule places vessel rectangles k on the time space diagram one by one in the order specified by \mathcal{L}_v at the earliest possible berthing time (i.e., the smallest time $t \geq a_k$ with h_k available contiguous berth sections), and the lowest index berth section at that time. For each vessel k placed, the MHF rule allocates cranes to holds in the decreasing order of the hold processing time, and for the hold i to which a crane is next assigned it finds the earliest time period t_k^i where at least one crane is available in the next p_k^i time periods.

Thus, initial solutions can be generated using any method for creating a sorted \mathcal{L}_v . In our approach, we use \mathcal{L}_v^{FCFS} , \mathcal{L}_v^{EDD} , and \mathcal{L}_v^{mEDD} which respectively denote the priority lists created by first-come first-served, earliest due date, and modified earliest due-date (minimum d_k/f_k first) rules.

The initial solution is chosen to be the one of the above with the smallest objective function value. This simple approach for building an initial solution mimics decision rules-of-thumb that a terminal operator might use to generate berth schedules.

5.4.3 Single Phase Search

The single phase search algorithm developed in this section integrates the tabu search algorithm presented for the QCSP in Chapter 4 as a third layer to the nested tabu search algorithm provided for the BAP in Chapter 2. This integration is illustrated in Figure 18 where the QCSP search is referred to as TS_3 . The resulting layered search

approach modifies the berth schedule in the first two layers, and tries to find a good crane schedule in the innermost layer for each berth schedule visited.

The integration of TS_3 to TS_2 is very similar to the integration of TS_2 to TS_1 described in Chapter 2. In the previous one, we initiate a search on berthing positions of vessels, which changes \mathcal{B} , for each qualified move that modifies \mathcal{L}_v in the first layer search. The best berthing position vector \mathcal{B} found by the inner search defines the cost of the corresponding first layer move. In such a nested search algorithm, each move considered in the second layer results in a berth schedule. Using this, for each berth schedule visited in the second layer, in order to find a good crane schedule, we initiate TS_3 whose best solution will define the cost of the corresponding second layer. Such a three layer nested search structure enables us to evaluate different combinations of berth scheduling and crane scheduling decisions systematically.

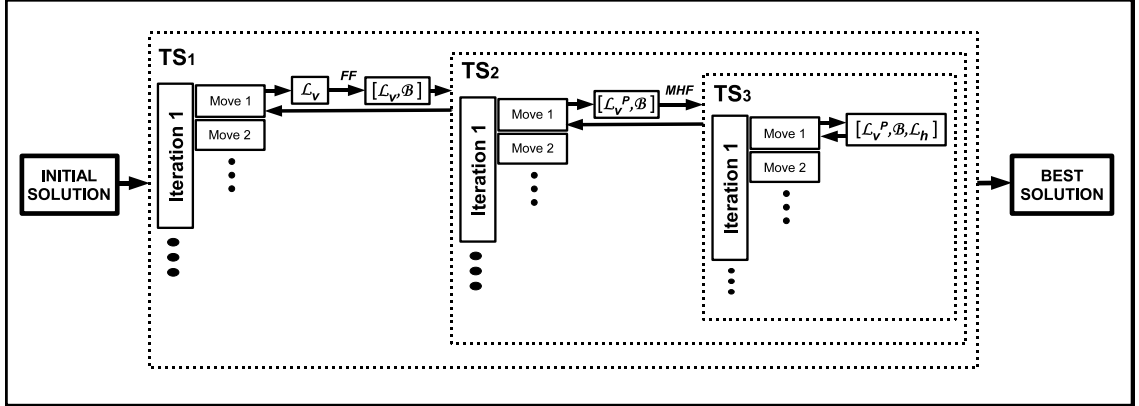


Figure 18: Schematic representation of the single phase search approach for the BQCSP

At the beginning of each TS_3 run initiated for the corresponding berth schedule component of the partial solution x , we generate the initial \mathcal{L}_h by sorting holds of vessels in the order of primal vessel list $\mathcal{L}_v(x)$ using *MHF* rule. We then decode the resulting \mathcal{L}_h into the initial quay crane schedule for the given berth schedule, and follow the steps of the TS_3 provided in Chapter 4. At the end of each TS_3 run, the best quay crane schedule found is combined with the corresponding berth schedule

to yield the complete berth and quay crane schedule x .

We provide the step-by-step detail for the proposed search algorithm below. Let x^0 , y^0 , and z^0 denote the initial solution; x , y , and z be the current solution; and x^* , y^* , and z^* represent the best solutions found by TS_1 , TS_2 and TS_3 respectively. Let $C(x)$ represent the objective function value for solution x . Furthermore, p_1 , r_1 , and q_1 are neighborhood parameters for TS_1 , and p_3 , r_3 , and q_3 are neighborhood parameters for TS_3 .

5.4.3.1 Steps of $TS_1(x^0, x^*)$:

Step 1: Initialization. Set p_1 , r_1 , and q_1 . Let x^0 be the lowest-cost solution found from among the three initial vessel lists, \mathcal{L}_v^{FCFS} , \mathcal{L}_v^{EDD} , and \mathcal{L}_v^{mEDD} , each with a berth position vector determined by the FF rule and a crane schedule determined by MHF rule. Let \mathcal{L}_v , \mathcal{L}_h and \mathcal{B} be the vessel list, the hold list and the berth position vector associated with x^0 . Set $x = x^* = x^0$. Initialize counters $t_1^T = 0$ and $t_1^S = 0$, which denote the total number of iterations, and the number of successive iterations with no improvement in the best objective function value respectively. Set the value of t_1^M , which is the maximum number of iterations allowed with no improvement.

Step 2: Neighborhood Search. Set $t_1^T = t_1^T + 1$. For each candidate list $\mathcal{L}_v^i \in \mathcal{N}_{\mathcal{L}}(p_1, r_1, q_1, \mathcal{L}_v)$, construct an associated \mathcal{B}_i by the FF rule and L_h^i by the MHF rule, and let x_i be the associated solution. For each solution x_i such that $M(x_i)$ is not in the tabu list and has not yet been considered during this step, run $TS_2(x_i, x_i^*)$. Let a be the move i with the smallest value of $C(x_i^*)$.

Step 3: Solution Updates. Set $x = x_a^*$, and let $\mathcal{L}_v = \mathcal{L}_v^a$. If $C(x) < C(x^*)$, set $x^* = x$ and $t_1^S = 0$. Otherwise, set $t_1^S = t_1^S + 1$. Update the tabu list, adding $M(x)$ for the appropriate random number of iterations and removing any marks whose duration has expired. If $t_1^S < t_1^M$, go to Step 2. Else, stop. The best solution

found is x^* .

5.4.3.2 Steps of $TS_2(y^0, y^*)$:

Step 1: Initialization. Set the value of s . Set $y = y^* = y^0$, and let \mathcal{B} be the berth position vector for solution y . Initialize counters $t_2^T = 0$ and $t_2^S = 0$, which denote the total number of iterations, and the number of successive iterations with no improvement the best objective function value respectively. Set the value of t_2^M , which is the maximum number of iterations allowed with no improvement.

Step 2: Neighborhood Search. Set $t_2^T = t_2^T + 1$. Construct set V_S for y . Consider each move y_j decoded from non-tabu berth position vector $\mathcal{B}_j \in \mathcal{N}_{\mathcal{B}}(s, \mathcal{V}_S, \mathcal{B})$ combined with primal list $\mathcal{L}_v^P(y)$ and the yielding MHF rule primal hold list $\mathcal{L}_h^P(y)$, and run $TS_2(y_j, y_j^*)$. Let b be the move j with the smallest value of $C(y_j^*)$.

Step 3: Solution Update. Set $y = y_b^*$, and let $\mathcal{B} = \mathcal{B}_b$. If $C(y) < C(y^*)$, set $y^* = y$ and $t_2^S = 0$. Otherwise, set $t_2^S = t_2^S + 1$. Update the tabu list by inserting the reverse berth position move associated with solution y_b into the tabu list for the appropriate random number of iterations, and removing reverse moves whose duration has expired. If $t_2^S < t_2^M$, go to Step 2. Else, stop. The best solution found is y^* .

5.4.3.3 Steps of $TS_3(z^0, z^*)$:

Step 1: Initialization. Set $s = 1$. Compute Π_k^s for each vessel k . Set p_3 , r_3 , and q_3 . Let z^0 be the initial solution found with the primal hold list $\mathcal{L}_h^P(z^0)$. Let $\mathcal{L}_h = \mathcal{L}_h^P(z^0)$, $z = z^* = z^0$. Initialize the counter $t_3^T = 0$ which denotes the total number of iterations at current stage. Set the value of t_3^M , which is the maximum number of iterations allowed at each stage.

Step 2: Neighborhood Search. Set $t_3^T = t_3^T + 1$. For each candidate list $\mathcal{L}_h^l \in$

$\mathcal{N}_{\mathcal{L}}^s(p, r, q, \mathcal{L}_h)$, first check feasibility. If \mathcal{L}_l is feasible, decode it into solution z_l . Construct a list \mathcal{U} by ranking the feasible moves in nondecreasing order of C . Select the first move c in \mathcal{U} which is not tabu or $C(z_c) < C(z^*)$.

Step 3: Solution Update. Set $z = z_c$ and let $\mathcal{L}_h = \mathcal{L}_h^c$. If $C(z) < C(z^*)$, set $z^* = z$. Update the tabu list by inserting the reverse swap move associated with solution z_c into the tabu list for the appropriate random number of iterations, and removing reverse moves whose duration has expired. If $t_3^T < t_3^M$, update $t_3^T = t_3^T + 1$ and go to Step 2. Else if $s < n$, set $t_3^T = 0$, update $s = s + 1$, recompute Π_k^s for each vessel k , set $z = z^*$ and $\mathcal{L}_h = \mathcal{L}_h^P(z^*)$, and go to Step 2. Else, stop. The best solution found is z^* .

5.4.4 Two Phase Search

The three layer neighborhood structure employed by the single phase search approach is expected to result in high computation times if we desire to search a somewhat large portion of the solution space. In this section, we describe how that algorithm can be decomposed into two phases for a more efficient and effective search.

Note that in the single phase algorithm, the first two layers look for possible good berth schedules, and the third layer is utilized to find a good feasible quay crane schedule for each berth schedule visited. In the two phase search approach described in this section, we separate the third layer from the first two layers. In the first phase, we run TS_1 and TS_2 just like in the BAP case. However, instead of running TS_3 for each berth schedule x visited, we first relax the crane capacity constraint by setting $Q = B$, and hence find a lower bound solution with objective function value $LB(x)$. Since $Q = B$ means that we have a quay crane ready for each hold immediately after mooring, such a lower bound schedule can be constructed by decoding the vector pair $(\mathcal{B}, \mathcal{L}_v)$ into a solution as described in Chapter 2 assuming $p_k = p_k^{max}$ for vessel k . Furthermore, for each berth schedule x visited in the first phase, we can quickly

construct a feasible solution using the *MHF* rule having an objective function value of $UB(x)$. $LB(x)$ gives an idea of how good the vessel rectangles are packed, and $UB(x)$ provides us a prediction of the impact of crane allocation. Using $LB(x)$ and $UB(x)$, we calculate a score $S(x)$ for berth schedule x as

$$S(x) = \lambda LB(x) + (1 - \lambda)UB(x) \quad (136)$$

where λ is a constant set by the decision maker and has a value between 0 and 1. During the first phase, we store a set \mathcal{W} of best different berth schedules, where $w = |\mathcal{W}|$ is defined by the decision maker and may depend on the size of the instance. We say that two berth schedules x_a and x_b are same if they have the same mark M determined by an arbitrary function $g(\{b_k\}, \{t_k\})$.

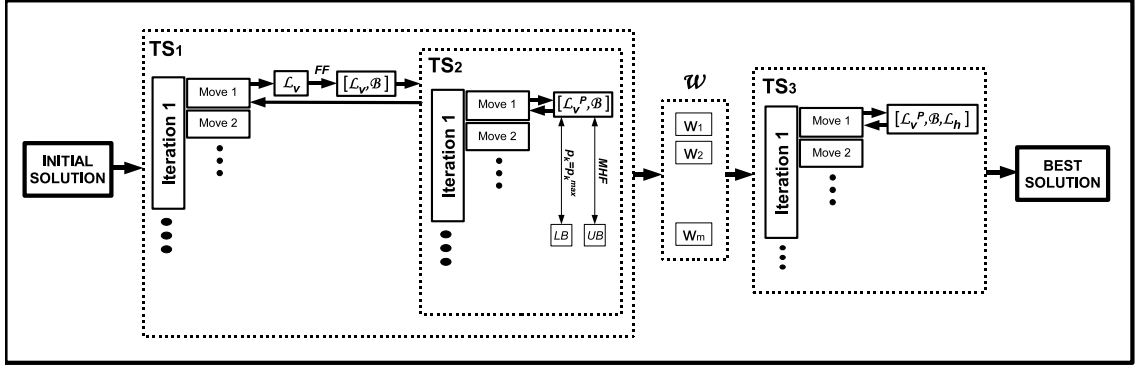


Figure 19: Schematic representation of the two phase search approach for the BQCSP

In the second phase, for each $x_j \in \mathcal{W}$, we execute TS_3 to find a near optimal quay crane schedule. In the end, we select the best resulting berth and quay crane schedule x^* found after the w TS_3 runs performed in the second phase. Figure 19 illustrates the overall procedure. Since TS_3 is called only w times for the promising berth schedules, such a two phase search procedure enables us to search a larger and more promising portion of the solution space.

5.4.5 Additional Remarks

The lower bound $LB(x)$ computed in the two phase search can also be utilized in the single phase search to increase run time efficiency. At any TS_2 iteration, after move construction, we compute $LB(y_j)$ for each move j as described in the previous section. We then sort them in the nondecreasing order of $LB(y_j)$, and start running TS_3 for these moves in that order. In this manner, before evaluating a move, we can first compare its lower bound with the cost of the best move evaluated so far. If the lower bound is not less than the cost of the best move found so far, we can skip this move since it will not be selected at this iteration. As a result, some of the moves yielding bad solutions can be detected and eliminated without running TS_3 . Similarly, in two phase search, for each candidate berth schedule x stored in \mathcal{W} , we first compare $LB(x)$ to the best solution found so far by running TS_3 for the previous berth schedules in \mathcal{W} . If the lower bound of the corresponding candidate is not smaller than the best solution found so far, we can skip that candidate.

We can also use $LB(x)$ to improve the run time of TS_3 . Note that the objective function value of the best solution that we can find for the corresponding berth schedule x cannot be less than $LB(x)$. Therefore, if we find such a solution, we can terminate TS_3 . For this reason, whenever the best solution z^* is updated, we compare it with the corresponding lower bound for early termination.

5.5 Computational Experiments

Computational experiments were conducted on test instances generated in Chapter 2 and used in Chapter 4 in order to evaluate the performance of the algorithms proposed.

Table 19: Single Phase Search and Two Phase Search parameters used in the computational experiments

Parameter	1PhS	2PhS
p_1	3	5
r_1	8	10
q_1	3	5
t_1^M	$5n$	$50n$
θ_1^{min}	$n/2$	$n/2$
θ_1^{max}	n	n
s	$\max\{2, V /3\}$	$\max\{2, V /3\}$
t_2^M	5	$n/2$
θ_2^{min}	2	2
θ_2^{max}	5	5
α	1	1
p_3	3	5
r_3	10	10
q_3	3	5
t_3^M	5	$\lceil 10 \frac{UB}{LB} \rceil$
w	-	$10n$

The parameter values for each layer of the search algorithms were determined by experimentation, and are presented in Table 19 for both the single phase search (1PhS) and the two phase search (2PhS) versions. Although the parameters used in the second layer are similar, there exist differences in the first and the third layer. A smaller neighborhood is considered and fewer iterations are allowed in the first and second layer searches for 1PhS compared to 2PhS.

Note also that, the number of iterations allowed at each stage of TS_3 is dependent on the gap between lower bound and the upper bound of the corresponding berth schedule transferred from the second layer. If this gap is larger, such a dynamic parameter setting lets TS_3 run longer.

The mixed integer program provided in Section 5.2 could not solve any of the

Table 20: Test results for small BQCSP instances

Instance	IS	LB_{LP}	LB_1	LB_2	1PhS	2PhS	1PhS Imp. (%)	2PhS Imp. (%)
10	111	45	67	65	98	98	30	30
11	74	41	49	54	64	63	50	55
12	235	46	110	114	159	159	63	63
13	137	44	53	70	92	87	67	75
14	95	40	45	53	66	64	69	74
15	113	46	61	66	83	83	64	64
16	79	40	47	52	70	70	33	33
17	209	51	108	118	154	154	60	60
18	239	49	89	107	160	154	60	64
19	170	45	69	82	130	130	45	45
20	219	54	87	77	160	160	45	45
21	257	50	103	105	173	173	55	55
22	191	53	85	112	148	142	54	62
23	266	54	84	90	130	130	77	77
24	152	47	66	71	87	87	80	80
25	232	55	84	118	156	155	67	68
26	222	58	93	109	146	146	67	67
27	340	56	107	98	164	164	76	76
28	208	57	91	134	162	162	62	62
29	200	52	82	113	159	160	47	46
30	337	60	112	124	168	168	79	79
31	490	61	194	262	307	301	80	83
32	297	61	95	101	145	141	78	80
33	506	61	190	257	313	305	78	81
34	371	62	115	139	192	192	77	77
35	451	52	150	192	245	231	80	85
36	302	59	101	113	154	154	78	78
37	240	60	93	101	133	132	77	78
38	731	65	234	269	391	392	74	73
39	609	65	253	273	357	351	75	77

instances considered to optimality. Therefore, we generated an additional set of 100 very small instances with $B = 7$, $Q = 4$ and $n = 5$. CPLEX was able to solve them in 10 minutes on average on an Intel 2.4 GHz Pentium processor running Linux with 2 GB memory. However, the maximum run time observed reached 2.5 hours for some instances. On the other hand, both 1PhS and 2PhS provided the optimal solution under 2 seconds for these instances.

Table 20 and Table 21 present the three polynomially-computable lower bounds, LB_{LP} , LB_1 and LB_2 . Theorem 8 proves that $\max\{LB_1, LB_2\} \geq LB_{LP}$, and the empirical results show that the lower bounds LB_1 and LB_2 introduced improves LB_{LP} substantially. This improvement is 73%, 95%, and 201% for small instances (1y, 2y,

Table 21: Test results for large BQCSP instances

Instance	IS	LB_{LP}	LB_1	LB_2	1PhS	2PhS	1PhS Imp. (%)	2PhS Imp. (%)
40	177	82	102	95	114	113	84	85
41	199	92	108	96	126	126	80	80
42	287	87	112	156	196	188	69	76
43	171	87	102	101	122	122	71	71
44	98	70	71	61	77	77	78	78
45	194	91	97	86	111	111	86	86
46	307	90	106	118	160	152	78	82
47	248	93	106	102	140	140	76	76
48	116	89	90	75	96	96	77	77
49	177	86	93	86	113	112	76	77
50	515	119	200	225	283	264	80	87
51	153	93	95	88	119	114	59	67
52	545	108	175	245	311	305	78	80
53	435	116	156	202	256	244	77	82
54	651	114	280	311	435	420	64	68
55	302	114	145	142	179	179	78	78
56	362	116	135	170	201	197	84	86
57	102	78	80	81	92	91	48	52
58	527	116	148	212	260	253	85	87
59	409	115	171	188	252	245	71	74
60	562	123	256	246	370	359	63	66
61	1102	134	377	508	605	587	84	87
62	1535	149	715	712	965	927	70	74
63	845	139	288	312	467	449	71	74
64	542	126	198	188	252	240	84	88
65	766	126	245	212	427	373	65	75
66	851	145	278	381	474	466	80	82
67	668	144	260	304	424	428	67	66
68	305	128	146	150	198	190	69	74
69	384	123	161	181	262	242	60	70

and 3y), and 20%, 69%, and 138% for large instances (4y, 5y, and 6y) respectively.

The objective function values of the best solutions found by 1PhS and 2PhS are also presented in Table 20 and Table 21. Note that for 16 of the small instances in Table 20, both approaches found solutions with the same cost. For the remaining, 2PhS found better solutions in 12 cases and 1PhS performed better in only 2 instances. For large instances, 2PhS substantially outperformed 1PhS by finding better solutions in 22 instances. In 7 of the remaining instances they found solutions with the same objective function, and in only one case 1PhS found a slightly better solution.

The last two columns in Table 20 and Table 21 measure the percentage improvement (reduction) in the optimality gap computed between the best lower bound and

the initial solution versus the gap computed for the best solutions by the 1PhS and the 2PhS respectively. The average gap improvement is 53%, 63% and 78% by the 1PhS, and 56%, 64% and 79% by the 2PhS for instances 1y, 2y, and 3y. For large instances 4y, 5y and 5y, these figures are 77%, 72% and 71% for 1PhS, and 79%, 76% and 76% for 2PhS respectively. This indicates that the performance of the search algorithms proposed do not appear to diminish as problem size increases.

Table 22: Average computation times observed (in sec.) for test BQCSP instances

Set	1PhS	2PhS	2PhS (P1)	2PhS (P2)
1	57	41	11	30
2	99	60	21	39
3	257	104	32	72
4	2810	241	133	108
5	5059	1234	540	694
6	10408	1817	916	901

Table 22 presents the computational times in seconds observed during tests. As one can see clearly, two stage search method reduces the computational time while providing better solutions. The last two columns in the table presents the run time used by each phase of the 2PhS. With the parameters used, almost equal time was spent on searching for promising berth schedules and good quay crane schedules.

Lastly, Table 23 exposes the benefit of simultaneous berth and quay crane scheduling. The columns SEQ and SIM respectively present the objective function values of the best solutions found by the sequential and the simultaneous berth and quay crane scheduling. The values in SEQ column are the best solutions found for the QCSP discussed in Chapter 4. Recall that in that chapter, we first solved the BAP and used its solution to solve the QCSP. On the average we observe 6.9% and 7.3% reduction in cost for small and large instances respectively. This saving reaches almost 28% for some instances.

Table 23: Comparison of sequential and simultaneous berth and quay crane planning

Instance	SEQ	SIM	Diff.(%)	Instance	SEQ	SIM	Diff.(%)
10	106	98	8.2	40	118	113	4.4
11	65	63	3.2	41	128	126	1.6
12	159	159	0.0	42	236	188	25.5
13	100	87	14.9	43	125	122	2.5
14	67	64	4.7	44	78	77	1.3
15	83	83	0.0	45	111	111	0.0
16	86	70	22.9	46	171	152	12.5
17	160	154	3.9	47	144	140	2.9
18	161	154	4.5	48	96	96	0.0
19	135	130	3.8	49	112	112	0.0
20	164	160	2.5	50	286	264	8.3
21	182	173	5.2	51	124	114	8.8
22	158	142	11.3	52	344	305	12.8
23	134	130	3.1	53	274	244	12.3
24	96	87	10.3	54	425	420	1.2
25	184	155	18.7	55	180	179	0.6
26	154	146	5.5	56	221	197	12.2
27	164	164	0.0	57	94	91	3.3
28	175	162	8.0	58	300	253	18.6
29	171	159	7.5	59	243	245	-0.8
30	195	168	16.1	60	364	359	1.4
31	333	301	10.6	61	653	587	11.2
32	145	141	2.8	62	1039	927	12.1
33	323	305	5.9	63	476	449	6.0
34	196	192	2.1	64	240	240	0.0
35	254	231	10.0	65	383	373	2.7
36	161	154	4.5	66	535	466	14.8
37	134	132	1.5	67	465	428	8.6
38	441	391	12.8	68	203	190	6.8
39	356	351	1.4	69	309	242	27.7
<i>Avg.</i>			<i>6.9</i>	<i>Avg.</i>			<i>7.3</i>

CHAPTER VI

THE VOYAGE AND BERTH SCHEDULING PROBLEM

6.1 Introduction

Ocean shipping is the major transportation mode of international trade, and it is likely to be the predominant method of transporting large volumes at very low costs between continents in the near future. Parallel to increases in world trade and ocean shipping demand, the world vessel fleet has experienced enormous growth in the last few decades. There exist three general modes of operation in ocean shipping: industrial, tramp, and liner [36]. In industrial shipping, the cargo owner or shipper also controls the ships and manages this fleet to serve its cargo demand at minimal cost. In tramp shipping, a carrier has a certain amount of contract cargo that it is committed to carry between specified ports within a specific time frame, and it attempts to maximize the profit earned from both contracted and spot cargo solicited when excess capacity is available. Liner shipping carriers operate according to published itineraries and schedules available to shippers. Although the way they operate may differ, all shipping companies have one main objective in common: to utilize their fleets optimally.

In this chapter, we focus on a tactical level problem in liner shipping, which accounts for more than half of the total cargo volume shipped. Liner shipping differs significantly from industrial and tramp services especially when it comes to routing and scheduling issues. The differences result primarily from the fact that liner services operate on fixed schedules that are determined by vessels sailing circuits of ports. The number of ships operating on a given liner service route, the distance traveled by a ship on the route, and the speed of the ships all determine the frequency of service

provided on the route, and thus also implies a certain level of customer service between port pairs.

A liner carrier faces important decisions at different planning levels. At the strategic level, the carrier must determine the fleet size and mix. Because a ship is a very large capital investment, this strategic decision is extremely important. Then, a service network is designed which includes both terminal contracts and the determination of service routes. Given a service network, the carrier makes tactical decisions such as the assignment of ships to routes, and scheduling routes at each port visited; these decisions then allow the carrier to publish itineraries. Cargo booking which is the decision of accepting or rejecting cargo and routing which is the decision of selecting paths for booked cargo to connect origin and destination points are key operational level decisions.

In this study, we focus on the problem of scheduling liner carrier routes (or vessel voyages) considering berthing limitations at container terminals. A route defines a sequence of ports to visit for vessels assigned to the route. Many researchers have studied the problem of designing vessel routes. A comprehensive literature review is provided in [64], [65] and [12] on the research work conducted in the last three decades. In [61], a nonlinear integer program and a lagrangean relaxation method is provided to maximize the total profit by finding the optimal sequence of ports to visit for each ship as well as the optimal amount of cargo to be transported between each pair of ports by each ship. More recently, the study presented in [3] integrates tactical level ship scheduling and an operational level cargo routing problem. A mixed integer program is formulated and algorithms are developed to solve realistic instances. Constraints enforce weekly frequency requirements and handle transshipments, which require overlapping of vessel schedules at some ports to smoothly transfer containers between routes. A heterogeneous fleet with ships of different sizes, cost structures and speeds is considered.

The problem we study, on the other hand, uses a set of ship routes as input and schedules vessel voyages on these routes considering berth capacities and terminal business. At each port visited, ships dock at berths of container terminals for loading and unloading. Many of these terminals, including large ports in Europe, China, Singapore, Taiwan, are multi-user terminals. The terminal operator decides when and where to berth each arriving vessel. Thus, arriving vessels are not always processed immediately upon their arrival. The actual berthing time of a vessel depends on the ship traffic and arrangements made with operators of other vessels by the terminal operator. On the other hand, many terminals in the US, Japan, and Korea are dedicated terminals. Berths within dedicated terminals are leased by ocean carriers. Because of high leasing costs, only large ocean carriers and their alliance partners operate with dedicated berths. Under such arrangements, only vessels employed by the lessee carrier are processed at the berth, and as a result dedicated terminals have lower berth utilizations than multi-user terminals. At dedicated terminals, however, the terminal operator and the carrier practically operate as the same party and thus the carrier may decide on mooring schedules of all vessels visiting the terminal.

Large ocean carriers have many vessels operating on different routes which visit both busy multi-user terminals but also intersect at some dedicated terminals. In this study, we aim to solve the problem of determining when to visit each port for vessels operating on each predetermined route to minimize total operating cost while respecting capacity limitations at dedicated terminals and avoiding congested time periods at multi-user terminals. We name the resulting problem the voyage and berth scheduling problem (VBSP). The itinerary of a liner route has two components; ports to visit, and the times to visit these ports. Hence, the VBSP tries to determine the latter component of liner shipping itineraries optimally for a given set of routes.

6.2 Problem Formulation

In the VBSP, we consider a large ocean carrier operating on a set R of predetermined vessel routes. We assume that vessels operating on the same route are similar in size, fuel consumption, etc. The set of ports visited by vessels traversing route r is denoted by $N_r \in N$. $N = N^D \cup N^M$ represents the set of all ports served by the carrier where N^D is the set of ports with dedicated terminals, and N^M is the set of ports with multi-user terminals. R_i represents the subset of routes that visit port i . For each route r , $pred_r(i)$ and $succ_r(i)$ respectively denote the ports visited before and after port $i \in N_r$ within the route, and d_{ij} represents the sailing distance between port i and port j . We assume that dedicated terminals have discrete berthing areas such that only one vessel can be processed at a berth, and all cranes on the berth can be allocated to the vessel receiving service at the berth. Hence, the capacity, B_i , of a dedicated terminal (at port i) is expressed by the number of berths leased by the carrier. We assume that the container handling time required by ships of route r at terminal i is pre-computed using demand estimates and denoted by p_i^r . The pre-computed processing time should almost accurately represent the actual dwell time in dedicated terminals since we know the number of quay cranes available for each ship. However, in multi-user terminals the processing time experienced by a vessel may be affected by the traffic at the terminal at the time of arrival.

Most large ocean carriers provide regular weekly service at each port in their networks for each route. Thus, we assume that the service frequency is equal to one per week for all routes, implying that a vessel will visit each port in a route on the same day every week. Note that the number of vessels operating on a route depends on the total duration of the route. For example, a route providing service between ports on the US East Coast and Europe with a total cycle time of 4 weeks, would require 4 vessels to provide weekly service.

6.2.1 Cost Structure

In this section we discuss costs observed while operating liner shipping routes. While traversing a route, a vessel can be in three states; it can be sailing, it can be waiting to get moored, or it can be receiving service at a terminal. Different cost structures are associated with each of these three states. Furthermore, the number of ships required to provide weekly service varies by the total route duration. Therefore, we also assume that each ship assigned to a route requires a fixed cost.

6.2.1.1 Sailing Cost

The two components of sailing cost are the cost of the crew and the fuel consumption. Crew cost can be approximated by an hourly rate paid for the aggregate crew. Therefore, the crew cost function resembles the one depicted in Figure 20.

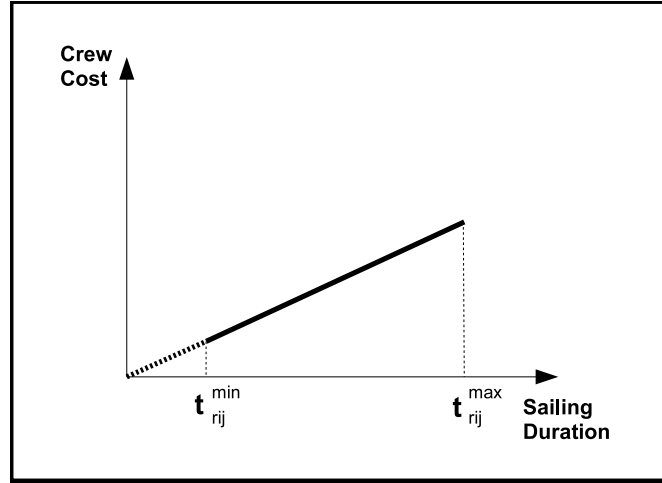


Figure 20: Illustration of a typical vessel crew cost function used in VBSP

Fuel consumption, on the other hand, depends on the speed of the ship, and it is expected to be the major component of the operating expenses of a ship. For a general diesel-powered vessel, the fuel consumption of the main engines is directly related to the third power of the sailing speed [63]. Although the relationship between speed and fuel consumption is an empirical one, the third power of the speed is a good

approximation and supported by regression analysis as well as theoretical models [15]. A small size ship of 1500 TEUs which burns around 40 tons of fuel per day at nominal (maximum) speed may save 20 tons per day by reducing the speed by 20%, and at \$1250 per ton these savings amount to \$25000 per day. The largest container ship in use, Emma Maersk, with a capacity of 11000 TEUs burns 310-320 tons of fuel per day at nominal speed. Using the following notation,

d : leg distance,

ν_0 : nominal speed,

ν : sailing speed,

t_0 : minimum possible sailing duration (d/ν_0),

t : sailing duration (d/ν),

f_0 : daily fuel consumption at nominal speed,

f : daily fuel consumption at sailing speed,

F_0 : total fuel consumption at nominal speed,

F : total fuel consumption at sailing speed,

[63] provides the following expression:

$$\frac{f}{f_0} = \left(\frac{\nu}{\nu_0}\right)^3 = \left(\frac{t_0}{t}\right)^3.$$

Since $F_0 = f_0 t_0$ and $F = f t$, the following expression relates total fuel consumption to sailing time:

$$F = F_0 \left(\frac{t_0}{t}\right)^2.$$

Therefore, the fuel cost function is expected to resemble the one provided in Figure 21. As depicted in the figure, the fuel cost function is defined within the range of minimum and maximum possible ship speeds, ν_{rij}^{min} and ν_{rij}^{max} , for a given leg from i to j of route r . ν_{rij}^{min} is set in order to satisfy a desired maximum travel time along the leg, such that the maximum travel time is acceptable from the perspective of customer service and will not lead to a decrease in shipping demand. On the other

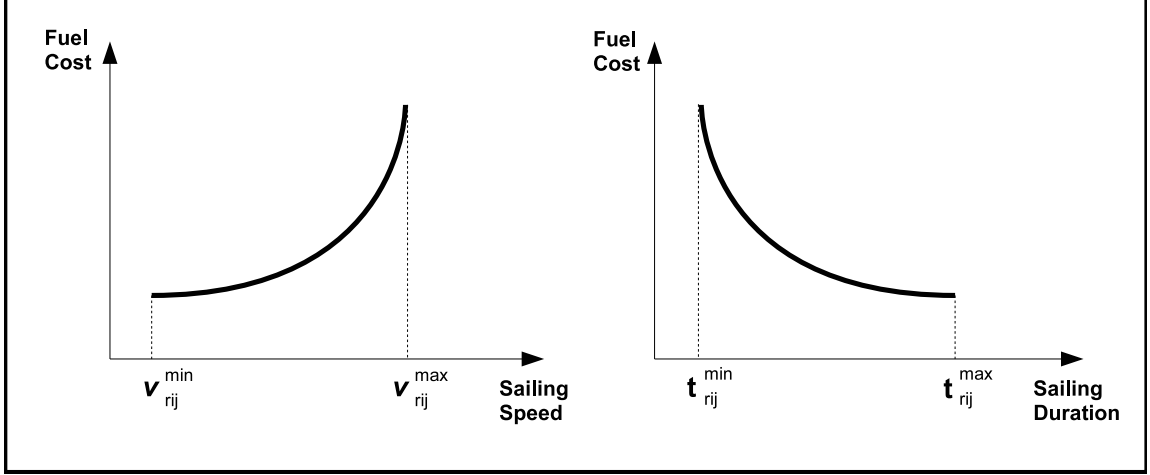


Figure 21: Illustration of a typical fuel consumption function used in the VBSP

hand, ν_{rij}^{max} is determined by considering the absolute vessel maximum speed and the fuel tank capacity of the ships used on route r , and the distance between port i and port j . If d_{ij} is short, ν_{rij}^{max} is set equal to the maximum speed. However, for long legs, ν_{rij}^{max} is set to be the fastest speed for the vessel to complete the leg without fuel replenishment. Then, minimum and maximum sailing times are computed as $t_{rij}^{min} = d_{ij}/\nu_{rij}^{max}$ and $t_{rij}^{max} = d_{ij}/\nu_{rij}^{min}$.

Fuel cost and crew cost functions may be different for each route and for each leg of the route.

6.2.1.2 Waiting Cost

While waiting at port to be berthed, no fuel is burned by main engines of the ship; however, a small amount of fuel is used for auxiliary systems and the cost of the crew is incurred. Therefore, waiting cost is modeled as a linear function of time.

6.2.1.3 Terminal Cost

The cost incurred at container terminals depends on whether the terminal is dedicated or multi-user. However, in both cases the total cost is related partly to the number of containers handled. Ocean carriers usually enter into long term leases for dedicated

terminals. Lease costs may include fixed charges for handling containers, or there may exist additional per visit charges related to the volume of cargo handled. We assume that these handling costs are not dependent on the day/time of the week the terminal is visited. This assumption is reasonable since the time required to load and unload containers can be predicted accurately at dedicated terminals. Thus, the cost of docking at such terminals can be omitted in our optimization problem.

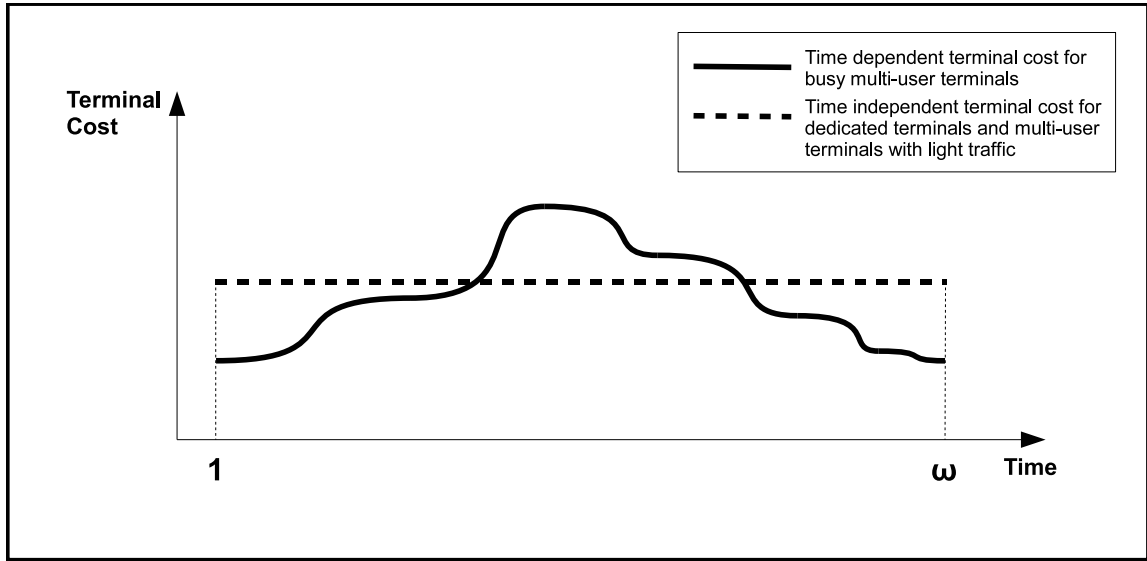


Figure 22: Illustration of typical terminal cost functions used in the VBSP

In busy multi-user terminals, on the other hand, it is possible that the cost of berthing will be related to the time of the visit. In such terminals, there exist heavy and light traffic periods throughout the week. In heavy traffic periods, ship processing may be delayed because of berth and quay crane limitations along with the scarcity of other resources. In such time periods, terminal operators may charge more for each container handled. The expected terminal cost functions are illustrated by Figure 22, and we assume that they may vary for each route visiting the same terminal.

6.2.1.4 Ship Cost

The number of ships required to operate a route with the desired service frequency is related to the total duration of the route. Naturally, there exists a desired number of ships for each route; however, traversing a route in a shorter or longer amount of time with fewer or more ships may result in better overall operating cost. In our model we define the number of ships operating on each route as a decision variable. The marginal cost of assigning an additional ship to a route can be constant or may increase as we add more ships to the route.

6.2.2 Modeling with Network Flow

In this section, we provide a mathematical formulation based on network flow which can be used to solve the VBSP. This formulation enables us to model the nonlinear sailing cost by discretizing time. We construct a time expanded network represented by a directed graph $G = (V, A)$. V is constructed by defining ω vertices for each port that the ocean carrier provides service. ω represents the number of time periods in one week. Thus, each time period is equal to $(7 \times 24)/\omega$ hours. Each vertex is represented by $v_{(i,t)}$ where i is the port index and t is the time index. The total number of vertices in G is $\omega|N|$.

The arc set A contains two disjoint subsets A_1 and A_2 . A_1 is the set of *terminal arcs* defined for each route $r \in R$. These arcs connect $v_{(i,t)}$ to $v_{(i,t+1)}$ for $1 \leq t < \omega$, and $v_{(i,\omega)}$ to $v_{(i,1)}$ for all $i \in N_r$. Terminal arcs represent berthing at the corresponding terminal for container unloading and loading. There exist $\omega|N_r|$ terminal arcs defined for route r , and the total number of terminal arcs is $\omega \sum_{r \in R} |N_r|$ in the network. A_2 is the set of *voyage arcs* defined for each route $r \in R$. Voyage arcs connect $v_{(i,t)}$ to $v_{(j,u)}$ where $i \in N_r$, $j = \text{succ}_r(i)$, and $1 \leq t, u \leq \omega$. As the name indicates, voyage arcs represent vessel voyages from one port to another. For example, when $\omega = 28$ (each time period is a quarter day), an arc connecting $v_{(1,19)}$ to $v_{(2,11)}$ represents

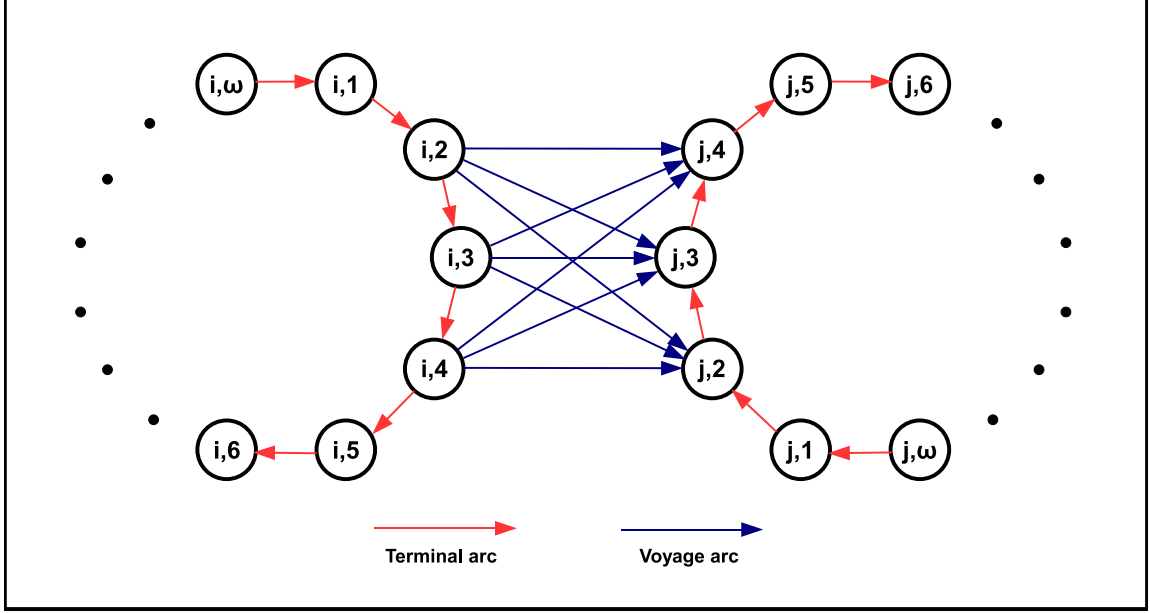


Figure 23: Representation of the graph used to model the VBSP

a voyage that starts at 19th time period (Friday 12:00pm) in the week at port 1, and ends at 11th (Wednesday 12:00pm) time period in the week at port 2. This arc represents a voyage duration of $20 + \omega k$ time periods ($5 + 7k$ days) where k is a nonnegative integer and given by the data. For each leg of route r , ω^2 voyage arcs are defined to cover all temporal possibilities. The total number of voyage arcs in G is $\omega^2 \sum_{r \in R} |N_r|$. Therefore, $|A_2| = \omega |A_1|$ and $|A| = \omega(\omega + 1) \sum_{r \in R} |N_r|$. This indicates that the choice of the time period length significantly affects the size of the graph constructed. Assuming a long time period length like one day reduces the size of the graph and probably makes the problem easier to solve; however, the solution obtained will be too rough to accurately handle dedicated terminal capacity constraints and any transshipment modeled. On the other hand, shorter time period lengths (like one hour) provide better planning while increasing the graph size and making the problem harder to solve in practice.

As mentioned before, a vessel can be in either one of the three states: sailing, waiting, and berthed. However, waiting can be considered as sailing with zero speed.

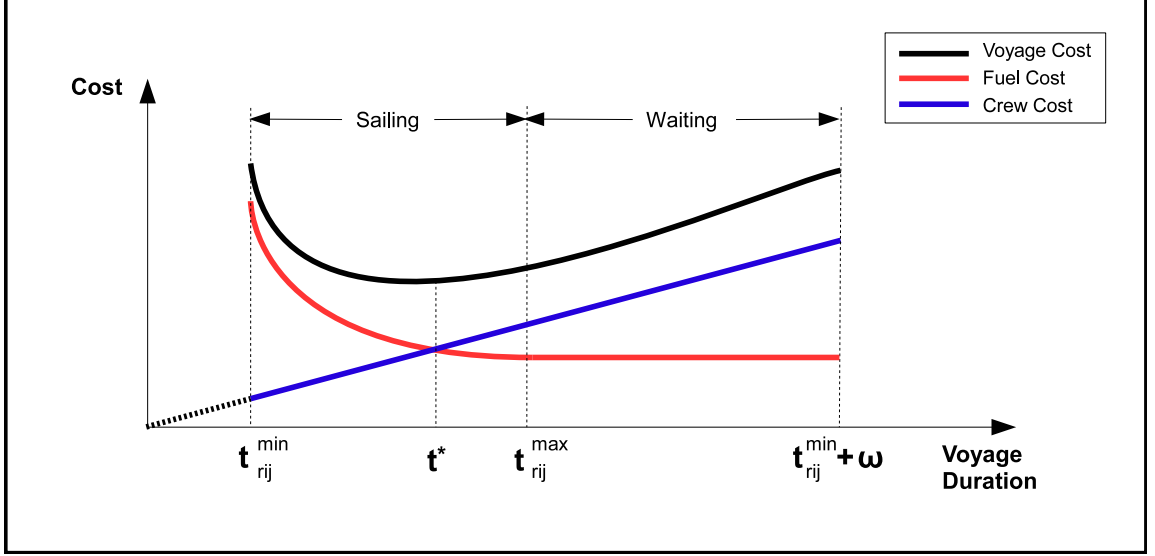


Figure 24: Illustration of the voyage cost function used in the model constructed for the VBSP

This reduces the number of states to two, and we say that a vessel is either sailing or berthed at a terminal at any time period. These two states can be covered by voyage arcs and terminal arcs in the graph constructed.

Voyage cost can be defined by combining fuel cost and crew cost as depicted in Figure 24 for each route and each leg of the route. It is a convex function of time having a minimum value at t^* . We assume that waiting, if there is any, occurs after sailing. The variation in leg duration is limited by one week. Therefore, the voyage cost is defined in $[t_{rij}^{min}, t_{rij}^{min} + \omega)$. Note that this does not limit sailing for more than one week which is very common for trans-Atlantic or trans-Pacific lags, and the maximum one week variation on leg duration is a valid assumption in practice.

In our model, V_{rij}^τ denotes the cost of route r ships going from port i to port j in τ time periods, P_{it}^r is the cost of receiving service by ships traversing route r at terminal i at time period t in the week, and Q_l^r is defined as the cost of operating route r with l ships. $f_{(i,t)(j,u)}^r$ represents the amount of flow from $v_{(i,t)}$ to $v_{(j,u)}$. The variable κ_l^r has a value of 1 if l ships are assigned to route r , and 0 otherwise. Furthermore,

$[t]_\omega \equiv (t - 1)(\text{mod } \omega) + 1$ is defined to calculate and expose time correctly. Hence, the VBSP can be formulated as follows.

$$\begin{aligned}
\text{Minimize} \quad & \sum_{r \in R} \sum_{\substack{i \in N_r \\ j = \text{succ}_r(i)}} \sum_{\tau = t_{rij}^{\min}}^{t_{rij}^{\min} + \omega - 1} V_{rij}^\tau \sum_{t=1}^{\omega} f_{(i,t)(j,[t+\tau]_\omega)}^r + \sum_{r \in R} \sum_{i \in N_r \cap N^M} \sum_{t=1}^{\omega} P_{it}^r f_{(i,t)(i,[t+1]_\omega)}^r + \\
& + \sum_{r \in R} \sum_{l = \nu_r^{\min}}^{\nu_r^{\max}} Q_l^r \kappa_l^r
\end{aligned} \tag{137}$$

subject to

$$\begin{aligned}
& \sum_{\substack{\tau = t_{rki}^{\min} \\ k = \text{prec}_r(i)}}^{t_{rki}^{\min} + \omega - 1} f_{(k,[t-\tau]_\omega)(i,t)}^r + f_{(i,[t-1]_\omega)(i,t)}^r = f_{(i,t)(i,[t+1]_\omega)}^r + \sum_{\substack{\tau = t_{ij}^{\min} \\ j = \text{succ}_r(i)}}^{t_{ij}^{\min} + \omega - 1} f_{(i,t)(j,[t+\tau]_\omega)}^r \\
& \forall r \in R, \forall i \in N_r, 1 \leq t \leq \omega
\end{aligned} \tag{138}$$

$$\begin{aligned}
& \sum_{\substack{\tau = t_{rij}^{\min} \\ k = \text{prec}_r(i)}}^{t_{rij}^{\min} + \omega - 1} f_{(k,[t-\tau]_\omega)(i,t)}^r = \sum_{\substack{\tau = t_{rij}^{\min} \\ j = \text{succ}_r(i)}}^{t_{rij}^{\min} + \omega - 1} f_{(i,[t+p_i^r]_\omega)(j,[t+p_i^r+\tau]_\omega)}^r \quad \forall r \in R, \forall i \in N_r, 1 \leq t \leq \omega
\end{aligned} \tag{139}$$

$$\sum_{l = s_r^{\min}}^{s_r^{\max}} \kappa_l^r = 1 \quad \forall r \in R \tag{140}$$

$$\sum_{\substack{i \in N_r \\ j = \text{succ}_r(i)}} \sum_{\tau = t_{rij}^{\min}}^{t_{rij}^{\min} + \omega - 1} \tau \sum_{t=1}^{\omega} f_{(i,t)(j,[t+\tau]_\omega)}^r + \sum_{i \in N_r} p_i^r = \omega \sum_{l = s_r^{\min}}^{s_r^{\max}} l \kappa_l^r \quad \forall r \in R \tag{141}$$

$$\sum_{\substack{\tau = t_{rij}^{\min} \\ j = \text{succ}_r(i)}}^{t_{rij}^{\min} + \omega - 1} \sum_{t=1}^{\omega} f_{(i,t)(j,[t+\tau]_\omega)}^r = 1 \quad \forall r \in R, \forall i \in N_r \tag{142}$$

$$\sum_{r \in R_i} f_{(i,t)(i,[t+1]_\omega)}^r \leq B_i \quad \forall i \in N^D, 1 \leq t \leq \omega \quad (143)$$

$$\kappa_l^r \in \{0, 1\} \quad \forall r \in R, s_r^{min} \leq l \leq s_r^{max} \quad (144)$$

$$f_{(i,t)(i,[t+1]_\omega)}^r \in \{0, 1\} \quad \forall r \in R, \forall i \in N_r, 1 \leq t \leq \omega \quad (145)$$

$$f_{(i,t)(j,[t+\tau]_\omega)}^r \in \{0, 1\} \quad \forall r \in R, \forall i \in N_r, j = succ_r(i), 1 \leq t, \tau \leq \omega \quad (146)$$

Objective function (137) minimizes total flow cost on voyage and terminal arcs as well as total cost of vessels used. Constraint (138) is the general flow balance constraint defined for each $v_{(i,t)}$. Vessels operating on route r are required to spend p_i^r time periods at each terminal $i \in N_r$. In the other words, a vessel that berths terminal i at time period t leaves at time period $t + p_i^r$. For this reason, Constraint (139) sets voyage arc inflow at time period t equal to voyage arch outflow at time period $t + p_i^r$. Constraint (140) guarantees that only one of the possible fleet size possibilities can be selected for each route r . In this constraint, s_r^{min} and s_r^{max} are the minimum and maximum number of vessels that can be assigned to route r respectively. The expression $\sum_{l=s_r^{min}}^{s_r^{max}} l \kappa_l^r$ gives the number of vessels operating on route r which is equivalent to the duration of route r in weeks. Therefore, Constraint (141) relates the duration of route r to the number of vessels assigned to route r . Constraint (142) ensures that the total amount of flow on each leg of each route is equal to one. This also initiates flow on G for each route considered. Capacity of dedicated terminals are considered by Constraint (143) which limits the total amount of flow passing on terminal arcs by the number of berths the terminal has. (144) sets κ_l^r as binary variables. Arc flow variables are also defined as binary variables by (145) and (146).

6.2.3 Analysis of the Network Flow Formulation

Definition 4 *A valid route flow (say defined for route r) is a flow that circulates on a cycle in G which consists of N_r voyage arcs (one voyage arc per leg (i, j)), and $\sum_{i \in N_r} p_i^r$ terminal arcs (p_i^r adjacent terminal arcs for each port i) where $i \in N_r$ and $j = \text{succ}_r(i)$.*

Lemma 6 *The mathematical program (138) - (146) solves the corresponding VBSP.*

Proof. A feasible solution of the VBSP can be represented as a collection of unit valid route flows defined for each route on G . Such a flow also respects dedicated terminal capacities. Any feasible solution to (138) - (146) defines a set of unit valid route flows because flow variables are binary and (142) guarantees unit flow on exactly one arc per leg for each route while (138) and (139) guarantee unit flow on exactly p_i^r adjacent terminal arcs for port i of route r . Furthermore, (143) ensures dedicated terminal capacity feasibility. Therefore, any feasible solution to the VBSP is also a feasible solution to the mathematical program (138) - (146) and vice versa. \square

The mathematical model (138) - (146) is a multi-commodity network circulation model with side constraints where each route r can be considered as a commodity. Constraint (143), which is actually a flow bundle constraint, is the only constraint that binds commodities (routes). In the other words, if Constraint (143) is relaxed, the problem can be decomposed into $|R|$ separate problems defined for each route. This also implies that any route which operates only on multi-user terminals can be optimized independent of others. In the general multi-commodity network flow problem, when flow bundle constraints are relaxed the subproblems are reduced to single commodity network flow problems with unimodular node-arc incidence matrix. This means that when flow variables are set as continuous variables, there exists an integer flow which is optimal for each subproblem. However, in our problem with the

model presented, this is not the case when Constraint (143) is relaxed, and (145) and (146) are replaced with (147) and (148).

$$0 \leq f_{(i,t)(i,[t+1]_\omega)}^r \leq 1 \quad \forall r \in R, \forall i \in N_r, 1 \leq t \leq \omega \quad (147)$$

$$0 \leq f_{(i,t)(j,[t+\tau]_\omega)}^r \leq 1 \quad \forall r \in R, \forall i \in N_r, j = \text{succ}_r(i), 1 \leq t, \tau \leq \omega \quad (148)$$

Lemma 7 *When Constraint (143) is relaxed, for given feasible κ_i^r values, there may exist a fractional solution to the linear program (138) - (142), (147), and (148) which is better than any of the feasible integer solutions.*

Proof. We prove the correctness of Lemma 7 using the following example. Consider a route r that consists of two ports 1 and 2. Let $\omega = 4$, $p_1^r = p_2^r = 2$. The cycle represented by red arcs and the cycle represented by blue arcs both define feasible solutions with unit flow, and they yield valid route flows meaning feasible solutions to the corresponding VBSP. However, the two cycles represented by solid arcs also define a feasible solution with a flow of $1/2$. Note that such a flow is not a valid route flow and cannot represent a feasible solution for the corresponding VBSP. On the other hand, such a flow does not use any terminal arcs and expected to have a smaller cost than any other valid route flow when terminal arc costs are high. \square

In order to eliminate fractional solutions similar to the one illustrated above, we define the following two valid inequalities. Expression (149) says that the total flow on terminal arcs for each route-terminal pair should be at least as large as the processing time required by the corresponding route at the corresponding terminal. For an integer solution this implies that at least p_i^r arcs should carry a unit flow for each route r visiting terminal i . Expression (150) indicates that starting at time t , the successive p_i^r terminal arcs should carry a flow at least as large as the total flow

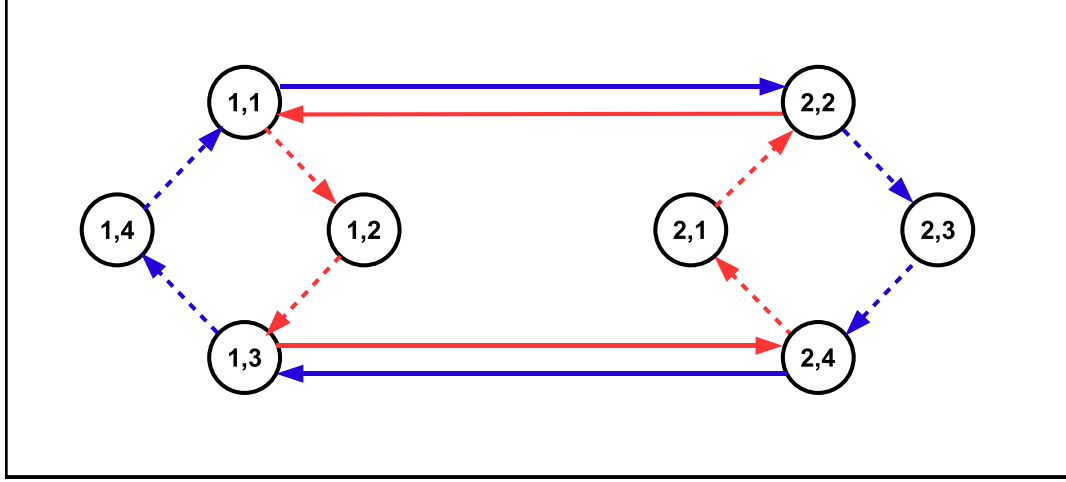


Figure 25: An example of a possible fractional optimal solution for the model

on voyage arcs entering $v_{(i,t)}$ for each route r . Therefore, (149) and (150) together with (139) guarantee that a flow entering $v_{(i,t)}$ via a voyage arc should follow the path $v_{(i,t)}, v_{(i,t+1)}, \dots, v_{(i,t+p_i^r)}$ on terminal arcs and leaves $v_{(i,t+p_i^r)}$ via a voyage arc.

$$\sum_{t=1}^{\omega} f_{(i,t)(i,[t+1]_{\omega})}^r \geq p_i^r \quad \forall r \in R, \forall i \in N_r \quad (149)$$

$$\sum_{\substack{\tau=t_{ij}^{min}+w-1 \\ \tau=t_{ij}^{min} \\ k=prec_r(i)}}^{t_{ij}^{min}+w-1} f_{(k,[t-\tau]_{\omega})(i,t)}^r \leq f_{(i,t+\bar{t})(i,[t+\bar{t}+1]_{\omega})}^r \quad \forall r \in R, \forall i \in N_r, \quad (150)$$

$$1 \leq t \leq \omega, \quad 0 \leq \bar{t} < p_i^r$$

Lemma 8 *When Constraint (143) is relaxed, for given feasible κ_i^r values, there exists an integer optimal solution to the linear program (138) - (142), and (147) - (150).*

Proof. In this formulation, any feasible flow defined for route r should travel on p_i^r successive arcs on each $i \in N_r$, and Constraint (141) sets the duration of total flow equal to K weeks where K is the number of vessels operating on the route. Furthermore, since there is no source or sink nodes and any feasible flow is a circulation, it can be decomposed into a set of flows traversing a finite number of cycles. Therefore, when Constraint (143) is relaxed, any feasible flow defined for each route should either

consists of valid route flows, or be a flow on a cycle illustrated as in part *A* of Figure 26.

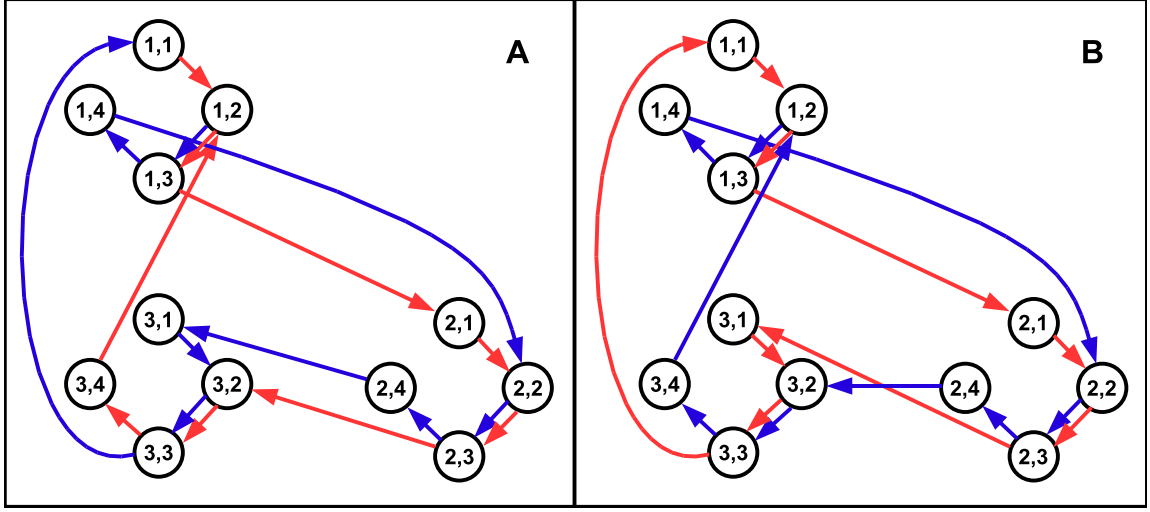


Figure 26: Existence of an optimal valid route flow on the graph

The flow on the cycle illustrated in part *A* has a value of $1/2$ and travels on two adjacent paths having durations of $K - \epsilon$ and $K + \epsilon$ weeks (a combination of more than two paths is also possible and the following discussion can easily be modified for any such case). In the illustration, the flow of $1/2$ units travel on adjacent blue (starts at $v_{(1,2)}$ and ends at $v_{(1,1)}$) and red (starts at $v_{(1,1)}$ and ends at $v_{(1,2)}$) paths having durations of 11 and 13 time periods respectively where $K = 3$, $\omega = 4$, and $p_i^r = 2$ for $i = \{1, 2, 3\}$. For any such flow, there exists a leg where one path has an arc having a duration of $k + \epsilon$ and the other one has an arc having a duration of $k - \epsilon$. In the illustration, the blue path has an arc from $v_{(2,4)}$ to $v_{(3,1)}$ having a travel time of 1 time period, and the red path has an arc from $v_{(2,3)}$ to $v_{(3,2)}$ having a travel time of 3 time periods. However, there exists another solution which uses the arcs from $v_{(2,4)}$ to $v_{(3,2)}$ and $v_{(2,3)}$ to $v_{(3,1)}$ instead of the arcs from $v_{(2,4)}$ to $v_{(3,1)}$ and $v_{(2,4)}$ to $v_{(3,1)}$. Since voyage cost is a convex function of time, such an alternative solution cannot have a higher cost. Moreover, it is a combination of a number (in this case two) of valid route flows as depicted in part *B* of Figure 26. In such a solution blue

and red cycles should have the same unit flow cost in order to be optimum, and a unit flow traversing any one of these cycles defines an alternate optimal solution which is integer. \square

Lemma 9 *When terminal arc flow variables, $f_{(i,t)(i,[t+1]_\omega)}^r$, are set binary for each $r \in R$, $i \in N_r \cap N^D$ and $1 \leq t \leq \omega$, there exists an optimal solution to (138) - (144), and (147) - (150) which defines a valid route flow for each route r , and hence provides an optimal solution to the VBSP.*

Proof. When dedicated terminal flow variables are set binary a solution illustrated by part A of Figure 26 cannot be feasible. Hence, for each route, any feasible solution can be decomposed into a set of valid route flows which use the same terminal arcs for dedicated terminals. Therefore, for each route, a unit flow traversing on the corresponding cycle of any one of the valid route flow in the optimal solution does not violate terminal capacity constraints and generates an alternate optimal solution which is integer. \square

6.3 Hardness of the Problem

In this section we investigate the theoretical hardness of the VBSP. We show that the VBSP is NP -hard in general whereas polynomial time algorithms can be constructed for special cases. We also discuss the effect of Lemma 8 and 9 as well as the impact of dedicated berth utilization on the practical hardness of the problem and the efficiency of the model presented. We assume that $\sum_{r \in R_i} p_i^r \leq \omega B_i$ for obvious feasibility and $\sum_{r \in R_i} p_i^r / \omega B_i$ gives the utilization of terminal $i \in N^D$.

Theorem 11 *If dedicated terminal capacity constraints are relaxed, the VBSP can be solved in polynomial time.*

Proof. Lemma 8 says that for given feasible κ_l^r values, the linear program (138) - (142), and (147) - (150) solves the VBSP with relaxed dedicated port capacity constraints optimally. The number of binary κ_l^r variables is equal to $s_r^{max} - s_r^{min} + 1$ for each route r , and Constraint (140) indicates that only one of them has a value of 1 in any feasible solution. Moreover,

$$s_r^{min} \geq \left(\sum_{\substack{i \in N_r \\ j = succ_r(i)}} t_{rij}^{min} + \sum_{i \in N_r} p_i^r \right) / \omega$$

and

$$s_r^{max} \leq \left(\sum_{\substack{i \in N_r \\ j = succ_r(i)}} (t_{rij}^{min} + \omega) + \sum_{i \in N_r} p_i^r \right) / \omega = \left(\sum_{\substack{i \in N_r \\ j = succ_r(i)}} t_{rij}^{min} + |N_r|\omega + \sum_{i \in N_r} p_i^r \right) / \omega$$

Consequently,

$$s_r^{max} - s_r^{min} \leq |N_r| \leq |N| \quad (151)$$

We know that when dedicated terminal capacities are relaxed, the problem can be decomposed into $|R|$ subproblems, one defined for each route. Since each of these problems can be solved by setting one of the $(s_r^{max} - s_r^{min} + 1)$ κ_l^r variables equal to 1 in turn, (151) indicates that the VBSP with relaxed dedicated port capacity constraints can be solved by solving $|R||N|$ linear programs, which can be achieved in polynomial time. \square

Theorem 11 yields the following two corollaries.

Corollary 1 *Optimum voyage and berth schedules for vessels operating on routes that visit only multi-user terminals can be found in polynomial time independent of other routes.*

Corollary 2 *A polynomially computable lower bound can be obtained for the VBSP by relaxing dedicated terminal capacity constraints.*

Theorem 12 *If $|N^D| = 1$, $B_i = 1$ for $i \in N^D$, and multi-user terminal cost is not time dependent, the VBSP can be solved in polynomial time.*

Proof. In such a VBSP variant, the cost of docking at multi-user terminals can be removed from the objective function since terminal cost and terminal processing times are constant. This also implies that the schedule at the dedicated terminal d has no impact on total cost, and a feasible solution can be found trivially since $\sum_{r \in R_d} p_d^r \leq \omega$. Therefore, the optimization problem is reduced to the problem of finding c_r which is the sum of the optimal voyage and vessel costs for each route r . Since scheduling at the dedicated terminal is trivial, this can be achieved by relaxing Constraint 143, and solving each subproblem defined for each route. Theorem 11 says that this can be achieved in polynomial time. \square

Theorem 13 *If $|N^D| = 1$, $B_i = 1$ for $i \in N^D$, and multi-user terminal cost is time dependent, the VBSP is NP-hard.*

Proof. When multi-user terminal cost is time dependent, the schedule generated for the dedicated terminal d affects route cost. For any time period that route r vessels are scheduled at the dedicated terminal, there may exist a different optimum cost (voyage and ship cost) for traversing the rest of the route. However, the optimum cost of docking at the dedicated terminal at time t for route r vessels, which is denoted by c_{rt} , can be found in polynomial time as Theorem 11 indicates. Therefore the VBSP can be transformed into a variant of the single machine scheduling problem with time dependent processing cost, and can be reformulated as follows.

$$\text{Minimize} \quad z = \sum_{r \in R_d} \sum_{1 \leq t \leq \omega} c_{rt} x_{rt} \quad (152)$$

subject to

$$\sum_{t=1}^{\omega} x_{rt} = 1 \quad \forall r \in R_d \quad (153)$$

$$\sum_{r \in R_d} \sum_{t=\hat{t}-p_d^r+1}^{\hat{t}} x_{r[t]\omega} \leq 1 \quad 1 \leq \hat{t} \leq \omega \quad (154)$$

$$x_{rt} \in \{0, 1\} \quad \forall r \in R_d, 1 \leq t \leq \omega \quad (155)$$

In the formulation, x_{rt} is a binary variable equal to 1 when route r vessels are scheduled to dock at time period t at the dedicated terminal and 0 otherwise.

We know that the single machine scheduling problem $1||\sum_r w_r U_r$ is NP -hard [60]. In this problem, p_r and d_r denotes the processing time and due time of job r . If t_r denotes the start time of job r , $U_r = 1$ if $t_r + p_r > d_r$ and $U_r = 0$ otherwise. w_r denotes weight defined for job r . Any instance of $1||\sum_r w_r U_r$ can be converted into the decision version of the corresponding VBSP formulated above as follows.

$$\omega = \sum_r p_r, \quad z = 0, \quad \text{and} \quad c_{rt} = \begin{cases} 0 & \text{if } t + p_r \leq d_r, \\ w_r & d_r < t + p_r \leq \omega \\ M & \text{otherwise.} \end{cases}$$

Therefore, we can conclude that when $|N^D| = 1$, $B_i = 1$ for $i \in N^D$, and multi-user terminal cost is time dependent, the VBSP is NP -hard. \square

Theorem 14 $B_i \geq 2$ for at least one $i \in N^D$, the VBSP is NP -hard.

Proof. Consider a dedicated terminal d with $B_d = 2$. By obvious feasibility assumption $\sum_{r \in R_d} p_d^r \leq 2\omega$; however, in order to have a feasible solution to the corresponding VBSP we need to answer whether or not there exists subsets S_1 and S_2 where $S_1 \cup S_2 = R_d$ and $S_1 \cap S_2 = \emptyset$ such that $\sum_{r \in S_k} p_d^r \leq \omega$ for $k = 1, 2$. This decision problem is NP -complete because it is an instance of the partition problem. Therefore, we conclude that when $B_i \geq 2$ for at least one $i \in N^D$, the VBSP is NP -hard. \square

Corollary 3 *The VBSP is NP-hard in general.*

Lemma 8 and Theorem 11 indicate that any route that operates only on multi-user terminals can be separated from the model and the optimal voyage schedules on such a route can be determined in polynomial time. This helps reducing the size of the NP-hard problem. Furthermore, Lemma 9 suggests that defining flow variables only on dedicated terminal arcs as binary is enough to find the optimal solution of the VBSP. This further reduces the practical hardness of the problem since at most $\frac{|N^D|}{|N|(\omega+1)}$ of the flow variables are defined for dedicated terminal arcs.

Utilization of dedicated terminals also impacts the practical hardness of the VBSP. Lower berth utilization is equivalent to relaxing the corresponding berth capacity constraint. Theorem 11 says that when this constraint is relaxed, the VBSP can be solved easily. Therefore, when dedicated berth utilization in the problem instance is lower we expect the proposed model to perform better in finding feasible integer solutions.

6.4 Further Discussion

An ocean carrier considers anticipated shipper demand while determining ship routes and scheduling ships on these routes. Routes should be constructed to cover all ports that the carrier provides service and ship schedules should enable efficient and effective cargo booking and routing while providing customer satisfaction and consistent demand.

In the VBSP, we basically focus on the temporal aspect of operating predetermined ship routes. The model we constructed determines optimal berthing times at each port while considering berth capacities at dedicated terminals and congested time periods at multi-user terminals. This tactical level planning affects decision making in the operational level. Because of this reason, while determining berth and

voyage schedules, the ocean carrier may want to consider additional important issues including terminal time windows, reasonable service line durations, planning of transshipments, and reliability of constructed itineraries. In this section, we describe how the proposed model can be extended to handle these additional important real life issues.

6.4.1 Terminal Time Windows

Terminal time windows may exist both in dedicated and multi-user terminals. One reason is the temporal regularity of demand. If some cargo regularly appears at a terminal at a given time period in each week, the ocean carrier may want ships of a suitable route to visit this terminal at or around the corresponding time period. Besides, terminal operators of multi-user terminals may provide time windows for the ocean carrier. In such cases, the ocean carrier can receive service at such terminals only in the time periods specified.

Terminal time windows can be incorporated in our model as follows. Let's $[l_i^r, u_i^r]$ denote a time window where l_i^r is the earliest time period that a ship operating on route r can dock at terminal i , and $u_i^r = l_i^r + \lambda_i^r$ is the latest time period it can leave. Here, $p_i^r \leq \lambda_i^r < \omega$ for feasibility. Then, Constraint (156) provided below correctly models terminal time windows.

$$\sum_{t=l_i^r}^{l_i^r+\lambda_i^r} f_{(i,[t]_\omega)(i,[t+1]_\omega)}^r \geq p_i^r \quad \forall r \in R, \forall i \in N_r \quad (156)$$

6.4.2 Service Line Durations

Ocean carriers compete on providing fast service for shippers. In order to have a consistent demand, duration of service lines should be as short as possible or at least no longer than ones of competitors. A liner route consists of two parts. We call the first part the service path, and the set of ports in the service path is denoted by $P_r \subset N_r$. The second part is called the return path, and the set of ports in the return

path is $N_r \setminus P_r$. The total duration of the route is related to the number of vessels operating on the route as depicted by Expression (141); however, the ocean carrier may want to limit the duration of the service path in order to provide fast service. This can be modeled by Constraint (157) provided below where μ_r is the maximum allowed duration for the service path of route r .

$$\sum_{\substack{i \in P_r \\ j = \text{succ}_r(i)}} \sum_{\tau = t_{rij}^{\min}}^{t_{rij}^{\min} + w - 1} \tau \sum_{t=1}^{\omega} f_{(i,t)(j,[t+\tau]_{\omega})}^r + \sum_{i \in P_r} p_i^r \leq \mu_r \quad \forall r \in R \quad (157)$$

6.4.3 Transshipments

Some of the ports that vessels of an ocean carrier visit while traversing their routes may act as transshipment ports. Transshipment ports are intermediate ports where cargo may change ships (routes). Transshipments are inevitable since an ocean carrier can only operate a limited number of routes, and the origin and destination of many cargoes do not both appear on the same route. Therefore, at some terminals cargo is moved from one ship to another, in the other words, from one route to another. This can be seen as what airline passengers having itineraries with multiple flights do where airplanes follow their own regular routes and passengers change airplanes to reach their destinations.

In order to reduce waiting of cargo at terminals the ocean carrier would like to overlap berthing of ships on which a transshipment is scheduled as much as possible. Assume that a transshipment is planned between two routes r and q at terminal i where $p_i^r \leq p_i^q$. If δ_{rq}^i denotes the maximum allowed difference between the time period route r vessels berth and the time period route q vessels berth at terminal i , the following constraints can be used to handle the corresponding transshipment. Constraint (158) is used if route r ships need to berth earlier, and Constraint (159)

is employed otherwise.

$$f_{(i,t)(i,[t+1]_\omega)}^r \leq \sum_{t'=t}^{t+\delta_{rq}^i} f_{(i,[t']_\omega)(i,[t'+1]_\omega)}^q \quad 1 \leq t \leq \omega \quad (158)$$

$$f_{(i,t)(i,[t+1]_\omega)}^r \leq \sum_{t'=t-\delta_{rq}^i}^t f_{(i,[t']_\omega)(i,[t'+1]_\omega)}^q \quad 1 \leq t \leq \omega \quad (159)$$

6.4.4 Schedule Reliability

The reliability of service promised by ocean carriers is very important for shippers. Highly reliable itineraries increase the competitive advantage of an ocean carrier and have a positive effect on demand. In this section we describe how the reliability of voyages and berth schedules can be increased by appropriately modifying the instance data and parameters.

6.4.4.1 Voyage Reliability

Voyage reliability is defined as the ability to sail at the planned speed. In the previous section, we mentioned that the maximum speed of a ship on a leg is not only determined by the speed limit of the ship but also affected by the leg distance, fuel consumption, and fuel tank capacity. Schedules with very high speeds may increase the possibility of running out of fuel, and hence decreases reliability of the schedule. Furthermore, voyage speed and fuel consumption is also affected by weather conditions. As expected, under severe weather conditions, the maximum speed that a ship can reach decreases while the fuel consumed to achieve that speed increases. Since our problem is a tactical level problem and weather cannot be predicted in advance, the ocean carrier may want to put an additional speed buffer, ϵ_{ij}^r , for sailing from port i to port j within route r . Therefore, the corresponding modified maximum sailing speed $\hat{\nu}_{rij}^{max}$ and minimum possible sailing time \hat{t}_{rij}^{min} are defined as follows.

$$\hat{\nu}_{rij}^{max} = \nu_{rij}^{max} - \epsilon_{ij}^r \quad (160)$$

$$\hat{t}_{rij}^{min} = \frac{d_{ij}}{\nu_{rij}^{max} - \epsilon_{ij}^r} \quad (161)$$

Such a modification may not be necessary for all legs. That's because the problem has a built-in incentive to move away from the maximum speed since sailing around maximum speed is expensive. However, especially for long legs fast sailing speeds can create large absolute reduction in route duration, and decreases the number of ships required by the route which may generate savings in ship cost. Therefore, for some instances sailing at high speeds may be attractive for the model, and buffers may be employed efficiently in order to increase voyage reliability of such legs even though it can increase the total cost.

6.4.4.2 Berth Reliability

Consider a berth schedule where ships are planned to dock immediately after another for processing. In such a schedule, any delay observed in the voyage of a ship creates a delay in the berthing time of that ship and the following ship scheduled on the berth. This may cause a catastrophic overall disruption of the voyage and berth schedule constructed. By the obvious feasibility assumption, $\sum_{r \in R_i} p_i^r \leq \omega B_i$ for terminal $i \in N^D$. Therefore, $\omega B_i - \sum_{r \in R_i} p_i^r$ gives the idle capacity of the terminal. Distribution of this idleness over the berth schedule determines the reliability of the schedule. If idle time periods are grouped together in the schedule, we expect the schedule be less reliable as opposed to an alternative schedule where idle time periods are distributed between ship processing. In the model presented, such reliable schedules can be achieved by properly defining a berth buffer, ε_i^r , for each dedicated terminal i visited by ships of route $r \in R_i$. To preserve obvious feasibility,

$$\sum_{r \in R_i} \varepsilon_i^r \leq \omega B_i - \sum_{r \in R_i} p_i^r \quad (162)$$

and processing times are modified as follows.

$$\hat{p}_i^r = p_i^r + \varepsilon_i^r \quad (163)$$

We assume that buffer is placed after vessel processing in the schedule. Hence, in order to maintain correctness of the model, the voyage cost in the following leg should be modified as described below.

$$\hat{V}_{rij}^{\tau} = V_{rij}^{\tau + \varepsilon_i^r} \quad t_{rij}^{min} - \varepsilon_i^r \leq \tau < t_{rij}^{min} + \omega - \varepsilon_i^r. \quad (164)$$

Furthermore, the minimum sailing time in the following leg should be adjusted as follows.

$$\hat{t}_{rij}^{min} = t_{rij}^{min} - \varepsilon_i^r \quad (165)$$

$$(166)$$

The resulting modified voyage cost function for the corresponding leg is illustrated in Figure 27.

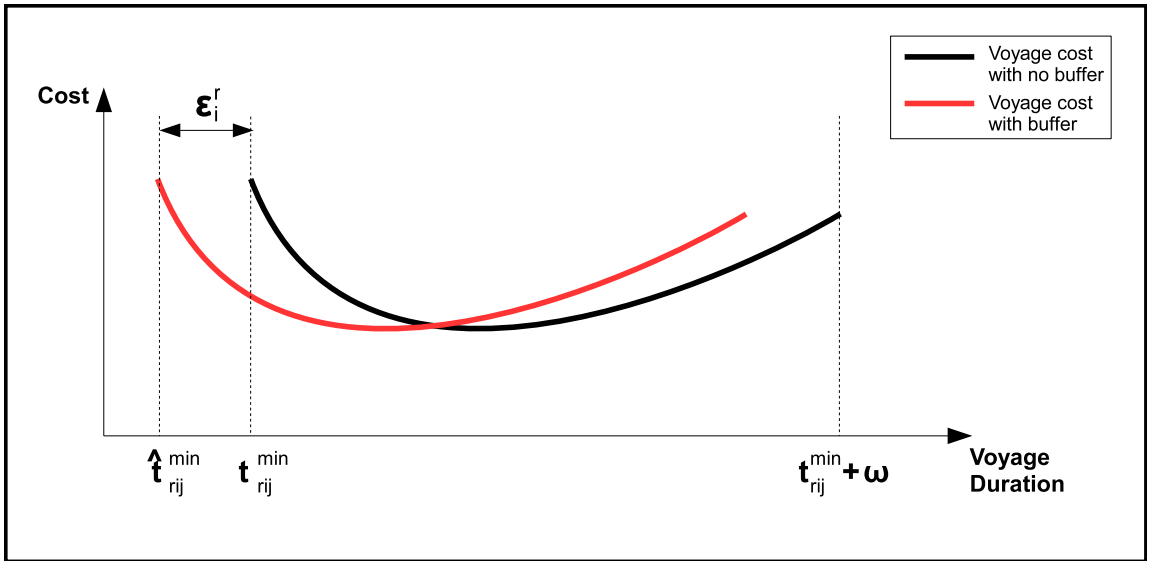


Figure 27: Change in voyage cost function with berth schedule buffer

Such a modification creates more reliable schedules with possible total cost increases. Furthermore, (163) yields higher berth utilizations, which means a tighter Constraint (143), and makes the problem practically harder to solve. Moreover, when $B_i \geq 2$ for terminal $i \in N^D$ where buffers are used, parallel to Theorem 14, feasibility

of the problem may be endangered. Therefore, the decision maker should be careful while determining ε_i^r values.

6.4.5 Fleet Restrictions

In the model proposed, we define the number of vessels operating on each route as a decision variable, and assume that each vessel assigned to each route comes with a cost. In such a case, finding a feasible solution is not hard since any arbitrary feasible berth schedule also defines feasible voyages between ports. However, such an arbitrary solution may yield very long leg and route durations and may require many vessels. In the optimal solution found by the model, since vessel cost is taken into account, we do not expect too many vessels required. However, the decision maker may still want to limit the total number of ships used for all routes or for a subset of routes which require similar type of vessels. This can be modeled by adding the following constraint into the model where $\hat{R} \subseteq R$, and K denotes the maximum number of ships that can be used by routes in \hat{R} .

$$\sum_{r \in \hat{R}} \sum_{l=s_r^{min}}^{s_r^{max}} l \kappa_l^r \leq K \quad (167)$$

6.5 Computational Experiments

In this section we present the results of our computational experiments. The efficiency of the mathematical model proposed and the practical hardness of the problem has been discussed in Section 6.2.3 and Section 6.3 respectively. Parallel to these discussions, our experiments show that the mathematical model proposed can efficiently solve realistic instances with an off-the-shelf commercial solver. In our experiments, we use CPLEX 9.0 on an Intel 2.4 GHz Pentium processor running Linux with 2GB memory.

We first describe how test instances used are generated. The performance of the

model is presented on these variable size test instances. Then, impact of time windows, service line durations, transshipments, and schedule reliability improvements on the cost and the solution time will be discussed.

6.5.1 Instance Generation

The VBSP uses the output of a higher level network design or route construction problem. Thus, we relate the size of our problem instances to the size of instances solved by previous papers which study the problem of liner route construction. For example; [3] solves the corresponding route determination problem for instances having 6 to 20 ports. The number of routes in their final solution ranges from 3 to 8 routes. Test instances they consider are close to being realistic since there exists a limited number of major ports in the world and the number of routes that an ocean carrier can operate is restricted by the size of its fleet. For instance; a route from Asia to US East Coast may have a total duration of 8 weeks, which means 8 ships are required to operate such a route, and most large ocean carriers have around 100 vessels. Therefore, we consider networks of 3 sizes having 10, 15, and 20 ports denoted by A , B , and C respectively. For each of these networks we consider five different route sets. We consider instances with 5, 7, 11, 13, and 15 routes for A ; instances with 7, 9, 11, 13, and 15 routes for B ; and instances with 9, 11, 13, 15, and 17 routes for C . Each route visits between 3 and 10 ports. Since we only need to consider routes visiting at least one dedicated terminal (Corollary 1), such networks represent realistic instances and are compatible with published itineraries of major ocean carriers (see [46], [55], [26], and [6]). In A , ports are distributed in two regions representing US East Coast and Southeast Asia. In B , we add ports located in regions representing US West Coast, Korea, and Japan to the ones in A . In addition to these, network C has ports in regions representing Europe, and South Asia. The distance matrix is generated according to the published itineraries of major ocean carriers.

The number of dedicated terminals ranges from 1 to 5. In our base instances we set dedicated berth utilization as 65% for all dedicated terminals. The impact of higher berth utilization is also investigated in the experiments. Container processing time is generated randomly between half a day and 3 days for each terminal, and adjusted randomly in order to match utilization figures used for dedicated terminals. The fuel consumption and crew cost of ships operating on each route is randomly generated within appropriate realistic values. Similarly, multi-user terminal cost is generated according to a randomly generated busyness pattern for each terminal. We assume a fixed ship cost for each route which is a relatively higher figure compared to voyage and terminal costs. Finally, we assume that each time period represents quarter days, hence, $\omega = 28$.

6.5.2 Results

We set the optimality gap limit as 0.03% and let CPLEX terminate after 3 hours if optimality cannot be proved earlier. Table 24 summarizes results. Each grid presents the polynomially computable lower bound, the optimal cost, total voyage cost, total terminal cost, total ship cost, and run time in seconds from top to bottom respectively for each network and dedicated terminal combination. For example; the first grid 1 – A5 provides the result of the scenario with 5 routes and 1 dedicated terminal on network A, which consists of 10 ports. For each network, we solve instances with fewest number of routes only with one dedicated terminal, and as the number of routes increase for the same network we define additional terminals as dedicated. For example; the instance A5 is solved with only one dedicated terminal. A7 is first solved with the same single dedicated terminal. Then, an additional terminal is set as dedicated, and the instance is solved with two dedicated terminals, and so on. In the other words, going from one row to the next row for a network (column) exposes the effect of leasing an additional terminal. Same terminals are defined as dedicated

for instances on the same row.

Computational experiments show that the model proposed can be used efficiently to solve realistic instances with commercial MIP solvers. Optimal solutions are found in 52 to 1245 seconds, which is acceptable for a tactical level problem. We expect to see an increase in computation time as problem size increases by defining more routes for a given network; namely, as we go from left to right on the same row within the network. Although this is the general pattern, for some networks the computation time is not affected much or even dropped. For example, the scenario $1 - A5$ is solved in 77 seconds whereas $1 - A13$ takes only 61 seconds.

Table 24: Summary of results for test VBSP instances

N^D	A5	A7	A9	A11	A13	B7	B9	B11	B13	B15	C9	C11	C13	C15	C17
1	32401	42515	47468	58529	66078	40444	48283	61006	69138	80134	50758	59974	70550	82092	93724
	32444	42517	47499	58550	66697	40482	48325	61027	69148	80216	50812	59984	70560	82124	93866
	8671	11482	12617	17262	17530	11375	12274	15057	18352	23056	14528	15655	19049	22870	27465
	773	1035	882	1288	1167	1107	1051	970	1796	2160	1284	1329	1511	2254	2401
2	23000	30000	34000	40000	48000	28000	35000	45000	49000	55000	35000	43000	50000	57000	64000
	77	52	82	90	61	73	63	104	87	306	131	76	137	116	433
		42177	47174	58141	66317		47959	60965	68878	79835		59792	70344	81606	93489
		42199	47216	58169	66377		47985	61045	68937	79942		59839	70377	81717	93654
3	-	11432	12569	17229	17529	-	12168	15069	18293	23010	-	15626	18972	22883	27459
		767	647	940	848		817	976	1644	1932		1213	1405	1834	2195
	30000	34000	34000	40000	48000		35000	45000	49000	55000		43000	50000	57000	64000
	117	119	119	107	96		95	122	196	377		291	271	193	628
4			46950	58050	66265			60764	68689	79688			70053	81502	93330
			46981	58162	66434			60891	68752	79742			70102	81532	93498
	-	-	13314	17237	17588	-	-	14967	18359	23019	-	-	18864	22869	27387
			667	925	846			924	1393	1723			1238	1663	2111
5			33000	40000	48000			45000	49000	55000			50000	57000	64000
			125	358	551			373	289	639			333	215	887
				57784	66006				68565	79418				81214	93094
				57830	66154				68594	79563				81277	93242
5	-	-	-	17184	17537	-	-	-	18247	22158	-	-	-	22839	27451
				646	617				1347	1405				1438	1791
				40000	48000				49000	56000				57000	64000
				545	616				394	880				433	1054
5					65976					79292					92822
					66069					79389					92963
	-	-	-	-	17538	-	-	-	-	22235	-	-	-	-	27395
					531					1154					1568
5					48000					56000					64000
					1071					1245					1137

This is because of the fact that the number of routes visiting the single dedicated terminal is same in both scenarios, which is 5. Since binary variables are defined only for these routes at the corresponding terminal, the branching and bounding effort is expected to be similar for the scenarios on the same row. Hence, the increase in computation time, if there is any, is due to the additional routes that do not visit any dedicated terminal and do not require any binary variables. We know that such routes can be optimized separately in polynomial time. Therefore, the planner may expect to observe very little computation time increase when additional routes visiting only multi-user terminals are added in the service network if such routes are handled separately. On the other hand, we see a more consistent increase in computation time as we go from top to bottom in the same column. This is expected because as more terminals are defined dedicated more variables are set binary and less number of routes can be optimized separately in polynomial time. The largest scenario 5 – *C17*, which has 20 ports, 5 dedicated terminals, and 17 routes, takes 1137 seconds to solve optimally. The longest computation time is observed for the scenario 5 – *B15*, which is 1245 seconds.

Table 24 also shows the strength of the lower bound mentioned in Corollary 2. On average, the lower bound is only 0.11% below the optimal solution. We can say that the strength of the lower bound is not affected by the problem size. For scenario 1 – *A7*, the lower bond is only 0.005% below the optimal solution, whereas for scenario 3 – *A13* it is 0.254%, which is the largest gap observed. This means that the decision maker can have a good estimate of the optimal route operating cost by quickly computing the lower bound.

In the following sections, we present the effect of other considerations like terminal time windows, service line duration limits, transshipments, and schedule reliability improvements on the optimal operating cost and solution time. We select the scenario 5 – *B15* as our base scenario since it requires the most computation time to find the

optimal solution.

6.5.2.1 Terminal Time Windows

We generate 10 time windows for randomly selected port-route pairs in scenario 5 – B15. Results are presented in Table 25. Row 0 provides the statistics for the optimal solution with no terminal time windows. Row k presents the result in presence of the first k time windows generated. Hence, each row includes the time windows considered in the previous row and an additional one.

Table 25: Effect of time windows on the optimal solution

TW	LB	Total Cost	Voyage Cost	Terminal Cost	Vessel Cost	Run Time
0	79292	79389	22235	1154	56000	1245
1	79346	79420	22114	1306	56000	1026
2	79470	79534	22152	1382	56000	963
3	79741	79784	22139	1645	56000	956
4	79883	79933	22142	1791	56000	1067
5	79934	80020	23004	2016	55000	812
6	79952	80032	23024	2008	55000	566
7	80166	80215	23016	2199	55000	657
8	80409	80475	22453	2022	56000	623
9	80614	80686	22423	2263	56000	604
10	80650	80690	22413	2277	56000	552

We observe an average cost increase of 0.16% for each time window added. Since time windows may force berthing at unfavorable time periods, a pretty consistent increase in terminal cost is observed. The model tries to compensate that increase in terminal cost by selecting less costly voyage arcs in the first four scenarios, and by reducing the total number of vessels in rows 5 to 7. Time windows also increase the lower bound. The lower bound is 0.12% below the optimal solution in the base scenario, and 0.08% below the optimal solution on average for scenarios with time windows. Hence, the lower bound performance is not negatively affected by time window constraints.

We observe a downward trend in computation cost as we add more time windows. This is expected because adding time windows is equivalent to partially fixing variables or restricting the feasible set, hence makes the problem easier to solve.

6.5.2.2 Service Line Durations

We generate 5 service line duration constraints for 5 randomly selected routes. The number of legs in each service line varies between 3 and 5. Table 26 summarizes results. Each row represents the scenario generated by adding another constraint to the scenario given by the previous row. Row 0 provides the statistics for the optimal solution with no service line duration constraints.

Table 26: Effect of service line duration limits on the optimal solution

SLD	LB	Total Cost	Voyage Cost	Terminal Cost	Vessel Cost	Run Time
0	79292	79389	22235	1154	56000	1245
1	79362	79408	22225	1183	56000	721
2	79377	79423	22269	1154	56000	698
3	79406	79452	23118	1334	55000	732
4	79429	79479	23174	1305	55000	802
5	79441	79491	23184	1307	55000	1054

The average increase in cost is observed as 0.03% for each service line duration constraint added. Since such constraints force faster sailing, an increase in voyage cost is observed. Shorter service lines may also imply shorter route durations which translates into fewer ships. This is observed in scenarios 3, 4 and 5; however, ships should sail faster on each leg in such cases, hence voyages become a lot costly. Service line duration constraints also increase the lower bound. The lower bound is computed 0.06% below the optimal solution on average for scenarios with service line durations. Experiments also show that service line duration constraints do not increase computation time. Hence, they can be handled efficiently by the model proposed.

6.5.2.3 Transshipments

We generate 10 transshipment constraints where for each constraint a terminal and two routes visiting the terminal is randomly selected. Table 27 summarizes results. Each row represents the scenario generated by adding another constraint to the scenario given by the previous row. Row 0 provides the statistics for the optimal solution with no transshipment constraints.

Table 27: Effect of transshipments on the optimal solution

TRS	LB	Total Cost	Voyage Cost	Terminal Cost	Vessel Cost	Run Time
0	79292	79389	22235	1154	56000	1245
1	79292	79397	22203	1194	56000	1271
2	79292	79405	22185	1220	56000	1368
3	79292	79436	22218	1218	56000	1259
4	79292	79531	22225	1306	56000	1375
5	79292	79539	22192	1491	56000	3496
6	79292	79615	22220	1395	56000	1654
7	79292	79619	22229	1390	56000	1422
8	79292	79621	22231	1390	56000	1928
9	79292	79785	23054	1731	55000	3050
10	79292	79799	23949	1759	55000	2746

Because the lower bound is computed by treating each route separately, transshipment constraints do not have any influence on the lower bound. Therefore, as we have more such constraints, we expect to have a looser lower bound. For example, the lower bound is only 0.12% below the optimal solution for Row 0 whereas the gap is 0.64% for Row 10 where all 10 transshipment constraints are included in the model. Each transshipment modeled generates a 0.05% increase in the total operating cost on average. Most of this cost increase is due to the increase in terminal cost since overlapping of vessel schedules at terminals may require unfavorable berthing times. We may also expect to see an increase in computation time. Scenarios with transshipment constraints took 1957 seconds on average.

6.5.2.4 Schedule Reliability

We also investigate the effect of voyage speed buffers and berth buffers which can be used to increase the reliability of schedules. We first generate 10 voyage speed buffers for one leg of 10 randomly selected routes. Table 28 summarizes results. Each row includes the buffers considered in the previous row and an additional one. Row 0 is the base scenario with no speed buffers. We see that some speed buffers do not have any effect on the optimal solution. This is because of the expected tendency to move away from the maximum speed because sailing at maximum speed is highly costly. As a result, only 4 out of 10 additional constraints changed the optimal solution. We

see an average cost increase of 0.57% for these constraints. This is because speed buffers force slow sailing when the corresponding constraint is binding. This means that the total route duration and hence vessel cost may increase or in order to keep the number of ships the same, very costly speed increases may be utilized on the other legs of the route. We can also say that voyage speed buffers also impact the lower bound and the lower bound performance do not diminish in the presence of such constraints.

Table 28: Effect of voyage speed buffers on the optimal solution

VR	LB	Total Cost	Voyage Cost	Terminal Cost	Vessel Cost	Run Time
0	79292	79389	22235	1154	56000	1245
1	79292	79389	22235	1154	56000	1071
2	79687	79773	22543	1230	56000	1904
3	79687	79773	22543	1230	56000	1969
4	80292	80364	22088	1276	57000	1495
5	80292	80364	22088	1276	57000	1679
6	80875	80961	22741	1220	57000	1837
7	80875	80961	22741	1220	57000	1872
8	81135	81221	22963	1258	57000	1145
9	81135	81221	22963	1258	57000	1251
10	81135	81221	22963	1258	57000	1138

In the base 5 – B15 scenario, each dedicated terminal is visited by vessels of 5 different routes. Since we set utilization at 65%, the total idle time at these dedicated terminals is 10 time periods (60 hours) per week. In order to evaluate the effect of berth buffers, we generate two sets of scenarios. We define $\varepsilon = 1$ in the first set and $\varepsilon = 2$ in the second set for each possible route and dedicated terminal combination. Table 29 and 30 present results. In these tables, Row 0 gives statistics for the optimal solution with no berth buffer, and Row k represents the optimal solution with berth buffers on k dedicated terminals. Similar to other experiments, each row includes buffers defined by the previous row and an additional one.

Berth buffers do not have any effect on the lower bound since the lower bound is computed by relaxing capacity constraints at dedicated terminals and an each route is evaluated separately. Therefore, we expect to have looser lower bounds as we place

more berth buffers in the model.

Table 29: Effect of berth buffers with $\varepsilon = 1$ on the optimal solution

BRwB1	LB	Total Cost	Voyage Cost	Terminal Cost	Vessel Cost	Run Time
0	79292	79389	22235	1154	56000	1245
1	79292	79437	22189	1248	56000	1881
2	79292	79463	22224	1239	56000	2015
3	79292	79496	22227	1269	56000	3038
4	79292	79530	22239	1291	56000	4089
5	79292	79556	22235	1321	56000	5078

Table 29 summarizes the scenarios with $\varepsilon = 1$ for each route and dedicated terminal combination. Experiments show that as we place buffers of one time periods for each vessel at a dedicated berth, we see an average of 0.04% increase on the total cost. Since buffers are simply included in the processing time of vessels in the model, adding buffers is equivalent to increasing the utilization at the corresponding terminal. In our scenarios, adding $\varepsilon = 1$ to the processing time of each vessel increases the utilization of the corresponding berth from 65% to 82.5%. This means a tighter Constraint (143), and problem is expected to become harder to solve in practice. This is what we observe since computation time increases as we apply buffers at additional berths.

Table 30: Effect of berth buffers with $\varepsilon = 2$ on the optimal solution

BRwB2	LB	Total Cost	Voyage Cost	Terminal Cost	Vessel Cost	Run Time
0	79292	79389	22235	1154	56000	1245
1	79292	79471	22187	1284	56000	1424
2	79292	79575	23106	1469	55000	3777
3	79292	79669	22295	1374	56000	6837
4	79292	79737	23058	1679	55000	7215
5	79292	79844	23105	1739	55000	10800

Table 30 summarizes the statistics for the scenarios where $\varepsilon = 2$. Since buffers are larger, the cost increase is observed as 0.11% on average. Note that when buffers are applied at a dedicated terminal, the berth utilization increases from 65% to 100%, which implies a very tight capacity constraint. Therefore, we observe sharper increases in computation time as $\varepsilon = 2$ are used at additional terminals. In the scenario

presented in Row 5 of Table 29, utilization is 100% at all dedicated terminals, and the computation time hit the limit of 3 hours, and CPLEX terminated the run before optimality is proved.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

This thesis provides a comprehensive study on planning problems related to berth and quay cranes which are the most important resources in container terminals at seaports. It contributes filling many of the gaps in the literature that appear due to recent trends and changes in maritime logistics, like the introduction of mega-ships and increasing popularity of flexible continuous berth structures.

In Chapter 2, a variant of berth allocation problem (BAP) which considers dynamic arrival of vessels and a single berth having a long continuous structure that can serve multiple vessels simultaneously is considered. Two different mixed integer programming (MIP) formulations are provided, and a meta-heuristic algorithm based on tabu search which employs a novel nested neighborhood structure to solve large problems is developed. A polynomially computable lower bound is also introduced which can be computed quickly in polynomial time, and is provably tighter than the bound generated by solving the LP relaxation of the associated MIP. Computational experiments show that the algorithm is able to provide high quality solutions in relatively short computation times. Instances with 10 to 14 vessels were solved optimally by the nested tabu search algorithm under 4 seconds, whereas, for some of these instances, it took days to solve with the MIP. More realistic instances with 20 to 30 vessels were also solved within 100 seconds. For those instances, we were able to reduce the optimality gap of a reasonable initial solution by 70% on average.

In chapter 3, the multiple berth allocation problem (MBAP), where multiple physically-disjoint berths may be used to moor arriving vessels is introduced. Arriving vessels can be assigned to any berth within the terminal, but the processing

time required for vessels can change depending on berth assignments. Models and solution methods developed for the BAP are generalized to handle the MBAP. A lower bound analysis is again conducted, and the results are again used in the development of polynomially-computable bounds provably stronger than that of the LP relaxation. Computational experiments show that the solution approach designed provides near optimal solutions for real size problems in reasonable amount of time.

The quay crane scheduling problem (QCSP) is the focus of Chapter 4. Most earlier studies concerning quay crane scheduling focus on detailed models applicable to either one vessel or a set of vessels docked at the time of decision making. In this thesis, a QCSP variant defined for a given berth schedule is analyzed. Hence, in the problem variant studied, not only are the vessels that are present at the berth at the time of decision making considered but also vessels planned to arrive later. Concepts of crane blocking and crane shifting is defined, and two crane scheduling methods; dedicated crane scheduling and roaming crane scheduling are introduced. Exact optimization models are developed for each crane scheduling method under both blocking and shifting assumptions. These models are used to analyze the methods computationally using a set of small instances. It is shown that roaming crane scheduling with crane shifting can provide significantly better operational plans by increasing crane utilization. A tabu search algorithm to solve realistic instances under this scenario is designed. The individual crane assignment problem (CAP) is also introduced and a polynomial time solution method is provided. Computational experiments indicate that the tabu search algorithm designed is able to find optimal solutions almost instantaneously for small problems with 10 to 14 vessels and 6 cranes. For larger problem instances with 20 to 30 vessels and 10 cranes, the optimality gap was improved by approximately 68% within 5 to 15 seconds respectively. Designing fast search algorithms for the case with no crane shifting or crane roaming may be an interesting subject for future research activities.

In chapter 5, the problem of simultaneous scheduling of berth and quay cranes (BQCSP) is discussed. A mixed integer program and a two phase tabu search heuristic is provided. Two polynomially-computable lower bounds are also introduced. Computational experiments show that the algorithm designed can reduce the optimality gap of a reasonable initial solution by 66% within 1 minute for moderate size instances with 10 to 14 vessels and 6 cranes. The algorithm provided even better improvement for larger instances with 20 to 30 vessels and 10 cranes. An average of 77% reduction in the optimality gap of the initial solution in 3 to 30 minutes is observed. This thesis also shows that simultaneous scheduling of berth and quay cranes may yield savings as large as 35% over the hierarchical planning approach used by terminal operators. The simultaneous scheduling approach can easily be extended for the multiple berth case in a future study. The nested neighborhood structure and the two stage search procedure provided can be utilized efficiently to design a solution method for such an extension.

Finally, the voyage and berth scheduling problem (VBSP) is introduced in Chapter 6, which is the problem of scheduling vessel voyages considering dedicated berth resource limitations. A model based on multi-commodity network flow is presented. It is proven that when only dedicated terminal capacities are relaxed, the model can be decomposed for each route and an optimal integer flow can be found by an improved version of the LP relaxation with the help of additional valid inequalities. This result is used to reduce the problem size. Methods to handle transshipments, terminal time windows and service level requirements are provided. Ideas to improve the reliability of schedules are also discussed. The VBSP is defined for predetermined vessel routes. A future study can be conducted on the problem of determining vessel routes considering berth resource limitations. The efficiency of the model and the ideas presented in this thesis can guide a smooth integration of the vessel routing problem and the voyage and berth scheduling problem.

REFERENCES

- [1] “Sysmic risk management for port systems.” <http://www.neesgc.gatech.edu/>. Accessed in October 2008.
- [2] AGARWAL, R., *Network Design and Alliance Formation for Liner Shipping*. PhD thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, 2007.
- [3] AGARWAL, R. and ERGUN, O., “Ship scheduling and network design for cargo routing in liner shipping,” *Transportation Science*, vol. 42, 2008.
- [4] AHUJA, R. K., MAGNANTI, T. L., and ORLIN, J. B., *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.
- [5] AL-MAHMEED, A. S., “Tabu search combination and integration,” *Meta-Heuristics: Theory and Applications*, I. H. Osman and J. P. Kelly (eds.), Kluwer Academic Publishers, pp. 319–330, 1996.
- [6] APL. <http://www.apl.com>. Accessed in July 2008.
- [7] BAIRD, A. J., “Container vessels in the new millennium: Implications for sea-ports,” *Singapore Maritime and Port Journal*, 2001.
- [8] BAIRD, A. J., “Optimising the container transshipment hub location in northern europe,” *Journal of Transport Geography*, vol. 14, pp. 195–214, 2006.
- [9] BROWN, G. G., LAWPHONGPANICH, S., and THURMAN, K. P., “Optimizing ship berthing,” *Naval Research Logistics*, vol. 41, pp. 1–15, 1994.
- [10] CHARON, I. and HURDY, O., “Mixing different components of metaheuristics,” *Meta-Heuristics: Theory and Applications*, I. H. Osman and J. P. Kelly (eds.), Kluwer Academic Publishers, pp. 589–604, 1996.
- [11] CHEN, A. I. H. C., NISHIMURA, E., and PAPADIMITRIOU, S., “The simultaneous berth and quay crane allocation problem,” *Transportation Research, Part E*, vol. 44, pp. 900–920, 2008.
- [12] CHRISTIANSEN, M., FAGERHOLT, K., and RONEN, D., “Ship routing and scheduling: status and perspectives,” *Transportation Science*, vol. 38, pp. 1–18, 2004.
- [13] CHRISTIANSEN, M. and NYGREEN, B., “A method for solving ship routing problems with inventory constraints,” *Annals of Operations Research*, vol. 81, pp. 357–378, 1998.

- [14] CORDEAU, J., LAPORTE, G., LEGATO, P., and MOCCIA, L., “Models and tabu search heuristics for the berth-allocation problem,” *Transportation Science*, vol. 39, pp. 526–538, 2005.
- [15] D. A. SCHRADY, G. K. S. and VASSIAN, R. B., “Predicting ship fuel consumption,” Tech. Rep. NPS-OR-96-007, Naval Postgraduate School, Monterey, California, 1996.
- [16] DAGANZO, C. F., “The crane scheduling problem,” *Transportation Research, Part B*, vol. 23B, pp. 159–175, 1989.
- [17] DAGANZO, C. F., “Crane productivity and ship delay in ports,” *Transportation Res Record*, vol. 1251, pp. 1–9, 1990.
- [18] DAI, J., LIN, W., MOORTHY, R., and TEO, C.-P., “Berth allocation planning optimization in container terminal,” *Working Paper, Georgia Institute of Technology, National University of Singapore*, 2004.
- [19] EDMOND, E. D. and MAGGS, R. P., “How useful are queue models in port investment decisions for container berths,” *Journal of the Operations Research Society*, vol. 29, pp. 741–750, 1978.
- [20] FAGERHOLT, K., “Optimal fleet design in a ship routing problem,” *International Transactions in Operational Research*, vol. 6, pp. 453–464, 1999.
- [21] GENDREAU, M., LAPORTE, G., and POTVIN, J.-Y., “Metaheuristics for the vehicle routing problem,” *Local Search Algorithms, J.K. Lenstra and E.H.L. Aarts (eds.)*, John Wiley Sons, Chichester, 1995.
- [22] GLOVER, F., “Future paths for integer programming and links to artificial intelligence,” *Computers and Operations Research*, vol. 5, pp. 533–549, 1986.
- [23] GLOVER, F. W. and LAGUNA, M., *Tabu Search*. Kluwer Academic Publishers, 1997.
- [24] GUAN, Y. and CHEUNG, R. K., “The berth allocation problem: models and solution methods,” *OR Spectrum*, vol. 26, pp. 75–92, 2004.
- [25] GUAN, Y., XIAO, W., CHEUNG, R., and LI, C., “A multiprocessor task scheduling model for berth allocation: heuristic and worst-case analysis,” *Operations Research Letters*, vol. 30, pp. 343–350, 2002.
- [26] HANJIN. <http://www.hanjin.com>. Accessed in July 2008.
- [27] IMAHORI, S., YAGIURA, M., and IBARAKI, T., “Local search algorithms for the rectangle packing problem with general spatial costs,” *Math Program*, vol. 97, pp. 543–569, 2003.
- [28] IMAI, A., NISHIMURA, E., and PAPADIMITRIOU, S., “Berth allocation with service priority,” *Transportation Research, Part B*, vol. 37, pp. 437–457, 2001.

- [29] IMAI, A., NISHIMURA, E., and PAPADIMITRIOU, S., "The dynamic berth allocation problem for a container port," *Transportation Research, Part B*, vol. 35, pp. 401–417, 2001.
- [30] IMAI, A., SUN, X., NISHIMURA, E., and PAPADIMITRIOU, S., "Berth allocation in a container port: using a continuous location space approach," *Transportation Research, Part B*, vol. 39, pp. 199–221, 2005.
- [31] KIM, K., LEE, K., and HWANG, H., "Sequencing delivery and receiving operations for yard cranes in port container terminals," *International Journal of Production Economics*, vol. 84, pp. 283–292, 2003.
- [32] KIM, K. and MOON, K., "Berth scheduling by simulated annealing," *Transportation Research, Part B*, vol. 37, pp. 541–560, 2003.
- [33] KIM, K. and PARK, Y., "A crane scheduling method for port container terminals," *European Journal of Operational Research*, vol. 156, pp. 752–768, 2004.
- [34] LAI, K. and SHIH, K., "A study of container berth allocation," *Journal of Advanced Transportation*, vol. 26, pp. 45–60, 1992.
- [35] LAPORTE, G. and OSMAN, I., "Metaheuristics in combinatorial optimization," *Annals of Operations Research*, vol. 60, 1995.
- [36] LAWRENCE, S. A., *International Sea Transport: The Years Ahead*. Lexington Books, Lexington, MA, 1972.
- [37] LEE, D.-H., WANG, H., and MIAO, L., "Quay crane scheduling with non-interference constraints in port container terminals," *Transportation Research E*, doi:10.1016/j.tre.2006.08.001, 2006.
- [38] LENSTRA, J., RINNOOY KAN, A., and BRUCKER, P., "Complexity of machine scheduling problems," *Annals of Discrete Mathematics*, vol. 1, pp. 343–362, 1977.
- [39] LI, C., CAI, X., and LEE, C., "Scheduling with multiple-job-on-one-processor pattern," *IIE Transactions*, vol. 30, pp. 443–445, 1998.
- [40] LIM, A., "The berth planning problem," *Operations Research Letters*, vol. 22, pp. 105–110, 1998.
- [41] LIM, A., RODRIGUES, B., and XIAO, F., "Approximation schemes for the crane scheduling problem," *In: Algorithm theory, SWAT 2004: Ninth Scandinavian workshop on algorithm theory, Humlebaek, July 8-10, Springer, Berlin*, pp. 323–335, 2004.
- [42] LIM, A., RODRIGUES, B., and XIAO, F., "Solving the crane scheduling problem using intelligent search schemes (extended abstract)," *In: Wallace M (ed) Principles and practice of constraint programming - proceedings of 10th international conference CP 2004; Toronto, September 27 - October 1, Springer, Berlin*, pp. 747–751, 2004.

- [43] LIM, A., RODRIGUES, B., XIAO, F., and ZHU, Y., "Crane scheduling with spatial constraints," *Naval Research Logistics*, vol. 51, pp. 386–406, 2004.
- [44] LIU, J., WAN, Y.-W., and WANG, L., "Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures," *Naval Research Logistics*, vol. 53, pp. 60–74, 2006.
- [45] LODI, A., MARTELLO, S., and VIGO, D., "Approximation algorithms for the oriented two-dimensional bin packing problem," *European Journal of Operational Research*, vol. 112, pp. 158–166, 1999.
- [46] MAERSK. <http://www.maersk.com>. Accessed in June 2008.
- [47] MCLELLAN, R., "Bigger vessels: how big is too big?," *Maritime Policy and Management*, vol. 24, pp. 193–211, 1997.
- [48] MOCCIA, L., CORDEAU, J.-F., GAUDIOSO, M., and LAPORTE, G., "A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal," *Naval Research Logistics*, vol. 53, pp. 45–59, 2006.
- [49] MOGGIA, L., *New Optimization Models and Algorithms for the Management of Maritime Container Terminals*. PhD thesis, Universita Degli Studi Della Calabria, 2004.
- [50] MURATA, H., FUJIYOSHI, K., NAKATAKE, S., and KAJITANI, Y., "Vlsi module placement based on rectangle packing by the sequence pair," *IEEE Trans Comput Aided Des Integrated Circuits Sys*, vol. 15, pp. 1518–1524, 1996.
- [51] NG, W., "Crane scheduling in container yards with inter-crane interference," *European Journal of Operational Research*, vol. 164, pp. 64–78, 2005.
- [52] NG, W. and MAK, K., "Yard crane scheduling in port container terminals," *Applied Mathematical Modelling*, vol. 29, pp. 263–276, 2005.
- [53] NG, W. and MAK, K., "Quay crane scheduling in container terminals," *Engineering Optimization*, vol. 38, pp. 723–737, 2006.
- [54] NISHIMURA, E., IMAI, A., and PAPADIMITRIOU, S., "Berth allocation planning in the public berth system by genetic algorithms," *European Journal of Operations Research*, vol. 131, pp. 282–292, 2001.
- [55] OOCL. <http://www.oocl.com>. Accessed in June 2008.
- [56] OSMAN, I. H. and KELLY, J. P., *Metaheuristics: Theory and Applications*. Kluwer Academic Publishers, Norwell, MA, 1996.
- [57] PARK, K. and KIM, K., "Berth scheduling for container terminals by using a sub-gradient optimization technique," *Journal of the Operational Research Society*, vol. 53, pp. 1054–1062, 2002.

- [58] PARK, Y. and KIM, K., "A scheduling method for berth and quay cranes," *OR Spectrum*, vol. 25, pp. 1–23, 2003.
- [59] PETERKOFKY, R. and DAGANZO, C., "A branch and bound solution method for the crane scheduling problem," *Transportation Research, Part B*, vol. 24B, pp. 159–172, 1990.
- [60] PINEDO, M., *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, 1996.
- [61] RANA, K. and VICKSON, R., "Routing container ships using lagrangean relaxation and decomposition," *Transportation Science*, vol. 25, pp. 201–214, 1991.
- [62] RODRIGUE, J. P., "Six generations of containerships." <http://people.hofstra.edu/geotrans/eng/ch3en/conc3en/containerships.html>. Accessed in May 2008.
- [63] RONEN, D., "The effect of oil price on the optimal speed of ships," *The Journal of the Operational Research Society*, vol. 33, pp. 1035–1040, 1982.
- [64] RONEN, D., "Cargo ships routing and scheduling: Survey of models and problems," *European Journal of Operational Research*, vol. 12, pp. 119–126, 1983.
- [65] RONEN, D., "Ship scheduling: The last decade," *European Journal of Operational Research*, vol. 71, pp. 325–333, 1993.
- [66] STAHLBOCK, R. and VOSS, S., "Operations research at container terminals: a literature update," *OR Spectrum*, vol. 30, pp. 1–52, 2008.
- [67] STEENKEN, D., VOSS, S., and STAHLBOCK, R., "Container terminal operation and operations research - a classification and literature review," *OR Spectrum*, vol. 26, pp. 3–49, 2004.
- [68] STERLING, F. W., *Marine Engineer's Handbook*. 239 West 39th Street, New York: McGraw-Hill, 1920.
- [69] STOPFORD, M., "Is the drive for ever bigger containerships irresistible?" MD Clarkson Research, Lloyds List Shipping Forecasting Conference, April 2002.
- [70] TOTH, P. and VIGO, D., "The granular tabu search and its application to the vehicle routing problem," *Inform's Journal on Computing*, vol. 15, pp. 333–346, 2003.
- [71] VIS, I. and DE KOSTER, R., "Transshipment of containers at a container terminal: An overview," *European Journal of Operational Research*, vol. 147, pp. 1–16, 2003.
- [72] WANG, F. and LIM, A., "A stochastic beam search for the berth allocation problem," *Decision Support Systems*, vol. 42, pp. 2186–2196, 2007.

- [73] YANG, C. H., “The impact of bigger vessels on shipping and ports.” Shipping, Logistics and Port Research Center, Korea Maritime Institute, 2004.
- [74] ZHU, Y. and LIM, A., “Crane scheduling with spatial constraints: mathematical model and solving approaches,” *In: AIM 30-2004, eight international symposium on artificial intelligence and mathematics, Fort Lauderdale, January 4-6, 2004.*
- [75] ZHU, Y. and LIM, A., “Crane scheduling with non-crossing constraint,” *Journal of the Operational Research Society*, vol. 57, pp. 1464–1471, 2006.

VITA

Aykagan Ak received his BS degree in Industrial Engineering from Middle East Technical University in 2004. He joined the Ph.D program in Industrial Engineering with Manufacturing and Logistics track in the H. Milton Stewart School of Industrial and Systems Engineering at Georgia Institute of Technology in the same year. He received his MS degree in Industrial Engineering in 2006. During his Ph.D. study he worked with Dr. Alan L. Erera. He focused on alternative recourse strategies for stochastic vehicle routing problem and tactical and operational level scheduling problems at seaports. Aykagan's research interests are in mathematical programming and meta-heuristic algorithms for hard problems. He joined the science team at Manhattan Associates in August 2008.