MODELS AND NEW METHODS FOR QUAYSIDE OPERATIONS IN PORT CONTAINER TERMINALS

CHEN JIANG HANG

(B. Eng. Tsinghua University)

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

DEPARTMENT OF CIVIL & ENVIRONMENTAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2011

Acknowledgements

This thesis is the culmination of input, work and encouragement of many people who have helped and accompanied me for the four years that I have spent at National University of Singapore. First of all, my deepest gratitude goes to my family, my parents and my brother, for their endless love and unconditional support.

Many thanks to my PhD supervisor, Prof. *Lee Der-Horng*, for steering me through and pushing me further than I can imagine. His eloquence, sharpness, and great sense of humor have impressed me deeply and made my every meeting and discussion with him always full of joy and unforgettable memories.

I would also like to express deepest appreciation to my module teachers, Prof. *Phoon Kok Kwang*, Prof. *Meng Qiang*, Prof. *Ong Say Leong*, Prof. *Fwa Tien Fang*, Prof. *Chan Weng Tat*, Ms. *Susan Lopez-Nerney*, and Ms. *Ho Laina* for expanding my knowledge, extending my vision, and sharing me with their academic insights.

One of the best parts of attending NUS has been the wonderful opportunity to be surrounded by the many interesting, intellectually stimulating and impressive fellow students. The list includes: Dr. *Wang Huiqiu*, Dr. *Dong Meng*, Dr. *Huang Yikai*, Dr. *Khoo Hooi Ling*, *Qi Jin*, *Wang Xinchang*, *Wang Tingsong*, *Liu Zhiyuan*, *Weng Jinxian*, *Zhang Jian*, *Long Zhuoyu*, *Wu Xian*, *Ma Yaowen*, *Qu Xiaobo*, *Qu Fenghua*, *Yang Jiasheng*, *Wang Qing*, *Zheng Yanding*, *Jin Jiangang*, *Fu Yingfei*, *Zhang Yang*, *Huang Sixuan*. Specially thanks to Dr. *Cao Jinxin*, for his guidance, constructive suggestion for my research. Special appreciation also goes to Dr. *Zhang Xu* who let me know there is a wild and wonderful world out there beside the academic research.

A huge debt of gratitude is extended to my dear girl friend *Shen Yuanyuan* and other best friends in Singapore *Zheng Xiaolian*, *Yan Sen*, *Li Ti*, *Hannah Dai*, *He Xuan*, *Dong Xiangxu*, *Zhang Xiangjing*, *Tian Bo*, Dr. *Cui Rongxin*, Dr. *How Yoon Ee*, *Wah Yi Feng*, *Yap Kim Thow*, *Yeoh Ker-wei* who make my personal life colorful and shining.

Finally, I would like to thank the National University of Singapore for the President's Graduate Fellowship and all the serves of the university staffs particularly to Mr. *Foo Chee Kiong*, Ms. *Yap-Chong Wei Leng*, Ms. *Theresa Yu-Ng Chin Hoe*.

Table of Contents

Executive Summary

The swift pace of globalization has significantly increased the demand for containerized maritime transport services. Under the atmosphere, the competition among port container terminals has become acute and such drives the managers in port container terminals to pursue seamless flows of containers through terminals while to keep the operational costs as low as possible. To this end, operations research methods have received considerable importance for the operations management in port container terminals. From the angle of operations research and management science, this thesis aims to design models and devise the corresponding solving methods for the quayside operations in port container terminals. This enables port managers to come up with viable and cost-effective scheduling plans for quayside operation problem in a rapid manner.

The mathematical models introduced in this thesis have well covered each key component of the quayside operation problem, i.e., berth allocation problem, quay crane assignment problem, and quay crane scheduling problem. There are two highlights in this thesis with regarding to the aimed modeling issue: 1) the technology updates and innovative implementations occurred in the field have been reflected. For instance, the quay crane scheduling problem has been extended to the environment of indented berth, which is a noble idea to increase the ship-to-shore interface aiming to tackle the challenge raised by the emergence of more and more mega-containerships. 2) the integration issues to synchronize the decision-making processes for each key component of the quayside operation problem has been stressed. In this thesis, integrated models (including both discrete-berth and continuous-berth versions) to embrace all the information flow within the system of the quayside operation have been developed.

On the other hand, in the perspective of solving approaches and algorithms, a spectrum of methods to handle the proposed models has been devised. This thesis not only contains the meta-heuristics and approximation algorithms which are developed to generate sub-optimal solutions for the hard-to-solved problems, but also introduces and tailors some of the excellent methods proposed in the field of operation research and computer science to obtain exact solutions. Taking the berth allocation problem as an example, the devised Greedy Randomized

Adaptive Search Procedure (GRASP) outperforms the state-of-the-art algorithms appeared in the literature. Additionally, the GRASP algorithm possesses more flexibility than other methods, making it much easier to incorporate the decision rules of port managers into the planning procedure. Moreover, in the path to solve the integrated models for the quayside operation problem, after identifying the special properties of the problems at hand, the exact methods like Combinatorial Benders' Cuts algorithm and Local Branching method have been developed and tested to be promising methods for the complex problems through a series of comprehensive numerical experiments.

In summary, the research presented in this thesis provides new insights of modeling the quayside operation problem in port container terminals and introduces a set of potent tools to handle the challenging issues rising from this field.

List of Tables

List of Figures

Abbreviations and Symbols

Abbreviations

TEU	Twenty-foot Equivalent Unit
BAP	Berth Allocation Problem
QCAP	Quay Crane Assignment Problem
QCSP	Quay Crane Scheduling Problem
MIP	Mixed Integer Programming
B&B	Branch and Bound
B&C	Branch and Cut
GA	Genetic Algorithm
SA	Simulated Annealing Algorithm
DP	Dynamic Programming
GRASP	Greedy Randomized Adaptive Search Procedure
SBS	Stochastic Beam Search
BP	Best Partition
EBP	Enhanced Best Partition
SSH	Scheduling Subproblem Heuristic
ASH1	Assigning Subproblem Heuristic 1
ASH2	Assigning Subproblem Heuristic 2
DAG	Directed Acyclic Graph
CBC	Combinatorial Benders' Cuts
MIS	Minimal Infeasible Subsystem
LB	Local Branching

Commonly Used Symbols

V	the set of incoming container vessels
B	the set of discrete berths
Q	the set of quay cranes
Ω	the set of tasks
M	a sufficiently large positive constant number.
$c_{\max}$	makespan of a schedule

Chapter 1

Introduction

1.1 Research Background

As the trend of globalization inevitably moves forward, world trade has risen significantly during the past decades. Since over 80% of world merchandize trade by volume is carried by sea, the world seaborne trade has also achieved a rapid and vast growth. Figure 1.1 reveals the evolving relationship between the growth in world GDP and seaborne trade from 1990 to 2010, according to the report of United Nations Conference on Trade and Development (UNCTAD, 2008). Obviously, before year 2008, the booming of the world seaborne trade was remarkable. It is reported that in 2007, international seaborne trade was estimated at 8.02 billion tons of goods loaded. However, following the global financial crisis of late 2008, the year 2009 recorded the first and deepest drop in global output since the 1930s, with world GDP contracting by 1.9%. Correspondingly, in year 2009, world seaborne trade declined sharply and its volume fell by 4.5%. Thanks to the large scale of the fiscal stimulus package launched by the world's leading developed countries and the rise of the emerging countries such as China and India, a global recovery is under way and despite a number of challenges ahead, it is optimistical to expect a bright perspective of the world seaborne trade market in the near future.

The globalization of trade has also fuelled the strong demand of maritime transport services particularly for containerized freight transportation, which was started in the late 1950s. Before containerization, most general cargo was handled by building pallets and loading them into the

holds of vessels using cranes on the vessel 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. It is estimated that since 1990, container trade has increased by a factor of five. According to Drewry Shipping Consultants, container trade is forecast to double by 2016 to reach 287 million TEUs (Twenty-foot Equivalent Unit), and more than double by 2020 to exceed 371 million TEUs.

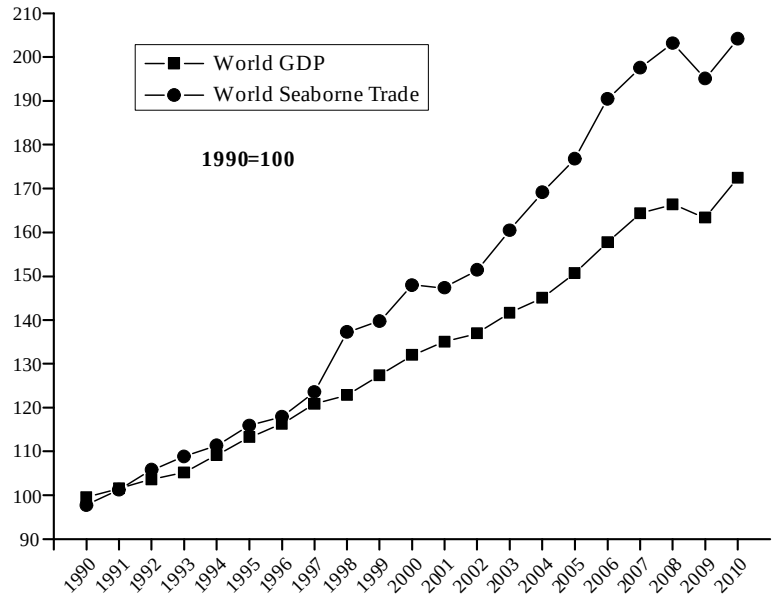


Figure 1.1: Indices for world economic growth and world seaborne trade (UNCTAD, 2008)

As the gateways in global containerized maritime shipping network, port container terminals play essential roles as interfaces between maritime shipping and land transportation. Spurred by container trade growth, port container handling activity has also increased. For example, it is reported that the share of transshipment in total port throughput has grown from 10% in 1980 to 27% in 2007. As a result, competition to attract more ship carriers among port container terminals has become acute especially for the geographically close container transshipment ports such as the ports of Singapore and the Tanjung Pelepas Port of Malaysia. Facing fierce competition, managers of many port container terminals have tried to provide more intensive logistic services while reducing costs by utilizing efficiently their existing resources, such as human resources, berths, container yards, and various container handling equipments. The

looming challenges for port container terminals have triggered the need for in-depth studies on the effective management of port container terminals, which includes the issues of quayside operations and internal logistics as well as landside operations, transport connection and routing within the surrounding area.

1.2 Port Container Terminals and Quayside Operations

In general terms, the container handling activities in a typical port container terminal can be classified into three groups in accordance with the area where the operations are taken place: quayside operations, yard-side operations, and hinterland operations. As depicted in Figure 1.2, after its arrival at a port container terminal, a container vessel is allocated to a berth equipped with quay cranes to load and unload containers. The unloaded import containers are transported to the yard area by yard trucks and stacked in the designated slot by yard cranes. The procedure of loading a container to a container vessel is done in a reverse manner. Obviously, the decision making process to come up with a good quayside operations planning is extremely critical for port operators due to the fact that the decision made for quayside operations lays the foundation for the subsequent phases of planning and scheduling in other areas of a port container terminal.

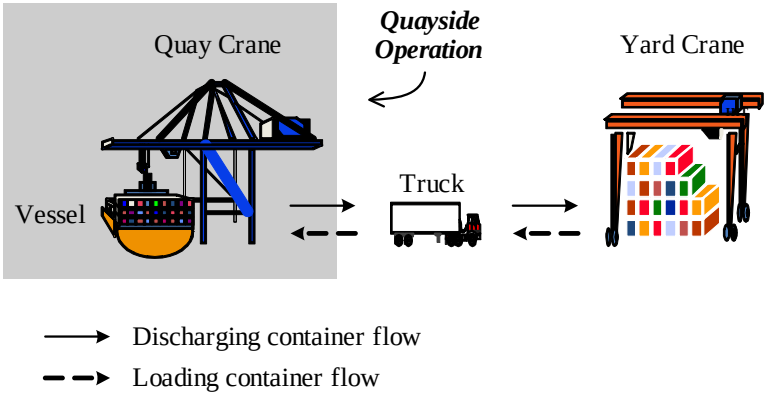


Figure 1.2: Schema for container flow in a port container terminal

There are two scarce resources that need to be utilized with care along the quay of port container terminals, i.e., berth and quay cranes. The berth resource corresponds to a linear

stretch of space in the port container terminal where vessels are able to moor. Quay cranes are industry-standard equipment for loading and discharging containers to and from vessels. Quay cranes are very expensive (around 10 million Singapore dollars for individual machine) and quay cranes along the same berth are mounted on the same tracks, which forbid them from crossing each other at any instant. Traditionally, there are three key problems needed to be addressed for quayside operations: the Berth Allocation Problem (BAP), the Quay Crane Assignment Problem (QCAP), and the Quay Crane Scheduling Problem (QCSP). The BAP is the problem to determine the berthing times and berthing locations for a set of vessels within the planning horizon by considering some factors such as the length, the expected arrival time, and the processing time of each vessel. The QCAP, also referred to as the crane split problem, is the decision to allocate quay cranes to container vessels with respect to the constraints of quay crane availability and accessibility. For the QCSP, given the number of quay cranes deployed to a certain vessel, the target of a QCSP is to assign each task in the vessel to a quay crane and meanwhile to determine the starting times of the process for all the tasks that have been assigned to a particular quay crane. Compared with the traditional machine scheduling problems, the QCSP is unique in the sense that the potential of physical interference between any two quay cranes should be avoided.

1.3 Research Objectives, Scopes, and Organization of the Thesis

The thesis presents a comprehensive study on how to formulate the models and to develop the corresponding solving algorithms for the problems arising from quayside operations in a typical port container terminal. It will deliver the following outputs:

- Devise the state-of-the-art heuristic and sophisticated exact solving algorithms for the existing models for the problems of quayside operations presented in the literature (see the works in Chapter 3 and Chapter 6);
- Examine the weaknesses appeared in the existing models for the problems of quayside

operations and provide the revised model and new algorithms for solution (see the work in Chapter 4);

- Extend the existing models to cover the latest updates and/or innovations occurred in the field of quayside operations and develop a more generic solution method (see the work in Chapter 5);
- Propose new models (integration models) and efficient algorithms to synchronize the information flow for quayside operations (see the works in Chapter 7 and Chapter 8).

Note that although the QCAP is one of the key decisions for quayside operation. However, as pointed out by Bierwirth and Meisel (2010), "in practice, the QCAP is not found as a difficult problem if solved by rules of thumb. Therefore, the problem has hardly received attention by its own in academic research. Due to the profound impact on vessels' handling times, however, crane assignment decisions are involved in some advanced berth planning models." In light of this, in this thesis, there is no intension to provide in-depth studies on the QCAP alone. On the contrary, the QCAP will only be involved when integration models for quayside operations are considered. Besides, all the research works presented in this thesis belong to deterministic models. To cope with uncertainty, several concerns and suggestions are highlighted in the future research remark in Chapter 9.

The thesis consists of 9 chapters. Figure 1.3 provides an overview for the structure for this thesis.

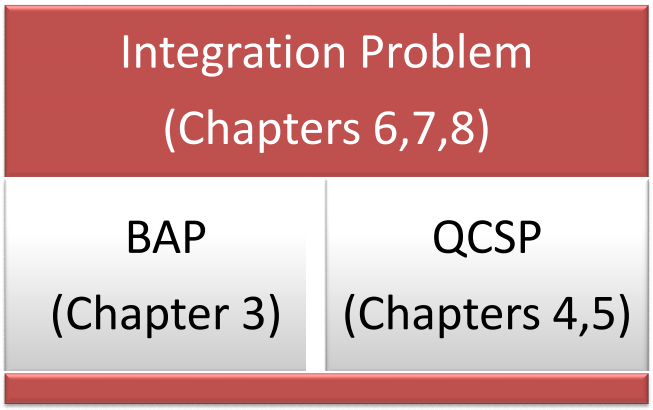


Figure 1.3: Overview for the PhD works

Chapter 1 is the introductive chapter which provides the general research background, the brief introduction on quayside operations, and the objectives and scope of the thesis.

Chapter 2 reviews the literature for previous studies. The literature review is divided into three parts: review on the BAP, the QCAP & the QCSP, and the integration models for quayside operations.

Chapter 3 describes a study on the continuous BAP and two versions of Greedy Randomized Adaptive Search Procedure (GRASP) are developed for solutions.

Chapter 4 proposes an improved approximation algorithm to solve the QCSP by considering the Non-crossing Constraints.

Chapter 5 discusses the QCSP at indented berth, in which the QCSP is treated as an Unrelated Machine Scheduling Problem.

Chapter 6 examines the integration framework proposed in Liu et al. (2006). A new method called Combinatorial Benders' Cuts algorithm is developed to solve the Berth-level Model in the framework.

Chapter 7 introduces a new integration model (with respect to discrete berths) for the problem of quayside operations.

Chapter 8 aims to develop an integrated framework (with respect to continuous berths) to consolidate all the key information from all aspects of the problem of quayside operations.

Chapter 9 draws the concluding remarks and presents the future research suggestions under this research topic.

Chapter 2

Literature Review

2.1 Literature on the BAP

There have been several studies on how to allocate berth to incoming vessels in the literature. Basically, the BAP can be categorized into two classes: discrete BAP and continuous BAP. For discrete version, the quay is viewed as a finite set of berths. Usually, one berth can only serve one vessel at a time. On the contrary, continuous BAP model allows vessels to berth anywhere along the quay so as to sufficiently utilize the quay resource. Another criterion to classify the BAP is based on whether all the vessels have arrived at the terminal or not before the berthing planning starts. If all the vessels have already arrived, this type of BAP is identified as static BAP. Otherwise, it will be considered as dynamic BAP.

In Brown et al. (1994), a ship berthing plan assigning surface naval ships a berth prior to entering port or reassigning ships once in port, was carried out. An MIP was formulated to maximize the ship-to-berth assignment benefits and solved by GAMS. In Brown et al. (1997), the authors tried to seek an optimal berthing plan for submarines by considering the shore power limitation and specific service facilities for each submarine etc. To deal with change of berthing plan, a persistence incentive was incorporated into the model and helped to minimize revisions to an approved plan. For the studies of Brown et al., however, are only suitable for naval ports since for commercial ports, parallel mooring and shifting of vessels are rare.

In Imai et al. (1997), discrete static BAP was formulated to minimize the sum of port stay-

ing times of vessels and the dissatisfaction of the vessels in terms of the berthing order. A weighted method was proposed to identify a set of non-inferior solutions. Unlike Imai et al. (1997), in Imai et al. (2001), a dynamic version BAP was introduced. In the study, the arrival time of some vessels was known in advance, but could be later than the beginning of the planning horizon. Incorporating this consideration into the BAP complicates the model but will achieve more practicable solutions for port operators. To solve the problem, a Lagrangian relaxation was proposed. As a related work, in Monaco and Sammarra (2007), a stronger formulation for the BAP introduced by Imai et al. (2001) was discussed. The compact formulation of the BAP, was solved by Lagrangian relaxation with a non-standard multiplier adjustment method. The reported numerical results in the study shown the effectiveness and efficiency of the proposed algorithm. In Nishimura et al. (2001), dynamic BAP with water draft constraints and consideration of multiple vessels in a berth at a time, was modeled as a non-linear integer programming problem. Due to the computational intractability, the entire problem was divided into several sub-problems and Genetic Algorithm (GA) was employed sequentially to each subproblem. For Imai et al. (2003), service priority issue in BAP was highlighted. The authors provided a generic formulation for BAP with service priority by introducing weight for each incoming vessels, where the weight could be measured by vessel size, handling volume, etc. Subgradient method with a Lagrangian relaxation technique was firstly examined. However, because of the problem complexity, a more efficient GA (as the one used in Nishimura et al. (2001)) was applied. A continuous BAP was considered in Imai et al. (2005), which was solved by a heuristic. To obtain a continuous BAP scheme, at the first stage of the heuristic, discrete BAP could be employed to provide a good guide (upper and lower bounds for the continuous counterpart). The second part was crucial for the proposed algorithm since it relocated the vessels to produce a valid berthing plan. In Imai et al. (2007), BAP was extended to indented berths for mega-containerships. At first, the authors rewrote the formulation of the problem proposed in Nishimura et al. (2001), modifying the original non-linear integer programming formulation to a linear integer programming one. However, at most two vessels were allowed to be served at the same berth simultaneously. Then, mega-containership was added into the model and assigned the highest priority which guaranteed that mega-containership would be served

without any delay. When no more mega-containerships would be serviced, the indented berths could be used to handle feeder ships. A corresponding model was also formulated to capture this situation. A GA-based heuristic was used to find the near optimal solutions. In Imai et al. (2008b), the authors examined the BAP under exceptional circumstances that due to limited quay capacity, some vessels with longer expected waiting time for service would be allocated to an external terminal. Both static and dynamic versions of the problem were examined and solved by GA.

In Li et al. (1998), BAP was treated as a machine scheduling problem with "multiple-job-on-one-processor" pattern. Herein, the processor is the berth while the multiple jobs are the vessels. An important constraint for the scheduling problem was that the total size of the jobs being processed could not excel the capacity of the processor (the length of the berth) at any time. Cases of "fixed location" and "non-fixed location" for BAP were studied. For each case, the heuristic based on First-Fit-Decreasing rule was developed. Meanwhile, the worst-case behavior for the heuristics was analyzed as well. For the rolling horizon situation, the problem with processor unavailability was addressed. There are two main limitations of the study: firstly, the arrival times of each vessels are not taken into account and secondly, the "agreeable" assumption (i.e., larger vessel will require more work) is not always valid in reality. In Guan et al. (2002), BAP was modeled as a multiprocessor task scheduling problem. The objective was to minimize the sum of weighted completion time for all vessels. A greedy heuristic was developed and the worst case analysis was also carried out for the proposed method. In Guan and Cheung (2004), the authors proposed a heuristic to handle continuous BAP for batch arriving vessels. At first, the greedy heuristic in Guan et al. (2002) was employed to allocate vessels in the same batch (i.e., a group of vessels with the same arrival time). Thereafter, between any two consecutive batches, the exchange of vessels was performed as long as the exchange could achieve a better objective value. Finally, an exact tree search procedure was applied for each of the resulting batches.

In Lim (1998), the continuous BAP was represented by a graph (with directed and undirected edges) and transformed into a problem of setting the direction of undirected edges in the graph such that the graph became directed acyclic and the longest path in the graph was minimized.

A heuristic was designed for fixing the edge direction in the graph representation of BAP to create a directed acyclic graph with the minimum longest path. One drawback of the study is that the beginning time of berthing for a vessel should always equal to its arrival time; therefore the delay of berthing for the vessels is not allowed. In Tong et al. (1999), an Ant Colony Optimization was adopted for the BAP introduced in Lim (1998), by representing the solution of BAP as a vertex permutation. For Wang and Lim (2007), a Stochastic Beam Search (SBS) was proposed for continuous BAP. The procedure was divided into N levels, where N was the number of incoming vessels. At each level, there were 3 node sets—R, U, and B. Set R was the node set generated by Set B in previous level; Set U was the node set generated from Set R by draft selection; Set B was the node set generated from Set U by detail selection. For selection scheme, the authors proposed the stochastic selection. In their numerical experiments, the SBS algorithm outperformed Simulated Annealing (SA) proposed in Dai et al. (2007) for continuous BAP.

Park and Kim (2002) addressed a continuous BAP by the Lagrangian relaxation technique. Unlike the works of Imai et al, the objective of the study was to minimize two kinds of costs: the costs resulting from delayed departures of vessels and the additional handling costs resulting from deviations of the berthing position from the favorite location on the berth. Medium sized numerical experiments (less than 20 vessels) were tested and showed the applicability of the proposed method. Kim and Moon (2003) tackled the same problem as the one in Park and Kim (2002). The method applied was SA. To decode a sequence of vessels (the coding scheme of one feasible solution in SA) into a solution, the properties of optimal solution for continuous BAP was examined by reducing the BAP to a sum minimization problem of a single facility under the rectilinear distance measure.

In Cordeau et al. (2005), a dynamic discrete BAP was formulated as a Multi-Depot Vehicle Routing Problem with Time Windows. Compared with the formulation in Imai et al. (2001), the formulation proposed by Cordeau et al. (2005) for BAP possessed more flexibility since it could easily accommodate a weighted sum of the vessel service times and time windows. A time based Tabu search was used to solve the BAP. To extend the work into a continuous case, an adjusted segment length scheme was proposed based on the vessel length distribution in

Medcenter Container Terminal.

In Dai et al. (2007), a berthing plan was encoded by a sequence pair (H,V). With a fixed sequence pair, BAP was separated into two independent subproblems, namely time cost minimization subproblem and space cost minimization subproblem. To obtain the near optimal solution, an SA algorithm was used to search through the space of all possible (H,V) sequence pairs. In Moorthy and Teo (2006), the decomposition idea for BAP in Dai et al. (2007) was adopted to generate a robust berth management plan. Based on the conclusion in the time cost minimization subproblem in Dai et al. (2007), the estimation of the expected delays was analyzed and used to offer a robust BAP solution compared with the deterministic version.

In Hansen et al. (2008), a Minimum Cost Berth Allocation Problem was modeled and solved by the Variable Neighborhood Descent heuristic.

In Lee and Chen (2009), a candidate-based approach was developed to handle BAP by incorporating some different considerations such as vessel shifting and the clearance distance between vessels which depended on the order of berthed vessels. A three-stage heuristic was proposed to solve the BAP.

Finally, it is noted that apart from the aforementioned analytical models for BAP, simulation techniques have also been applied to handle the problem, such as the works by Legato and Mazza (2001) and Henesey et al. (2004).

2.2 Literature on the QCAP and the QCSP

Over the decades, the QCSP has received great attention. In Daganzo (1989), static and dynamic quay crane scheduling problems for multiple container vessels were studied. The objective was to serve all these container vessels, while minimizing their aggregate cost of delay. Further, Peterkofsky and Daganzo (1990) developed a Branch and Bound (B&B) method for the static quay crane scheduling problem. However, in both studies, the berth length was assumed to be unlimited and especially, the Non-crossing Constraints between quay cranes were not considered, which may cause quay cranes unrealistically cross over each other during the the operations.

In Lim et al. (2004a), the authors examined the QCSP with spatial and separation constraints for a single container vessel. The problem was represented by a bipartite graph matching problem by defining quay cranes and bay jobs as vertices and crane-to-job throughput as the weights of connecting edges. DP algorithms, a Tabu search, and a Squeaky Wheel Optimization Heuristic were proposed for solving the QCSP with spatial constraints. Nevertheless, the study was suffered by the deficiencies such as: the definition of the profit value associated with a crane-to-job assignment and the lack of time component in the model. Lim et al. (2004b) analyzed the properties of the QCSP with Non-crossing Constraints for a single vessel. Based on the study, a full polynomial time approximation scheme was proposed to solve the QCSP with Non-crossing Constraints. Besides, three approximation algorithms were also provided in this study. In Zhu and Lim (2006), the study on the QCSP with Non-crossing Constraints was also conducted. In the study, the authors assumed that vessel could be divided into holds and that quay cranes could move from hold to hold but jobs were not pre-emptive. The problem was formulated as an MIP problem and solved by a B&B algorithm and an SA heuristic. As a summary work of previous studies, Lim et al. (2007) presented the numerical experiment results by applying the proposed algorithms for the QCSP with Non-crossing Constraints. In the paper, a backtracking algorithm was introduced which was inspired by the study in Lim et al. (2004b).

Kim and Park (2004) addressed a QCSP for single container vessel by formulating the problem as an MIP. In this model, the authors considered the Non-crossing and the Precedence Constraints for the QCSP. A B&B method and a GRASP heuristic were proposed for the solution. As a related work, Moccia et al. (2006) examined the problem formulation structure of Kim and Park (2004) and managed to ameliorate the mathematic model. Meanwhile, a more efficient B&C algorithm was developed. Based on Moccia et al. (2006), Sammarra et al. (2007) developed a Tabu search heuristic for the same problem. The QCSP was decomposed into two-level problems: routing and scheduling problems. Given the feasible results of upper problem (routing problem), the scheduling problem was equivalent to the searching of the longest path in a disjunctive graph constructed by considering the precedence and non-simultaneity constraints. Instead of enumeration, a local search approach was used to find the near-optimal orientation

of the disjunctive graph. The upper level problem was solved by a Tabu search heuristic. Two neighborhood generation operators—swapping adjacent tasks and insertion, were discussed. The effectiveness of the proposed heuristic was assessed by comparison with the B&C algorithm in (Moccia et al., 2006) and GRASP in (Kim and Park, 2004). Bierwirth and Meisel (2009) designed a heuristic to solve QCSP in a unidirectional manner. Similar to Sammarra et al. (2007), the idea to decompose the QCSP into easier subproblems (routing and scheduling) was also applied by Bierwirth and Meisel (2009). In contrast to Sammarra et al. (2007), in its routing subproblem, the sequences of tasks assigned to all quay cranes should be ordered by their stored locations in either an ascending or a descending way.

Ng and Mak (2006) discussed the problem of scheduling identical quay cranes moving along a common linear rail to handle containers for a vessel. Compared with the works of Lim et al., the uniqueness of this study was that the jobs in each bay were distinguished into discharging jobs and loading jobs, which should follow the precedence relationship. Due to the intractability of the problem, the authors suggested a scheduling heuristic. The philosophy of this heuristic was quite similar to the idea of approximation algorithms in Lim et al. (2004b). The difference is that in Ng and Mak (2006), each subproblem was equivalent to a single quay crane to process jobs with sequence-dependent process times while in Lim et al. (2004b), the jobs handled by a single quay crane were sequence-independent.

In Lee et al. (2008), an MIP model for the QCSP with Non-crossing Constraints was studied. The authors discussed the computational complexity of the QCSP with Non-crossing Constraints and they drew the conclusion that the problem was naturally $\mathcal{NP}$-complete. And for the solution, a GA was proposed for near optimal results.

For Tavakkoli-Moghaddam et al. (2009), this work was an extension of Kim and Park (2004). Although several vessels were taken into account, like Daganzo (1989), all the vessels had already arrived at the harbor and berthed along the quay side. Therefore, to some extent, the study also focused on a single "vessel" consisting of several vessels.

2.3 Literature on the Integration Models

The three problems related to the quayside operations are highly interrelated based on the observation that the output of the BAP is the key input for the QCAP and the QCSP, while after solving the QCAP and the QCSP the processing times for all the vessels would be updated to more realistic values and this triggers another round of problem solving for the BAP (see Figure 2.1). However, to deal with a complex system like a port container terminal, the conventional approach to handle the quayside operation problem is to solve the BAP, the QCAP, and the QCSP in a sequential manner. Such a decomposition method has disadvantages since the optimization of individual subproblems does not guarantee the overall efficiency of container terminal operations; besides, solving the quayside operation problem by decomposition method may generate infeasible solutions and several runs of time-consuming adjustment should be executed to seek a feasible plan.

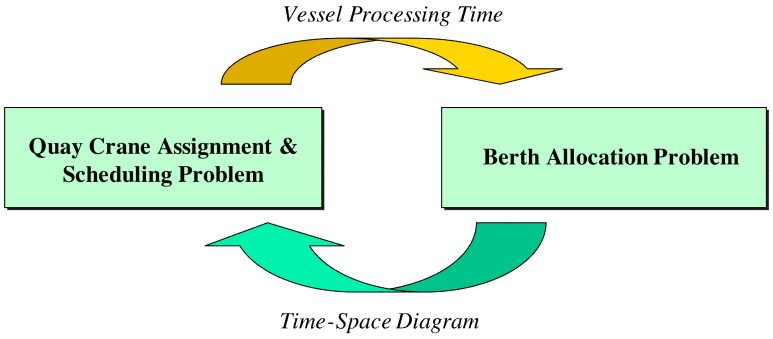


Figure 2.1: Relationship between the BAP and the QCAP & QCSP

Thereby, due to such a deficiency, in recent years, more and more researchers have been aware that improved terminal performance cannot be achieved by solving isolated problems alone but by pursuing the integration model to embrace all key components. For quayside operations, the pioneer work for integration can date back to Park and Kim (2003). The proposed method in Park and Kim (2003) was two-phased. In the first phase, an integer programming model was formulated to obtain the berth allocation plan and also the total number of quay cranes that should be assigned to each vessel. Subsequently, in the second phase, the detailed assignment for individual quay crane was solved by a DP method to match the information of the total

number of quay cranes assigned to each vessel realized in the previous phase. To facilitate the modeling in the first phase, the authors simplified the problem by assuming that the duration of berthing of a vessel was inversely proportional to the total number of quay cranes assigned to the vessel. However, the linearity assumption was over-simplified and did not agree with practice.

Liu et al. (2006) proposed a two-leveled framework to integrate quayside operations. The plausible idea presented in Liu et al. (2006) was that instead of assuming a function relationship between the processing time of a vessel and the number of quay cranes assigned to it, the authors introduced a series of parameters p_{vj}, i.e., the processing time of vessel j when v quay cranes were assigned to it, to explicitly represent the complex relationship. For each v and j, the value of p_{vj} was the optimal value of the Vessel-level Model, which was essentially a QCSP. Therefore, by parameterizing p_{vj}, the QCSP could be detached from the integration model for the problem of quayside operations and only the BAP and the QCAP needed to be combined and they were solved seamlessly in the Berth-level Model. However, the integration model presented in Liu et al. (2006) needed further improvement since in the Berth-level Model, the information of berth positions for all vessels was fixed and initialized by the output of an upper decision making problem.

Imai et al. (2008a) developed a simultaneous berth and quay crane allocation model. In this work, the authors assumed that "ship handling requires a specific number of cranes and it does not begin till that number of cranes are available". However, compared with industrial practices, this assumption was not pragmatic. As identified by the authors, such a methodology could only provide an upper bound for the genuine integration model for the problem of quayside operations.

Meisel and Bierwirth (2009) devised a new model formulation to address the deficiency of the study in Park and Kim (2003) by substituting the linearity assumption with a more realistic quay crane resource utilization function. In the resource utilization function, the authors used a predefined interference exponent α to reflect the impact of interference of quay cranes when they were all assigned to one vessel. Specifically, the productivity obtained from assigning v cranes to a vessel for one hour was measured by a total of v^{α} quay crane hours. Although

compared with Park and Kim (2003), this model presented in Meisel and Bierwirth (2009) had gained more advanced features, the approach might not be able to capture the real magnitude of interference of quay cranes when they were assigned to the same vessel.

For other works on the integration model for the problem of quayside operations, see the discussions and references in Bierwirth and Meisel (2010).

Chapter 3

Greedy Randomized Adaptive Search Procedure for Continuous Berth Allocation Problem

3.1 Background

To handle the continuous dynamic BAP, in the literature, several heuristics have been employed such as Ant Colony Optimization (Tong et al., 1999), SA (Kim and Moon, 2003; Dai et al., 2007), Tabu search (Cordeau et al., 2005), and SBS (Wang and Lim, 2007). According to the numerical experiments in Wang and Lim (2007), SBS outperforms the current state-of-the-art meta-heuristic such as the SA proposed in Dai et al. (2007). As an alternative, GRASP is developed in this study to handle the continuous dynamic BAP and its effectiveness will be assessed by comparisons with CPLEX and SBS. Additionally, one of the missing issues on how to identify the possible locations in the Time-space Diagram where next vessel can be placed existed in (Guan and Cheung, 2004) and (Wang and Lim, 2007), is addressed. Instead of exhaustive enumeration, a more efficient method is proposed.

3.2 Problem Description

For the continuous dynamic BAP, the mathematical formulation in Guan and Cheung (2004) is followed. In this formulation, vessel shifting is not considered since the interruption of the container handling is usually costly.

3.2.1 Parameters

- S, the length of the continuous berth
- T, the length of the planning horizon
- n, the number of vessels, $n = |V|$
- p_i, the processing time for Vessel i, $i \in V$
- s_i, the size of Vessel i, $i \in V$
- a_i, the arrival time of Vessel i, $i \in V$
- w_i, the weight assigned for Vessel i, $i \in V$

It is noted that s_i has included the requested gap between the adjacent vessels. And w_i is determined by port operators to represent the factors such as the priority for each Vessel i.

3.2.2 Decision variables

- u_i, the mooring time of Vessel i, $i \in V$
- v_i, the starting berth position occupied by Vessel i, $i \in V$
- c_i, the departure time of Vessel i, $i \in V$
- $x_{ij} \in \{0, 1\}$, equals to 1 if and only if Vessel i is completely on the left of Vessel j in the Time-space Diagram
- $y_{ij} \in \{0, 1\}$, equals to 1 if and only if Vessel i is completely below Vessel j in the Time-space Diagram

The solution of continuous BAP can be depicted in a Time-space Diagram where the allocation plan for each vessel is represented as a rectangle. Hereafter in this chapter, the terms, "vessel" and "rectangle" will be used interchangeably. As shown in Figure 3.1, the berthing plan for 3 incoming vessels is illustrated. In this example, $x_{12} = x_{21} = x_{31} = x_{32} = 0$, $x_{13} = x_{23} = 1$, and $y_{13} = y_{21} = y_{23} = y_{31} = y_{32} = 0$, $y_{12} = 1$.

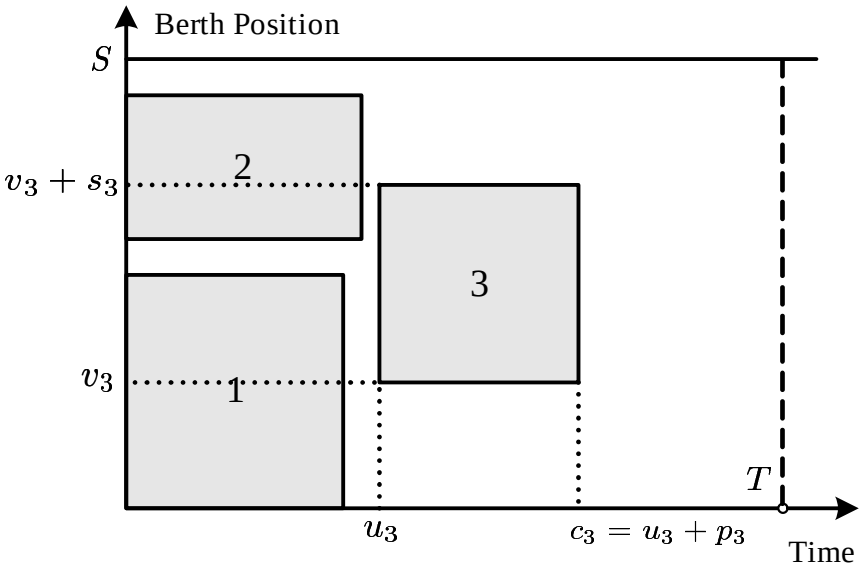


Figure 3.1: Time-space Diagram for a continuous BAP

3.2.3 Mathematical formulation

The mathematical formulation for the continuous dynamic BAP can be given as following:

$$\min \quad \sum_{i \in V} w_i(c_i - a_i) \tag{3.1}$$

s.t.

$$u_j - u_i - p_i - (x_{ij} - 1) \cdot T \geq 0, \qquad \forall\, i, j \in V, i \neq j \tag{3.2}$$

$$v_j - v_i - s_i - (y_{ij} - 1) \cdot S \geq 0, \qquad \forall\, i, j \in V, i \neq j \tag{3.3}$$

$$x_{ij} + x_{ji} + y_{ij} + y_{ji} \geq 1, \qquad \forall\, i, j \in V, i \neq j \tag{3.4}$$

$$x_{ij} + x_{ji} \leq 1, \qquad \forall\, i, j \in V, i \neq j \tag{3.5}$$

$$y_{ij} + y_{ji} \leq 1, \qquad \forall\, i, j \in V, i \neq j \tag{3.6}$$

$$p_i + u_i = c_i, \qquad \forall\, i \in V \tag{3.7}$$

$$a_i \leq u_i \leq (T - p_i), 0 \leq v_i \leq (S - s_i), u_i, v_i \in \Re^+ \quad \forall\, i \in V \tag{3.8}$$

$$x_{ij} \in \{0,1\}, y_{ij} \in \{0,1\}, \qquad \forall\, i,j \in V, i \neq j \tag{3.9}$$

In this formulation, the objective is to minimize the sum of weighted turnaround time for each incoming vessel. Constraints (2) and (3) give the definition of the decision variables x_{ij} and y_{ij}, respectively. Constraints (4) to (6) guarantee that Vessel i and Vessel j do not overlap in the Time-space Diagram. Constraints (7) show the relationship between the completion time (c_i) and the mooring time (u_i) for Vessel i. Constraints (8) and (9) define the feasible domains for decision variables u_i, v_i, x_{ij}, and y_{ij}.

In Lim (1998), continuous BAP has been proved to be $\mathcal{NP}$-complete. Hence, there is no efficient exact algorithm to solve BAP especially for large scale instances. However, one of the properties of the continuous BAP with the formulation described above, has been emphasized and applied to develop effective heuristics for the continuous BAP (Kim and Moon, 2003; Guan and Cheung, 2004; Wang and Lim, 2007).

Proposition 3.2.1. *There exists an optimal solution in which for a Vessel j,*

1. *either be berthed at its arrival time ($u_j = a_j$) or immediately after the completion time for some Vessel i ($u_j = u_i + p_i$);*

2. *either be allocated at lower or upper boundary of the berth ($v_j = 0$ or $v_j = S - s_j$) or to the adjoining regions for some Vessel i ($v_j = v_i + s_i$ or $v_j = v_i - s_j$).*

To see the proof of the Proposition 3.2.1, please refer to Guan and Cheung (2004). Nevertheless, compared with the statement of the proposition in Guan and Cheung (2004), more comprehensive description on the property is offered in this research particularly for item (2). In Guan and Cheung (2004), only $v_j = 0$ or $v_j = v_i + s_i$, for some Vessel i, is considered. Thus, in Guan and Cheung (2004), to search the possible locations in the Time-space Diagram where the next rectangle can be placed is intuitive since the contour of the packed vessels forms a stair and searching the possible locations for next vessel is equivalent to identify the steps of

the stair. However, in accordance with the restated proposition 3.2.1, how to find out the exact number of possible locations for next vessel is actually not trivial. Figure 3.2 demonstrates the oversimplification on the issue in Guan and Cheung (2004). According to Guan and Cheung (2004), the number of possible locations for Vessel 3 to place in the Time-space Diagram is just 3 (Figure 3.2.(a)). However, based on the restated Proposition 3.2.1 (Figure 3.2.(b)), beside Locations a, b, and c, Location d should be taken into account as well.

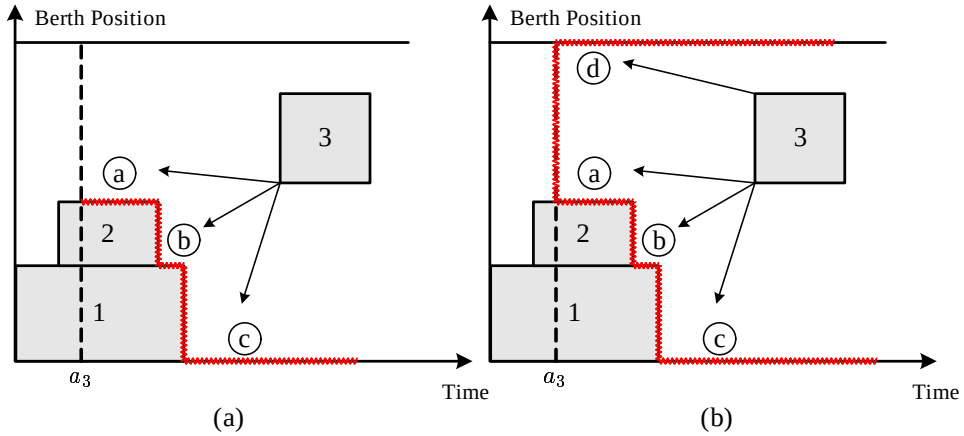


Figure 3.2: Possible locations to place next vessel

3.3 Method to Identify the Possible Locations in the Time-space Diagram

For heuristics (Kim and Moon, 2003; Guan and Cheung, 2004; Wang and Lim, 2007) proposed for solving the continuous BAP, a fundamental procedure for the algorithms is to construct a feasible solution. In the literature, the common strategy applied is the list insertion method. Given a list of all the incoming vessels, the algorithms try to insert the vessels orderly into the Time-space Diagram following the Proposition 3.2.1. During the procedure of insertion, an arising problem is how to determine the possible locations for next vessel by using the Proposition 3.2.1. As mentioned at the end of Section 3, Guan and Cheung (2004) addressed the issue by counting the step number of the stair shaped by the packed vessels in the diagram. However, this measure is not perfect since it is possible to filter out some potential locations for

next vessel. In fact, to identify all the possible locations in accordance with the Proposition 3.2.1 is a challenging problem, because as the number of packed rectangles increases, the configuration formed by the packed rectangles becomes more and more complicated. Figure 3.3 shows a case which reveals the difficulty to identify all the possible locations for next vessel (i.e., Vessel 5) by using the Proposition 3.2.1. In this example, the arrival times for Vessels 1, 2, and 3 are equal to 0 ($a_1 = a_2 = a_3 = 0$) and a_4 and a_5 are shown in the time axis. The previous 4 packed vessels plus the arrival time of Vessel 5, a_5, confine the possible domains for Vessel 5 to be allocated into two holes (the regions encircled by wave line). In light of the Proposition 3.2.1, inside the two holes, there are 6 possible locations for Vessel 5.

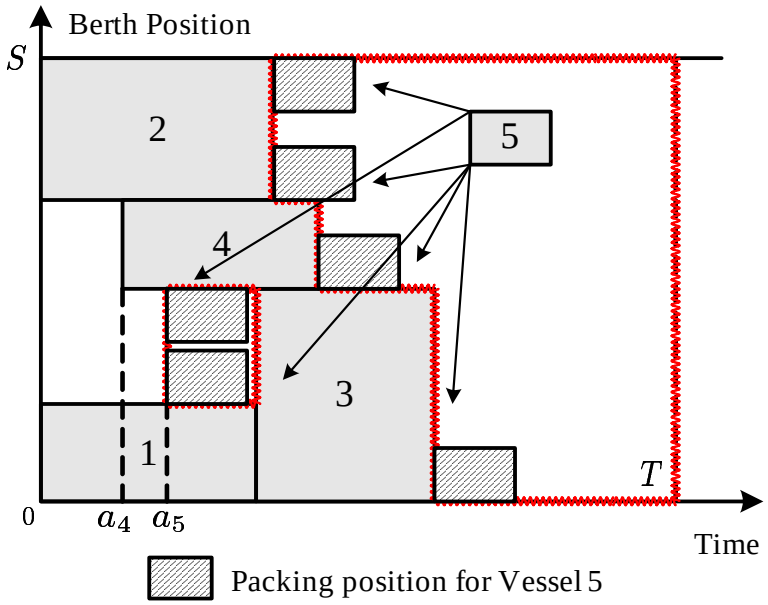


Figure 3.3: The possible locations for Vessel 5

A similar issue on how to report all the possible locations for next rectangle also occurs for Bin-packing Problem. Chazelle (1983) developed an efficient algorithm to identify all the possible locations where next rectangle can be placed for Bottom-left Bin-packing Heuristic. Firstly, the author defined certain special vertices and edges for a hole. Based on the property of Bottom-left Heuristic, firstly a hole in the Bin-packing Problem was divided into several sub-holes (see Figure 3.4). Then a "spring" device was used to test the feasibility of each sub-holes and report the possible locations for next rectangle.

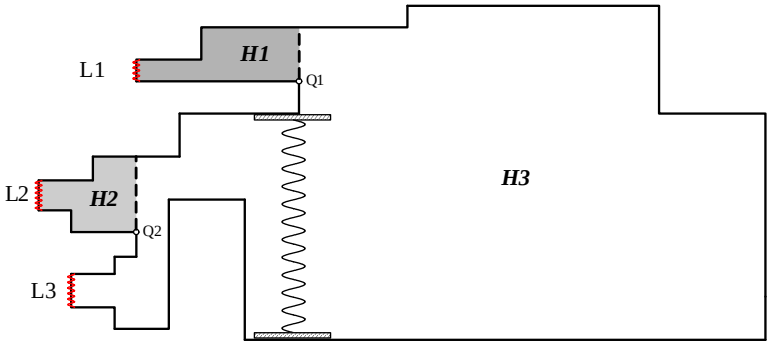


Figure 3.4: Separating the hole into 3 sub-holes

Nevertheless, in the continuous BAP, according to the Proposition 3.2.1, a hole in the Time-space Diagram is arbitrary enough that the property of Bottom-left Heuristic in Chazelle (1983) may not always hold. Hence, the algorithm proposed by Chazelle (1983) cannot be directly implemented in this research. However, the idea in Chazelle (1983) to classify the vertices of the holes provides some hints to construct an efficient method to solve the problem in the continuous BAP.

Before starting to describe the procedures of the proposed method, 4 terms will be clarified at first:

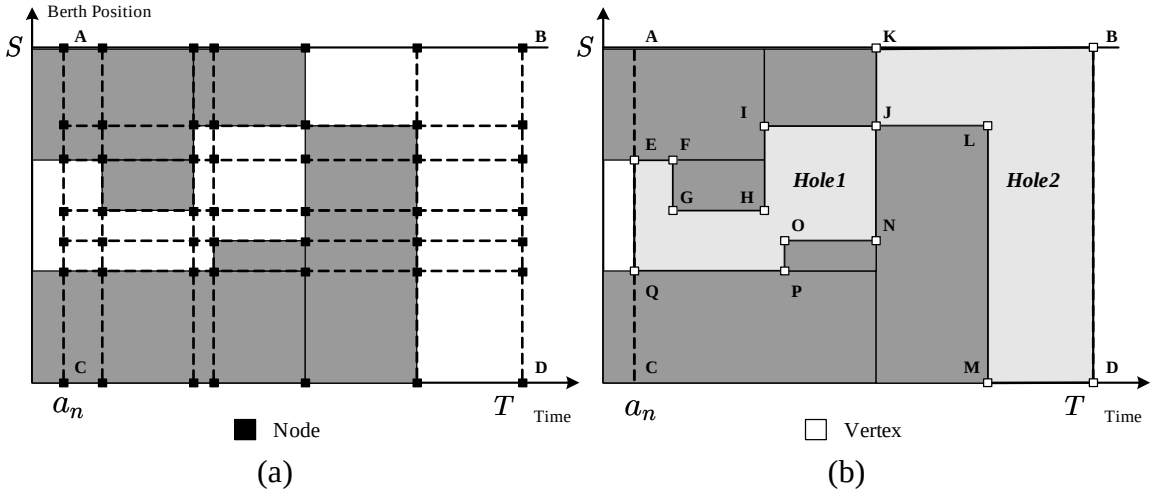


Figure 3.5: Nodes and vertices in the Time-space Diagram

- **Node**: Given the expected arrival time of current candidate Vessel n, and the boundary of the Time-space Diagram, a rectangle area can be defined to bound the possible region

for Vessel n to place (in the case of Figure 3.5(a), the rectangle is ABCD.). For all the packed vessels, they are either completely covered by ABCD or intersected with it. For the later situation, the rectangle of the packed vessel will be trimmed and the portion inside of ABCD will be considered only for the rest procedures. For packed Vessel i (or part of Vessel i) inside of ABCD, it will be associated with 2 horizontal segments and 2 vertical segments. Specifically, the two horizontal segments are $s = v_i$ and $s = v_i + s_i$; while the two vertical segments are $t = u_i$ and $t = u_i + p_i$ (for trimmed vessel, the two vertical segments are $t = a_n$ and $t = u_i + p_i$ instead). Group all the horizontal segments of the packed vessels as a horizontal segment set and denote it as $\mathbb{H}$. In the same manner, create a set $\mathbb{V}$ for all the vertical segments. For set $\mathbb{H}$, if segments $s = 0$ and/or $s = S$ are not included, add them into set $\mathbb{H}$; for set $\mathbb{V}$, if segments $t = a_n$ and/or $t = T$ are not included, add them into set $\mathbb{V}$. **A node in this method is defined as the intersection point between any horizontal segment in $\mathbb{H}$ and any vertical segment in $\mathbb{V}$** (as shown in Figure 3.5(a)).

- **Hole**: A hole is a closed area inside of ABCD which has not been occupied by any vessel so far. See Figure 3.5(b).
- **Vertex**: The term vertex is short for the vertex of a hole. A vertex is a special node in the Time-space Diagram.
- **Edge**: The edge of a hole.

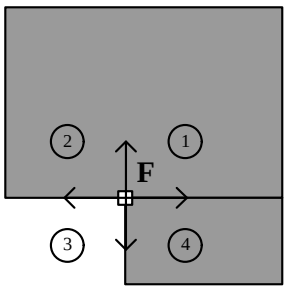


Figure 3.6: The 4 quadrants of Node F in Figure 3.5(b)

It can be observed that all nodes in the Time-space Diagram can be categorized into different groups after assigning a 4-element vector to each of them. The following describes how to assign

4-element vectors. For a node in the Time-space Diagram, the vicinity around the point can be divided into 4 quadrants, which are numbered increasingly in an anticlockwise direction (see Figure 3.6). For each quadrant, if the quadrant is impossible to provide room to pack any arbitrary rectangle, a number "0" will be tagged to the quadrant. Otherwise, number "1" will be chosen instead. For the case in Figure 3.6, since only Quadrant 3 is possible to pack another rectangle, a 4-element vector $[0, 0, 1, 0]$ will be associated with the Node F. Note that the value of i^{th} element of the vector represents the status of the Quadrant i.

After the assignment of vectors, the nodes can be classified into 5 groups according to the sum of the corresponding vectors. For example, the Node F in Figure 3.6 will be grouped into Class 1 since the sum of $[0, 0, 1, 0]$ is equal to 1. The following lists the vectors belonging to each class.

- Class 0: $[0, 0, 0, 0]$;
- Class 1: $[1, 0, 0, 0]$, $[0, 1, 0, 0]$, $[0, 0, 1, 0]$, $[0, 0, 0, 1]$;
- Class 2: $[1, 1, 0, 0]$, $[1, 0, 1, 0]$, $[1, 0, 0, 1]$, $[0, 1, 1, 0]$, $[0, 1, 0, 1]$, $[0, 0, 1, 1]$;
- Class 3: $[1, 1, 1, 0]$, $[1, 1, 0, 1]$, $[1, 0, 1, 1]$, $[0, 1, 1, 1]$;
- Class 4: $[1, 1, 1, 1]$.

The vertices of the holes are the pivots of the method. From Figure 3.5(b), it is clear that all the nodes in Class 1 and Class 3 are the vertices, and the nodes in Class 2 with vectors $[1, 0, 1, 0]$ and $[0, 1, 0, 1]$ (e.g., Node J in Figure 3.5(b)) also belong to the vertices of the holes. Actually, a node in Class 2 with vector $[1, 0, 1, 0]$ or $[0, 1, 0, 1]$ can be treated as the combination of two different but overlapping nodes in Class 1. For a node with $[1, 0, 1, 0]$, it can be separated into two overlapping nodes with $[1, 0, 0, 0]$ and $[0, 0, 1, 0]$; for a node with $[0, 1, 0, 1]$, it serves the same function as two Class 1 nodes with $[0, 1, 0, 0]$ and $[0, 0, 0, 1]$. Taking Node J in Figure 3.5(b) as an example, it can be decomposed into two nodes associated vectors $[1, 0, 0, 0]$ and $[0, 0, 1, 0]$, respectively. Hereafter, every node in Class 2 with vector $[1, 0, 1, 0]$ or $[0, 1, 0, 1]$ will be treated as two overlapping nodes in Class 1 in the Time-space Diagram. **Consequently, the set of**

vertices is equivalent to the nodes in Class 1 and Class 3 after the treatment of nodes in Class 2.

In light of the Proposition 3.2.1, two sorts of nodes are associated with the possible locations for next vessel. The first kind is the nodes in Class 1 with vector $[1, 0, 0, 0]$ and $[0, 0, 0, 1]$ (see Figure 3.7, Nodes I and Q); the other part is related to the nodes in Class 3 (see Figure 3.7, Nodes G, H, and O). For the nodes in the second part, the following will provide the rules to identify them.

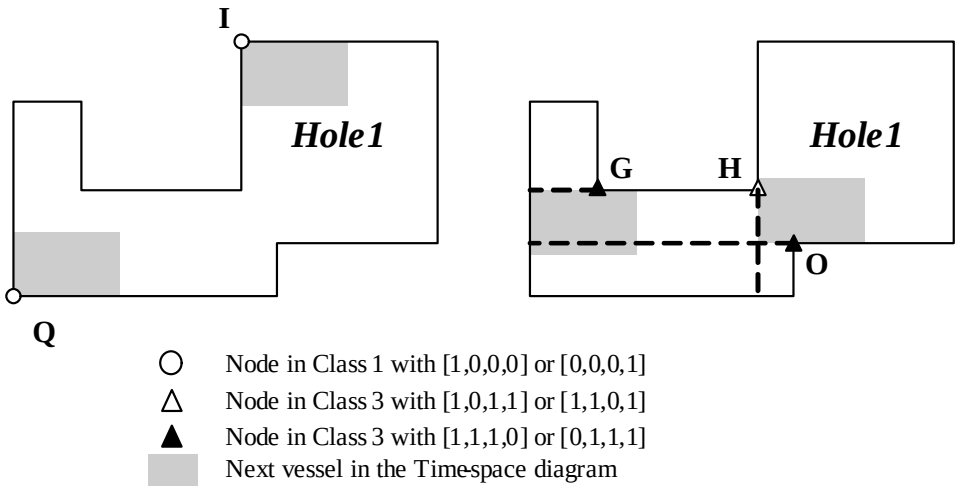


Figure 3.7: The nodes associated with possible locations for Hole 1 in Figure 3.5(b)

For a node with $[0, 1, 1, 1]$ or $[1, 1, 1, 0]$ (e.g., Nodes G and O in Figure 3.8), a leftward extension segment will be added and intersected with the boundary of holes; for a node with $[1, 0, 1, 1]$ (e.g., Node H in Figure 3.8), it will generate a downward segment while for a node with $[1, 1, 0, 1]$, an upward segment started from the node will be generated. Two kinds of intersection nodes will be highlighted in this procedure. The first kind of intersection nodes are the intersection points between the extension segments and the edges of the holes (e.g., Nodes R, T, and W in Figure 3.8). The second type of intersection nodes are the intersection points between the extension segments of node with $[1, 1, 1, 0]$ and node with $[1, 0, 1, 1]$ (like Node U in Figure 3.8), or between the extension segments of node with $[0, 1, 1, 1]$ and node with $[1, 1, 0, 1]$. Group the first kind of intersection nodes in the set $\mathbb{I}_1$; for the second kind of intersection nodes, group them in the set $\mathbb{I}_2$.

Table 3.1: Vector updating rules for the nodes in $(\mathbb{I}_1 \bigcup \mathbb{I}_2)$

For nodes in $\mathbb{I}_1$			For nodes in $\mathbb{I}_2$		
$\mathbf{a}$		$\mathbf{b}_{\mathbb{I}_1}$	$\mathbf{a}_1$, $\mathbf{a}_2$		$\mathbf{b}_{\mathbb{I}_2}$
$[0,1,1,1]$	$\Longrightarrow$	$[0,0,0,1]$			
$[1,0,1,1]$	$\Longrightarrow$	$[1,0,0,0]$	$[1,0,1,1]$, $[1,1,1,0]$	$\Longrightarrow$	$[1,0,0,0]$
$[1,1,0,1]$	$\Longrightarrow$	$[0,0,0,1]$	$[1,1,0,1]$, $[0,1,1,1]$	$\Longrightarrow$	$[0,0,0,1]$
$[1,1,1,0]$	$\Longrightarrow$	$[1,0,0,0]$			

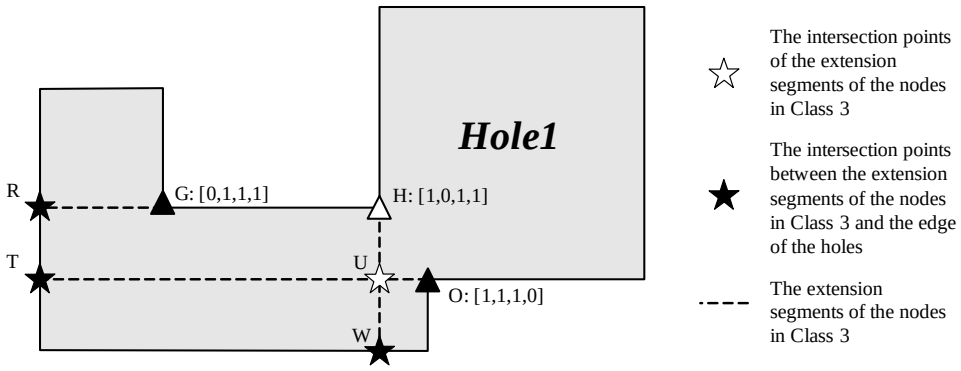


Figure 3.8: The extension segments of nodes in Class 3 and the intersection nodes

After identifying all the intersection nodes, their corresponding vectors will be updated according to Table 3.1. For a node in $\mathbb{I}_1$, if it is the ending point of an extension segment generated by a node with vector $\mathbf{a}$, then its vector will be changed to vector $\mathbf{b}_{\mathbb{I}_1}$. For example, for the Node R in Figure 3.8, its vector will be changed from $[1,0,0,1]$ to $[0,0,0,1]$, since Node R is the ending point of an extension segment generated by Node G with vector $[0,1,1,1]$. For a node in $\mathbb{I}_2$, if the two intersected extension segments are generated from nodes with vectors $\mathbf{a}_1$ and $\mathbf{a}_2$, then the new vector for the intersection node will be $\mathbf{b}_{\mathbb{I}_2}$. In the case of Node U in Figure 3.8, its vector will be updated from $[1,1,1,1]$ to $[1,0,0,0]$ as the vectors of Node H and Node O are $[1,0,1,1]$ and $[1,1,1,0]$, respectively.

So far, the potential nodes in the Time-space Diagram which may be associated with possible locations for next vessel have been identified (specifically, the nodes in Class 1 with vector $[1,0,0,0]$ and $[0,0,0,1]$ and the nodes in $\mathbb{I}_1 \bigcup \mathbb{I}_2$) and updated their corresponding vectors if necessary. However, taking the information of next vessel into account (such as the length of

the vessel, the berthing time of the vessel), packing next vessel in the available quadrant of a potential node may overlap with other packed vessels or violate the optimal conditions as stated in the Proposition 3.2.1. Therefore, for each potential node, feasibility and optimality tests should be carried out.

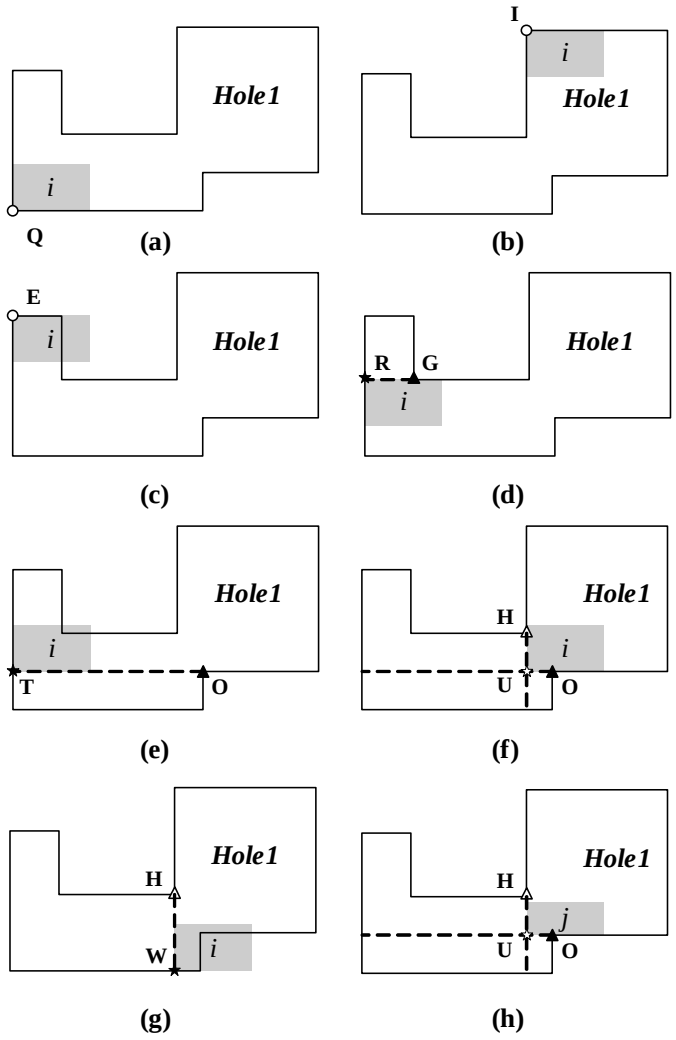


Figure 3.9: Feasibility and optimality test for Hole 1 in Figure 3.5(b)

Figure 3.9(a)-3.9(g) show the feasibility and optimality tests for the potential nodes when next vessel is Vessel i. In Hole 1, there are 3 nodes in Class 1 with $[1,0,0,0]$ and $[0,0,0,1]$ (i.e., Nodes Q, I, and E) and 4 nodes in $\mathbb{I}_1 \bigcup \mathbb{I}_2$ (i.e., Nodes R, T, U, and W). Among the 7 nodes, only cases (a), (b), (d), and (f) pass the feasibility and optimality tests. In cases (c), (e), and (g), Vessel i overlaps with some of the packed vessels. When next vessel is Vessel j, Figure

3.9(h) shows a case in which feasibility condition is satisfied but the optimality condition does not fulfill. In Figure 3.9(h), packing Vessel j at the 1^{st} quadrant of Node U does not overlap with other packed vessels. However, since the size of Vessel j is less than $|HU|$ and the rectangle can slide leftward to reduce cost, packing Vessel j at this location is not optimal according to the Proposition 3.2.1.

The method to identify the possible locations for next vessel by using the Proposition 3.2.1 is summarized as follows:

Step 1: Input the information of packing locations for the previous $(n-1)$ vessels and next vessel (Vessel n), such as the arrival time a_n, the processing time p_n, and vessel length s_n.

Step 2: Identify the nodes within ABCD (see Figure 3.5(a)) and classify them into 5 groups. If the set of nodes in Class 2 is not empty, for each node with vector $[1,0,1,0]$ or $[0,1,0,1]$ in Class 2, separate it into two different but overlapping nodes in Class 1.

Step 3: Group all the nodes in Class 1 with vector $[1,0,0,0]$ and $[0,0,0,1]$ as a node set $\mathbb{C}_1$; group all the nodes in Class 3 into a node set $\mathbb{C}_3$.

Step 4: For each node in $\mathbb{C}_3$, an extension segment will be generated from it and ended at the boundary of the holes. Identify the two kinds of intersection points and store them in the sets $\mathbb{I}_1$ and $\mathbb{I}_2$, respectively. Update the vector for each node in $\mathbb{I}_1 \bigcup \mathbb{I}_2$.

Step 5: Carry out feasibility and optimality tests for each node in $\mathbb{C}_1 \bigcup \mathbb{I}_1 \bigcup \mathbb{I}_2$. Each node passing the tests will be associated with a possible location for next vessel, by packing Vessel n at the available quadrant of it.

Compared with the exhaustive enumeration approach (see Figure 3.5(a), non-overlapping test and the check of optimal conditions will be carried out for each node within ABCD), the performance of the proposed method is shown in Table 3.2. For the 6 numerical experiments, the berth length $S = 100$, the size, processing time, and arrival time of the n vessels follow discrete uniform distributions $U(6,50)$, $U(20,80)$, and $U(0,50)$ respectively. In a greedy way, the previous $(n-1)$ vessels are packed and the two approaches are tested by inserting the Vessel

Table 3.2: The performance of proposed method compared with exhaustive enumeration

Number of	Enumeration approach		Proposed method	
vessels, n	Possible locations	CPU time (s)	Possible locations	CPU time (s)
51	3	8.125	3	0.0469
101	9	62.672	9	0.0938
151	3	161.657	3	0.188
201	10	297.594	10	0.234
251	7	536.672	7	0.359
301	10	740.641	10	0.438

n. Both methods are coded in Matlab and all the experiments are executed in a Pentium (R) IV PC with 3.00 GHz CPU 2.00 GB of RAM. From Table 3.2, it is obvious that the proposed method can identify all the possible locations for next vessel, but be executed in a highly efficient way.

3.4 GRASP for the Continuous BAP

Greedy Randomized Adaptive Search Procedure (GRASP) is a multi-start metaheuristic which was developed in the late 1980s to solve combinatorial problems. In the field of port operation, Kim and Park (2004) used GRASP to solve the QCSP. The framework of GRASP consists of two phases in each iteration: construction and local search. The pseudo code for GRASP is illustrated in Table 3.3. In the solution construction phase, candidate elements are inserted into the partial solution iteratively until a complete solution is constructed. The selection of the next element for insertion is determined by the evaluation of all candidate elements according to a greedy evaluation function (Resende and Ribeiro, 2003). In the second phase of GRASP, a local search technique is used to replace the current solution by a better solution in the neighbourhood of the current solution. The best solution is always updated at the end of the iteration. The procedure will be terminated until one of the stopping conditions is satisfied.

For the continuous BAP, two versions of GRASP are designed to obtain the near optimal solutions. The first GRASP (called GRASP_1) follows the rule of "first-come-first-pack" to construct the initial solution. In its second phase, swapping of two adjacent vessels in the

Table 3.3: The pseudo code for GRASP (Resende and Ribeiro, 2003)

```
Procedure GRASP(Max_Iterations, Seed)
1.    Read_Input();
2.    for k = 1, ..., Max_Iterations do
3.        Solution← Greedy_Randomized_Construction(Seed);
4.        Solution← Local_Search(Solution);
5.        Update_Solution(Solution,Best_Solution);
6.    end;
7.    return Best_Solution;
end GRASP.
```

insertion list and an A-star like tree search procedure is chosen as the local search technique. On the contrary, the second version of GRASP (called GRASP_2) does not follow the "first-come-first-pack" rule. In its construction phase, the arrival time of vessel is not considered as an important priority for the packing order of each vessel in the Time-space Diagram. The local search phase is as same as the one in GRASP_1 except that the swapping operation can happen between any two vessels. The followings describe the steps of GRASP.

3.4.1 GRASP_1: construction phase

Step 1: Input the data for all the incoming vessels and the berth length, S. Sort the vessels in an increasing order according to their arrival time. The order of the vessels after the sorting is denoted as Ves_Seq.

Step 2: Insert the vessels following the list of Ves_Seq. Suppose next vessel is Vessel j. Input the information of Vessel j and the packing position for the previous $(j-1)$ vessels in the Time-space Diagram into the method proposed in Section 4 to determine the possible locations for Vessel j. Assume that there are K possible locations for Vessel j. For each possible location k, after allocating Vessel j to the position, the cost calculated by Function (3.1) for the previous j vessels is recorded and denoted as $Cost_k$. Exclude the possible location k, if it fulfills

$$\frac{1}{Cost_k} < r \times \max_{i=1}^{K}\{\frac{1}{Cost_i}\} \tag{3.10}$$

where r is a constant in $[0, 1]$. Let the number of possible locations after the screening be $\widetilde{K}$.

Step 3: Associate each possible location after screening a probability. For example, for possible Location $\widetilde{k}$, the associated probability is defined as:

$$Pr_{\widetilde{k}} = \frac{\frac{1}{Cost_{\widetilde{k}}}}{\sum_{i=1}^{\widetilde{K}} \frac{1}{Cost_i}} \tag{3.11}$$

Step 4: Randomly Choose a possible Location $\widetilde{k}$ for Vessel j based on the probability distribution defined by Equation (3.11). In this step, roulette wheel selection method widely used in the genetic algorithm is adopted.

Step 5: Repeat Steps 2-4 until all the vessels are included.

3.4.2 GRASP_1: local search phase

The Local Search technique used in this study has two procedures: swapping the packing order of two adjacent vessels and an A-star like tree search procedure. Given an insertion list for the vessels $i_1, i_2, \ldots, i_e, i_{e+1}, \ldots, i_n$, the swapping procedure produces a new sequence $i_1, i_2, \ldots, i_{e+1}, i_e, \ldots, i_n$. The reason to limit the swapping operation just between two adjacent vessels is to follow the "first-come-first-pack" rule as much as possible. The A-star like tree search is activated after the swapping procedure. For insertion list $i_1, i_2, \ldots, i_e, i_{e+1}, \ldots, i_n$, a break point ($e$, for example) will cut the list into two substrings: $i_1, i_2, \ldots, i_e$ and $i_{e+1}, \ldots, i_n$, respectively. Keeping the packing locations of first e vessels intact, the A-star like tree search technique will be used to repack the next $(n - e)$ vessels. For example, Vessel j $(e + 1 \leq j \leq n)$ is the next considered vessel and there are K possible locations for it. For each Location k $(1 \leq k \leq K)$, a greedy heuristic for the rest $(n - j + 1)$ vessels will be used to evaluate the benefit value $h(j, k)$ by allocating Vessel j to Location k. The location $k^* = \text{argmax}_{k=1}^{K}\{h(j, k)\}$ will be chosen as the packing location for Vessel j. The procedure will repeat until the ideal position for Vessel n is assigned. Detail steps are summarized:

Step 1: Swap the packing order of two adjacent vessels and evaluate the benefit value of the modified insertion list.

Step 2: Repeat Step 1 for L_1 times and choose the best insertion list for the n vessels.

Step 3: Cut the insertion list into two pieces at the position of e $(1 < e < n)$. A-star like tree search technique is adopted to find a better (if possible) packing pattern for the rest vessels (from Vessel $e+1$ to Vessel n).

3.4.3 GRASP_2: construction phase

The construction steps of GRASP_2 is almost the same as the procedures of GRASP_1 except Steps 1 and 2.

Step 1: Input the data for all the incoming vessels and the berth length, S. Denote $V = \{1, 2, \ldots, n-1, n\}$ the index set of the vessels. Define an index set $Packed\_Seq$ as the set of vessels that have been packed in the Time-space Diagram and initialize it as $\emptyset$. Define another index set $Unpacked\_Seq = V \setminus Packed\_Seq$, the set of unpacked vessels.

Step 2: Let set Λ be the set of all possible locations for next vessel and initialize it as a null set. For each Vessel i, $i \in Unpacked\_Seq$, apply the proposed method in Section 3.4 to identify all the possible locations for it and insert them into the set Λ. Compute the cost associated with each location and use Inequality (3.10) to filter out some unpromising positions from Λ. Update Λ.

Step 3: Associate each possible location in Λ a probability like Equation (3.11).

Step 4: Using roulette wheel selection method to select a position. Update the set $Packed\_Seq$ and $Unpacked\_Seq$, respectively.

Step 5: Repeat Steps 2-4 until a complete solution is constructed.

3.4.4 GRASP_2: local search phase

As mentioned before, the local search phase is almost identical to the one of GRASP_1. The only difference exits in the procedure of swapping. In GRASP_2, the swapping can happen between any pair of vessels.

It is noted that compared with traditional GRASP, there is a unique feature of the implementation of GRASP in this research. Based on the solutions found so far, if the objective value becomes steady in the previous L_2 iterations, the cut position e for the A-star like tree search procedure in the local search phase will be decreased by 1. Once the value of e changes, the value of cut position cannot be altered for the next L_3 iterations. Beside, since the smaller the e is, more time will be consumed in the tree search procedure, there is a lower bound for e, LB. The adaptive scheme can help the GRASP to prevent stagnation and obtain better solutions. Both GRASP_1 and GRASP_2 will terminate as long as one of the following criteria is satisfied:

1. The maximum iteration number (L_4) is reached.

2. The best value of the objective function during the previous L_5 consecutive loops has converged.

3.4.5 GRASP compares with CPLEX

For small scale problem, the performance of GRASP_1 and GRASP_2 is tested through the comparison with CPLEX. In the numerical experiments, the berth length S is set to 80. The size, processing time, and arrival time of the vessels follow the discrete uniform distributions $U(6, 50)$, $U(20, 80)$, and $U(0, 20)$. The weights for the vessels are all equal to 1. Two scenarios with different vessel number (i.e., $n = 5$ and $n = 10$) are studied. For each scenario, 30 instances are randomly generated and solved by CPLEX and proposed GRASPs, respectively. Both GRASP_1 and GRASP_2 are coded in Matlab and executed in a Pentium (R) IV PC with 3.00GHz CPU and 2.00 GB of RAM. Due to the stochastic nature of GRASP, GRASP_1 and GRASP_2 are all executed 5 times for each instance. The following lists the parameters for the GRASPs:

Table 3.4: The performance of GRASP_1 and GRASP_2 compared with CPLEX

Number of	CPLEX	GRASP_1		GRASP_2	
vessels, n	ave_time†(s)	ave_time(s)	ave_Gap‡(%)	ave_time(s)	ave_Gap(%)
5	0.39	2.29	1.62	2.73	0.005
10	5977.01	11.90	7.89	16.59	3.70

†: average CPU time consumed
‡: average gap between the solutions found by GRASP and CPLEX

1. L_1, the number to swapping operation is set to 10;
2. L_2, L_3 used in the adaptive scheme are equal to 5 and 10;
3. L_4, the maximum iteration number equals to 200;
4. L_5, the value chosen to judge the convergence is 25;
5. e, the cut position follows the equation $e = [\frac{7}{8} \times n]$, where function $[x]$ round the value x towards the nearest integer;
6. LB, the lower bound for e, equals to $[\frac{3}{4} \times n]$;
7. r, the parameter used in Inequality (3.10), is set to 0.3.

Table 3.4 shows the computational results for the two scenarios. Obviously, as the size of the problem enlarges, CPLEX will consume unacceptable longer time to find the optimal solutions. While for the proposed GRASPs, the average time spent to search a good solution is within 17 seconds for all the 60 instances. GRASP_2 tends to consume more computational time while reap better solutions compared with GRASP_1. This phenomenon is reasonable because in the construction phase, GRASP_2 inclines to spend more time to construct a more promising initial solution than GRASP_1 as the former searches much broader solution space compared to the latter.

3.4.6 GRASP compares with the SBS

For large scale problems, the proposed GRASPs are compared with the stochastic beam search developed by Wang and Lim (2007). A brief description about the SBS has been given in the

Section 2. Note that the insertion list in Wang and Lim (2007) strictly follows the "first-come-first-pack" rule. Three key parameters used in the SBS suggested in Wang and Lim (2007) are listed below:

1. u, filtered beam width for draft selection: 1000;
2. b, filtered beam width for detail selection: 33, 66, 150, 330;
3. Δ, search depth for depth-first-search based detail selection: 3, 4, 5, 6.

To demonstrate the effectiveness of the proposed GRASPs, the extreme combination of parameters (u, b, Δ) is set to $(1000, 330, 6)$ in this comparison (since the larger the parameters are, more attractive solutions can be found by the SBS). The problem settings are kept as the ones used in the small scale numerical experiments. On the contrary, there are some adjustments for the tunable parameters used in GRASP so as to accelerate the algorithm for large scale problems.

1. L_1, 3;
2. L_2, 2;
3. L_3, 3;
4. L_4, 200;
5. L_5, 4;
6. e, $e = \max\left\{[\frac{7}{8} \times n], n - 20\right\}$;
7. LB, $LB = \max\left\{[\frac{3}{4} \times n], n - 40\right\}$;
8. r, 0.3.

5 scenarios with different vessel number are carried out. For each scenario, 30 instances are randomly generated and used to test the performance of SBS and proposed GRASPs. Note that for each instance, the 3 algorithms are all executed 3 times. Table 3.5 summarizes the computational results for the 150 instances. On average, GRASPs are able to find better

Table 3.5: The performance of GRASPs compared with SBS

Number of	SBS	GRASP_1		GRASP_2	
vessels, n	ave_time[std]†	ave_time[std]	ave_impro[std]‡	ave_time[std]	ave_impro[std]
40	188[17]	15[4]	0.99[0.03]	37[8]	0.98[0.03]
80	790[31]	112[32]	0.98[0.02]	271[55]	0.94[0.03]
120	911[20]	413[121]	0.97[0.01]	882[189]	0.93[0.03]
160	1703[46]	1103[367]	0.97[0.01]	2137[416]	0.92[0.02]
200	2809[64]	1763[702]	0.97[0.01]	4223[724]	0.92[0.02]

†: average CPU time consumed in second and standard deviation
‡: average ratio between the solutions found by GRASP and SBS and standard deviation

solutions in comparison with SBS. Specifically, GRASP_1 statistically outperforms SBS in terms of average computational time and the solution quality. While GRASP_2 is able to obtain the best solutions among the three algorithms, it will take longer time to accomplish that especially as the number of vessels expands. Based on the findings from the experiments, GRASP_1 is more recommended to handle large scale problem after considering the tradeoff between effectiveness and efficiency.

3.5 Summary

In this research, continuous and dynamic BAP is studied to minimize the total weighted flow time. Different from previous studies on continuous BAP, an efficient method is proposed to address the problem to identify the possible locations for next vessel in the Time-space Diagram. Then two versions of GRASPs are developed to search for near optimal solutions. Both small and large scale numerical experiments are tested to examine the effectiveness of the proposed GRASPs by comparison with CPLEX and SBS, respectively.

Chapter 4

An Improved Approach for Quay Crane Scheduling Problem with Non-crossing Constraints

4.1 Background

In container port operation, one of the critical issues is to schedule quay cranes to serve the berthed container vessels. Quay cranes are industry-standard equipment for loading and discharging containers to and from vessels. Practically, quay cranes along the same berth are mounted on the same tracks, which forbid them from crossing each other at any instant. To get the solution of quay crane scheduling accurately and efficiently will provide immediate benefit and input to subsequent port operations such as yard truck and yard crane scheduling. Several studies have been conducted to improve the efficiency of the QCSP with consideration of Non-crossing Constraints. However, it has been noted that the previous models of the QCSP with Non-crossing Constraints, may cause unrealistic model solutions. Therefore, in this chapter, firstly the found deficiencies in modeling QCSP with Non-crossing Constraints in the previous studies will be identified. Next, since the QCSP with Non-crossing Constraints is a strongly $\mathcal{NP}$-hard problem, in this chapter, two approximation algorithms will be proposed to obtain

the approximate solutions for the QCSP with Non-crossing Constraints.

4.2 Problem Description

The previous modeling efforts for the QCSP with Non-crossing Constraints (from Kim and Park 2004 to Tavakkoli-Moghaddam et al. 2009) generally follow the assumptions below:

- Quay cranes are identical machines which are operated on the same tracks.
- Once a quay crane starts processing a bay, it leaves only when it has finished the workload of the bay. In other words, the jobs of loading/discharging containers in a bay are without preemption.
- Compared with the processing time of a ship bay, the travel time of a quay crane between two bays is small and hence it is not considered.

Carefully observation can find out that in spite of adopting different parameters or decision variables, the main formulation structures of the previous studies for the QCSP with Non-crossing Constraints are almost the same. They all follow the prototype formulation as following:

4.2.1 Parameters

- m, the number of quay cranes
- n, the number of ship bays in a vessel
- p_i, the workload of Bay i, $1 \leq i \leq n$

4.2.2 Decision variables

- $c_{\max}$, the makespan for the QCSP with Non-crossing Constraints
- c_i, the completion time of Bay i, $1 \leq i \leq n$
- $x_{ik} \in \{0, 1\}$, equals to 1 if and only if Bay i is handled by Crane k
- $y_{ij} \in \{0, 1\}$, equals to 1 if and only if Bay i completes no later than Bay j starts

4.2.3 Mathematical formulation

The mathematical formulation can be given as:

$$\text{minimize} \quad c_{\max} \tag{4.1}$$

subject to

$$c_{\max} \geq c_i, \qquad \forall\, 1 \leq i \leq n \tag{4.2}$$

$$c_i - p_i \geq 0 \qquad \forall\, 1 \leq i \leq n \tag{4.3}$$

$$\sum_{k=1}^{m} x_{ik} = 1 \qquad \forall\, 1 \leq i \leq n \tag{4.4}$$

$$c_i - (c_j - p_j) + M \cdot y_{ij} \geq 0 \qquad \forall\, 1 \leq i, j \leq n \tag{4.5}$$

$$(c_j - p_j) + M \cdot (1 - y_{ij}) - c_i \geq 0 \qquad \forall 1 \leq i, j \leq n \tag{4.6}$$

$$M \cdot (y_{ij} + y_{ji}) \geq \sum_{k=1}^{m} k \cdot x_{ik} - \sum_{l=1}^{m} l \cdot x_{jl} + 1 \qquad \forall 1 \leq i < j \leq n \tag{4.7}$$

$$x_{ik}, y_{ij} \in \{0, 1\} \qquad \forall\, 1 \leq i,\, j \leq n, \forall 1 \leq k \leq m \tag{4.8}$$

$$c_{\max}, c_i \in \Re^{+} \qquad \forall\, 1 \leq i \leq n \tag{4.9}$$

The objective function (4.1) is to minimize the makespan of one container vessel. Constraints (4.2) define the property of $c_{\max}$. Constraints (4.3) make sure that the completion time of an individual bay should be larger than its workload. Constraints (4.4) ensure that one bay must be assigned to exactly one quay crane. Constraints (4.5) and (4.6) define the properties of y_{ij}. Constraints (4.7) specify the Non-crossing Constraints. That is, if Bay i is assigned to Crane k, Bay j is assigned to Crane l simultaneously (in this case, $y_{ij} + y_{ji} = 0$), then $l \geq k + 1$. Note that since quay cranes and bays are numbered in the same direction along the berth, $i \leq j$ and $k + 1 \leq l$ ensure the Non-crossing Constraints.

The MIP above represents the formulation of the QCSP with Non-crossing Constraints in the previous studies. However, after careful examination, it can be found out that this mathematic model is not complete in terms of that it may generate unrealistic solution even for optimal solution. For any feasible solution of the QCSP with Non-crossing Constraints, they could be presented in a Time-space Diagram. Figure 4.1 shows a case for 3 quay cranes assigned for

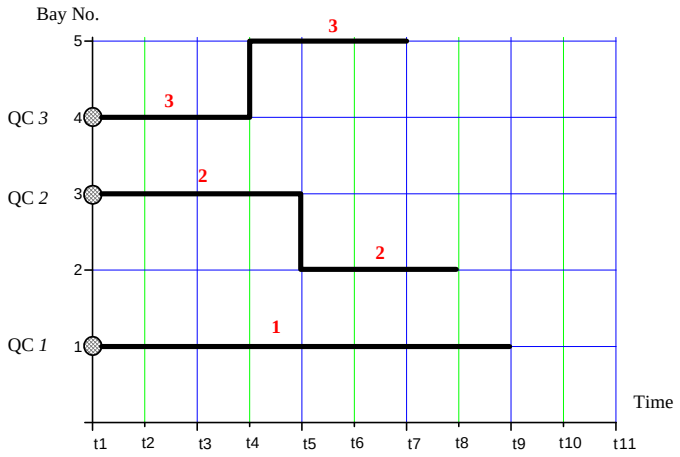


Figure 4.1: Time-space Diagram for a feasible solution of the QCSP with Non-crossing Constraints

a vessel with 5 ship bays. The stepwise horizontal dark lines for every quay crane represent the working schedules of the quay crane on the assigned ship bays. For example, Crane 3 will handle Bay 4 in time interval $[t_1, t_4]$ and Bay 5 in $[t_4, t_7]$. In the previous studies, a scheduling scheme will be considered as a feasible solution as long as the scheduling scheme satisfies:

- **Condition 1:** each ship bay is only handled exactly by one quay crane during the planning horizon.
- **Condition 2:** at any instant, quay cranes should not cross over each other (as illustrated in Figure 4.1, the dark lines belonging to a quay crane should not cross over dark lines belonging to other quay cranes).

Nevertheless, these conditions are not sufficient to guarantee a realistic solution. The problems existed in the aforementioned models on the QCSP with Non-crossing Constraints are summarized in the following:

- Although the Non-crossing Constraints are not violated at any moment, for some time intervals, the space between any two quay cranes may not be sufficient to accommodate the quay cranes to be positioned between them. A typical case is shown in Figure 4.2(a). For example, during time interval $[t_4, t_7]$, the space between Crane 1 and Crane 3 is zero

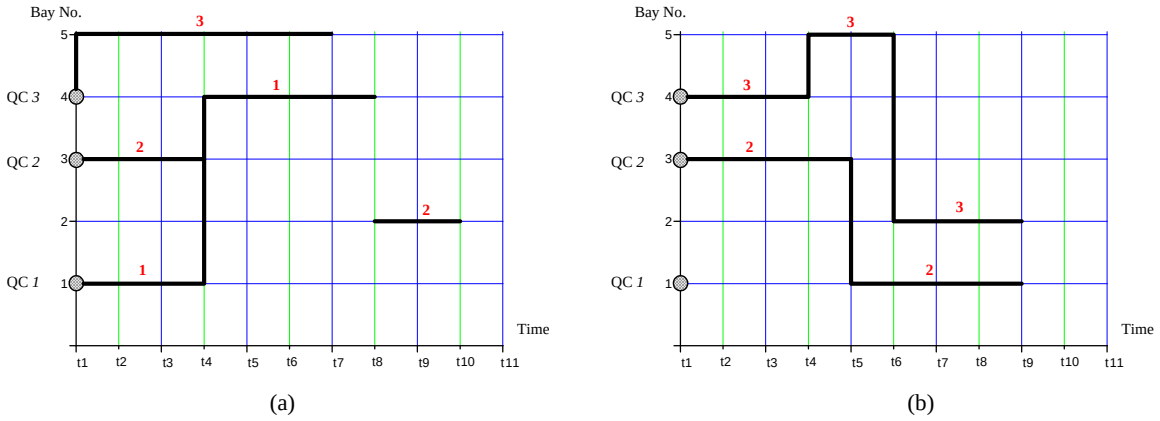


Figure 4.2: Two problems for the previous studies

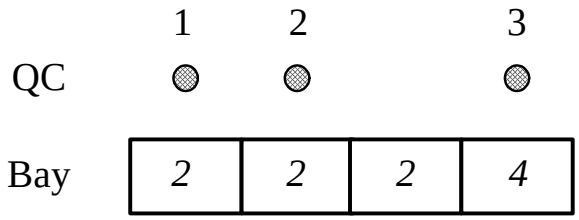


Figure 4.3: A example for the QCSP with Non-crossing Constraints

ship bay, which means that Crane 2 has to be "dragged" out of the tracks to let Crane 1 and Crane 3 be positioned next to each other.

- Even if the aforementioned Conditions 1 and 2 can be fulfilled, it is still possible that some quay cranes will be driven out of the space boundary of a vessel. As illustrated by Figure 4.2(b), it is clear that, to avoid crossing, Crane 1 will be driven out of the vessel boundary.

Other than the examples of feasible solutions depicted in Figure 4.2, it should be noted that these two unrealistic situations are also possible to happen in optimal solutions generated by the previous studies. The following is an example to illustrate the occurrence of these two unrealistic situations for the QCSP with Non-crossing Constraints. Figure 4.3 shows a small scale problem with 4 ship bays (the workload in each bay is $2, 2, 2, 4$, respectively) and 3 quay cranes. Formulate the problem according to the previous model and solve it in CPLEX. Two

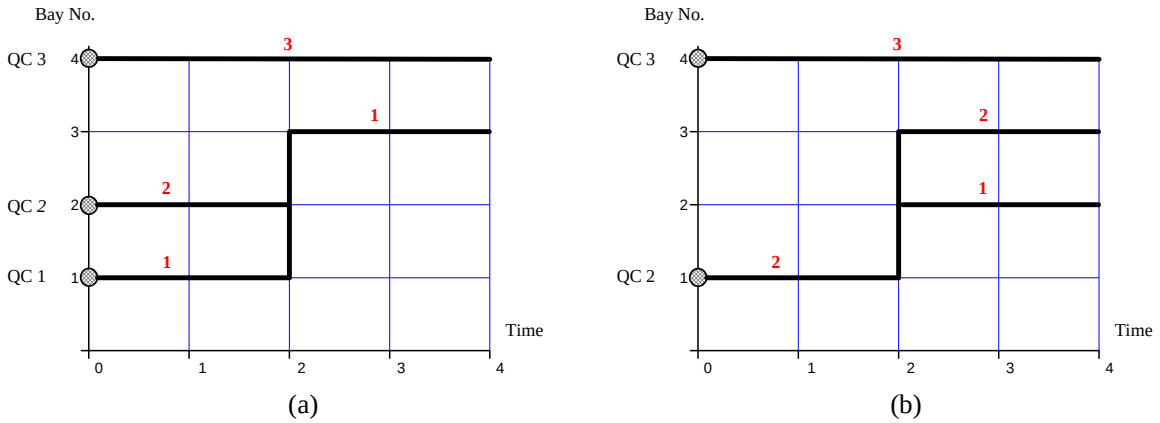


Figure 4.4: Two possible optimal solutions for the small scale problem

possible optimal solutions can be found and presented in Figure 4.4. In Figure 4.4(a), within the time interval $[2, 4]$, Crane 1 forces Crane 2 out of the tracks; however, in Figure 4.4(b), Crane 1 is out of the vessel boundary in $[0, 2]$.

4.2.4 Revised modeling for the QCSP with Non-crossing Constraints

To overcome these deficiencies, additional constraints are introduced to prevent the occurrence of these problems in optimal solution. For the first problem (as shown in Figure 4.4(a)), in order to guarantee that there is enough space between any two quay cranes, Constraints (4.10) can be added. For Constraints (4.10), they ensure that if Bay i is assigned to Crane k, Bay j is assigned to Crane l simultaneously, then $j - i \geq l - k$. Therefore, the quay cranes between k and l can be accommodated properly.

$$M \cdot (y_{ij} + y_{ji}) \geq \sum_{l=1}^{m} l \cdot x_{jl} - \sum_{k=1}^{m} k \cdot x_{ik} + (i - j) \qquad \forall\, 1 \leq i < j \leq n \tag{4.10}$$

The second problem (see Figure 4.4(b)) can be solved by introducing two dummy quay cranes and two dummy ship bays, namely Crane 0, Crane $(m + 1)$ and Bay 0 and Bay $(n + 1)$. To avoid any quay crane being driven out of the vessel boundaries, Crane 0, Crane $(m + 1)$ will be assigned to Bay 0 and Bay $(n + 1)$ respectively during the whole planning horizon (Let $x_{00} = 1, x_{n+1,m+1} = 1$ and additionally the processing time p_0 and p_{n+1} equal to $\sum_{i=1}^{n} p_i$, completion

time c_0 and c_{n+1} fulfill $c_0 = p_0$, $c_{n+1} = p_{n+1}$). Incorporated the above two considerations into model, the mathematical formulation for the QCSP with Non-crossing Constraints can be strengthened as following:

$$\text{minimize} \quad c_{\max} \tag{4.11}$$

subject to

$$c_{\max} \geq c_i, \qquad \forall\, 1 \leq i \leq n \tag{4.12}$$

$$c_i - p_i \geq 0 \qquad \forall\, 1 \leq i \leq n \tag{4.13}$$

$$\sum_{k=0}^{m+1} x_{ik} = 1 \qquad \forall\, 0 \leq i \leq (n+1) \tag{4.14}$$

$$x_{00} = 1, x_{n+1,m+1} = 1 \tag{4.15}$$

$$c_i - (c_j - p_j) + M \cdot y_{ij} \geq 0 \qquad \forall\, 0 \leq i, j \leq (n+1) \tag{4.16}$$

$$(c_j - p_j) + M \cdot (1 - y_{ij}) - c_i \geq 0 \qquad \forall\, 0 \leq i, j \leq (n+1) \tag{4.17}$$

$$M \cdot (y_{ij} + y_{ji}) \geq \sum_{k=0}^{m+1} k \cdot x_{ik} - \sum_{l=0}^{m+1} l \cdot x_{jl} + 1 \qquad \forall\, 0 \leq i < j \leq (n+1) \tag{4.18}$$

$$M \cdot (y_{ij} + y_{ji}) \geq \sum_{l=0}^{m+1} l \cdot x_{jl} - \sum_{k=0}^{m+1} k \cdot x_{ik} + (i - j) \qquad \forall\, 0 \leq i < j \leq (n+1) \tag{4.19}$$

$$x_{ik}, y_{ij} \in \{0, 1\} \qquad \forall\, 0 \leq i, j \leq (n+1), \forall\, 0 \leq k \leq (m+1) \tag{4.20}$$

$$c_{\max}, c_i \in \Re^+ \qquad \forall\, 0 \leq i \leq (n+1) \tag{4.21}$$

For the QCSP with Non-crossing Constraints in Figure 4.3, by using the revised formulation, CPLEX will generate the correct optimal solution as shown in Figure 4.5. It is clear that, in Figure 4.5 even Crane 2 becomes idle in $[2, 4]$, the optimal solution generated by modified formulation of the QCSP with Non-crossing Constraints will guarantee that there is space for Crane 2 to accommodate.

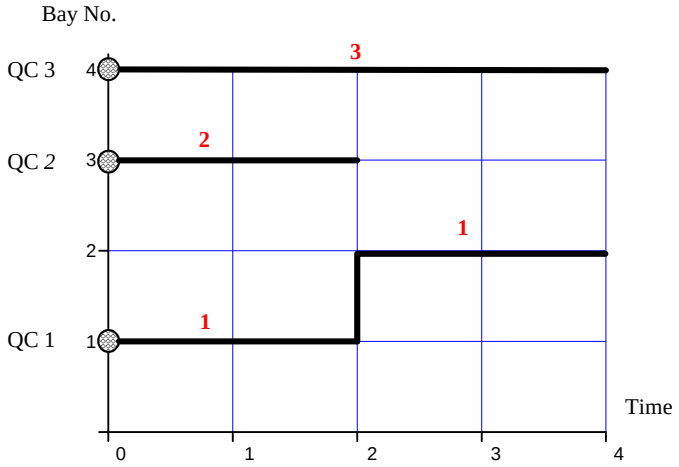


Figure 4.5: The optimal solution of modified model for the QCSP with Non-crossing Constraints

4.3 Approximation Algorithm

It has been proved that the QCSP with Non-crossing Constraints is an $\mathcal{NP}$-hard problem (Lim et al., 2007; Lee et al., 2008). To solve the QCSP with Non-crossing Constraints efficiently and effectively, several algorithms have been adopted in the literature. However, in this section, two approximation algorithms will be further studied due to the great convenience they bring to port operators in handling the QCSP with Non-crossing Constraints (Lim et al., 2004b).

4.3.1 Best Partition (BP) Method

The first approximation algorithm is called BP Method in this chapter, which tries to seek the best consecutive partition pattern for the ship bays and assigns each quay crane to the corresponding partition orderly. Specifically, BP Problem can be defined as: Given a job list $J = \{J_1, J_2, \ldots, J_{n-1}, J_n\}$, a partition of the job list separates the list into m ($m \leq n$) parts $\{J_1, \ldots, J_{i_1}\}, \{J_{i_1+1}, \ldots, J_{i_2}\}, \ldots, \{J_{i_{m-2}+1}, \ldots, J_{i_{m-1}}\}, \{J_{i_{m-1}+1}, \ldots, J_n\}$. Denote S_k ($1 \leq k \leq m$) the sum of $\{J_{i_{k-1}+1}, \ldots, J_{i_k}\}$, that is $S_k = \sum_{l=i_{k-1}+1}^{i_k} J_l$. The BP Problem is to seek the best partition pattern which will minimize $\max_{l=1}^{m} S_k$.

In Lim et al. (2004b), the authors proposed a Dynamic Programming (DP) algorithm with

S_k : the sum of workload in subsection k.
J_i : the workload of Bay i.

Figure 4.6: BP Problem

time complexity of $\mathcal{O}\left(mn^2\right)$ to solve the BP Problem. The DP equations are:

$$A[1,i] = T[1,i], \forall\, 1 \leq i \leq n \tag{4.22}$$

$$A[k,i] = \min_{k-1 \leq i_k \leq i-1} \max\left\{A[k-1,i_k], T[i_k+1,i]\right\}, \forall\, 2 \leq k \leq m, 1 \leq i \leq n \tag{4.23}$$

where $A[k,i]$ denotes the minimum latest completion time when Bays $1,2,\ldots,i$ are partitioned in an adjacent manner among Cranes $1,2,\ldots,k$; $T[i_k+1,i] = \sum_{j=i_k+1}^{i} J_j$ denotes the total processing time of jobs $i_k+1,\ldots,i$.

In fact, the $\mathcal{O}\left(mn^2\right)$ DP algorithm for BP can be improved to $\mathcal{O}\left(mn\log n\right)$ in light of the property of $A[k,i]$ shown in the Lemmas below.

Lemma 4.3.1. *Given i $(2 \leq i \leq n)$, $A[k,i] \leq A[k-1,i]$, where $2 \leq k \leq i$.*

Proof. If $k=2$, clearly, $A[2,i] \leq A[1,i]$. Hence the above inequation holds. For $k>2$, denote $P(k-1,i) = \{i_1,\ldots,i_{k-3},i_{k-2}\}$ ($i_1,\ldots,i_{k-2}$ are the partition points) one of the partition patterns when Bays $1,2,\ldots,i$ are partitioned in an adjacent manner among QCs $1,2,\ldots,(k-1)$ and $P^*(k-1,i) = \{i_1^*,\ldots,i_{k-2}^*\}$ the best partition pattern. Let $S_l^{P(k,i)}$ be the sum of workload of Partition l under Partition pattern $P(k,i)$. For $\{i_1^*,\ldots,i_{k-2}^*\}$, there exists a new partition point i_0 $(1 \leq i_0 \leq i, i_0 \neq i_1^*,\ldots,i_{k-2}^*)$. By inserting i_0 into $\{i_1^*,\ldots,i_{k-2}^*\}$, it forms a $P(k,i)= \{i_0\} \cup \{i_1^*,\ldots,i_{k-2}^*\}$. Obviously, $A[k-1,i] \geq \max_{l=1}^{k} S_l^{P(k,i)}$. Since by definition, $A[k,i] = \max_{l=1}^{k} S_l^{P^*(k,i)} \leq \max_{l=1}^{k} S_l^{P(k,i)}$, $A[k,i] \leq A[k-1,i]$. □

Lemma 4.3.1 shows that for a fixed i, the series $\{A[k,i]\}_k$ is a descending sequence.

Lemma 4.3.2. *Given k ($1 \leq k \leq m$), $A[k,i] \leq A[k,i+1]$, where $k \leq i < n$.*

Proof. For $P(k,i+1)$, it is easy to verify that either $P(k,i+1) = P(k,i)$ or $P(k,i+1) = \{i_1, \ldots, i_{k-2}, i\}$. In other words, $P(k,i+1)$ can be classified into two possible partition types: the first one is exactly as same as $P(k,i)$ and for the other pattern, the k^{th} part should be $\{J_{i+1}\}$. In the first situation, for any $P(k,i+1) = P(k,i)$, it should have $\max_{l=1}^{k} S_l^{P(k,i+1)} \geq \max_{l=1}^{k} S_l^{P(k,i)} \geq A[k,i]$. Since in this situation, this relationship will always hold for any $P(k,i+1)$, thus $A[k,i+1] = \max_{l=1}^{k} S_l^{P^*(k,i+1)} \geq A[k,i]$. On the other hand, if $P(k,i+1) = \{i_1, \ldots, i_{k-2}, i\}$, $\max_{l=1}^{k} S_l^{P(k,i+1)} = \max(J_{i+1}, \max_{l=1}^{k-1} S_l^{P(k-1,i)}) \geq \max(J_{i+1}, A[k-1,i])$. According to Lemma 4.3.1, $A[k,i] \leq A[k-1,i]$. Hence, $\max_{l=1}^{k} S_l^{P(k,i+1)} \geq A[k,i]$, for any $P(k,i+1) = \{i_1, \ldots, i_{k-2}, i\}$. Therefore, $A[k,i]$ will be less or equal to $A[k,i+1]$. □

Lemma 4.3.2 shows that given k , the serials $\{A[k,i_k]\}_{i_k}$ is a non-decreasing sequence when i_k increases (as *dcb* in Figure 4.7). Conversely, according to the definition, $\{T[i_k+1,i]\}_{i_k}$ is a non-increasing sequence when i_k increases (as *ace* in Figure 4.7). Therefore, in order to calculate $A[k,i]$ based on the recursive function (4.23), it is not necessary to compute $\max\{A[k-1,i_k], T[i_k+1,i]\}$ for all $k-1 \leq i_k \leq i-1$ (as *acb* in Figure 4.7) and then find the minimum value in the sequence of $\{\max\{A[k-1,i_k], T[i_k+1,i]\}\}_{k-1 \leq i_k \leq i-1}$. Since the minimum value can be found near the "intersection point" between series $\{A[k,i_k]\}_{i_k}$ and series $\{T[i_k+1,i]\}_{i_k}$. The optimal i_k^* should be next to the intersection point c. Consequently, a Binary Search algorithm or Golden Section Search algorithm could be embedded into the DP algorithm to facilitate the search of i_k^* for every pair (k,i). Because the time complexity for Binary Search or Golden Section Search is $\mathcal{O}(\log n)$, the time complexity of the algorithm can be reduced from $\mathcal{O}(mn^2)$ to $\mathcal{O}(mn \log n)$. The pseudo code for BP with $\mathcal{O}(mn \log n)$ is shown in Table 4.3.1.

In Table 4.3.1, $\text{sum}[i] = \sum_{a=1}^{i} J_a$ for $0 < i \leq n$. maxjob$[i]$ stores the maximum job in the job list $\{J_1, J_2, \ldots, J_i\}$. In the pseudo code, from line 5 to line 24, it is a traditional DP recursion for equations (4.22) and (4.23) except the embedded Binary Search starting from line 10 to line 20. Note that $T[i_k+1,i] = \sum_{j=i_k+1}^{i} J_j$. For the ease of computation, $T[i_k+1,i]$ is calculated by $\text{sum}[i] - \text{sum}[i_k]$. In the procedure of Binary Search (line 10 to line 21), *lp*, *mp*, and *rp* denote **left point**, **middle point**, and **right point** respectively. As shown in Figure 4.7, in order to

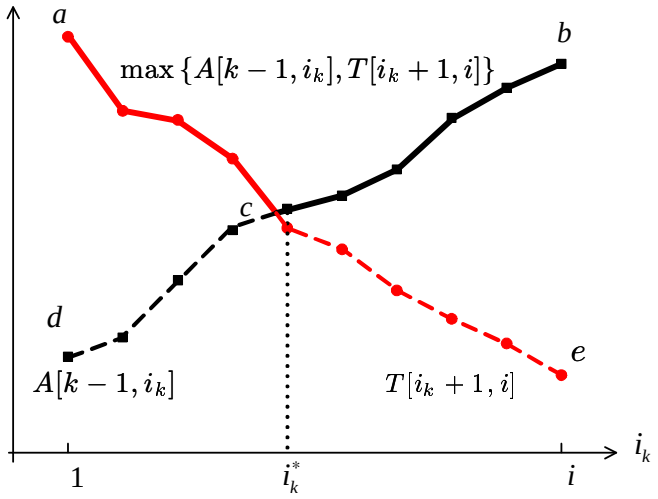


Figure 4.7: The optimal point of the recursive function

Table 4.1: The pseudo code for BP algorithm

Best Partition Method BP$[J, m]$:

```
1.  sum[0] = 0, maxjob[1] = J_1
2.  for i = 1 to n do sum[i] = sum[i − 1] + J_i end
3.  for i = 1 to n do A[1, i] = sum[i] end
4.  for i = 2 to n do maxjob[i] = max(maxjob[i − 1], J_i) end
5.  for k = 2 to m do
6.      for i = k to n do
7.          A[k, i] = +∞
8.          if (k = i) then A[k, i] = maxjob[i] end
9.          else
10.             lp = k − 1, rp = i − 1
11.             while (rp − lp) > 1 do
12.                 mp = [(lp+rp)/2]†
13.                 if (A[k − 1, mp] = (sum[i] − sum[mp])) then
14.                     rp = mp,break
15.                 else
16.                     if (A[k − 1, mp] > (sum[i] − sum[mp])) then
17.                         rp = mp
18.                     else lp = mp end
19.                 end
20.             end
21.             A[k, i] = min(A[k − 1, rp], (sum[i] − sum[lp]))
22.         end
23.     end
24. end
25. return A[m, n]
```

†:The function $[x]$ rounds the element x to the nearest integer.

Table 4.2: Comparisons between original DP and the proposed DP

Experiment No.	Size $(n \times m)$	Original DP results	CPU time (second)	Proposed DP results	CPU time (second)
1	1000×10	16071	0.531	16071	0.094
2	1000×20	8340	1.156	8360	0.188
3	1000×30	5754	1.781	5754	0.266
4	1000×40	4194	2.359	4194	0.375
5	2000×10	33247	2.281	33247	0.313
6	2000×20	16605	4.484	16605	0.406
7	2000×30	11266	7.047	11266	0.594
8	2000×40	8449	9.348	8449	0.734
9	3000×10	49329	5.078	49329	0.547
10	3000×20	25100	10.516	25100	0.813
11	3000×30	16751	14.813	16751	0.969
12	3000×40	12471	21.328	12471	1.297

find i_k^*, initially, lp is set to $k-1$, while rp is initialized as $i-1$. If $(rp-lp) > 1$, the program will proceed the Binary Search. The middle point mp is calculated by $[\frac{lp+rp}{2}]$. Therefore, after a loop (from line 11 to line 20), the searching space will roughly reduce by 50%. When $rp = lp+1$ (since $rp > lp$ and both of them are integers) or $A[k-1, mp] = \text{sum}[i] - \text{sum}[mp]$, the Binary Search will stop and according to the recursive function, $A[k, i]$ should be equal to the minimal value between $A[k-1, rp]$ and $(\text{sum}[i] - \text{sum}[lp])$.

Code the two algorithms (the original DP and DP embedded Binary Search) in Matlab and execute 12 randomly generated numerical experiments (the job lists follow uniform discrete distribution in the range $[30, 300]$) in a Pentium IV CPU 3.00GHz PC with 2.00 GB of RAM. The comparisons between the two algorithms are summarized in Table 4.3.1. It is clear that the proposed DP embedded Binary Search algorithm not only generates the same results with the original DP also reduces the computational time dramatically when the problem size expands.

4.3.2 Enhanced Best Partition (EBP) Method

In the previous section, the time complexity of BP has been reduced from $\mathcal{O}\left(mn^2\right)$ to $\mathcal{O}\left(mn \log n\right)$. However, for port operators, a primary concern is how to come up with a good approximation scheme easily which can also perfectly estimate the optimal makespan of the QCSP with Non-crossing Constraints. To achieve this goal, EBP based on both BP and an exact algorithm for

the QCSP with Non-crossing Constraints (a DP algorithm with pseudo-polynomial computational time) designed by Lim et al. (2004b) is proposed.

In Lim et al. (2004b), a pseudo-polynomial DP is proposed to solve the QCSP with Non-crossing Constraints. Since it is a pseudo-polynomial algorithm, the computational time for this DP will consume exponentially large times when the size of the problem increases. However, for the two quay cranes problem, the time complexity for this algorithm is $\mathcal{O}(nT)$, where n is the number of bays and T is an upper bound for the makespan $c_{\max}$ and set to $\sum_{i=1}^{n} J_i$ in this study. The following shows the special case for the DP algorithm when the number of quay crane is equal to 2.

$$A(i, w_2) = \min \begin{cases} \max\left[A(i-1, w_2), w_2\right] + p_i, & k = 1; \\ A(i-1, w_2 - p_i), & k = 2. \end{cases} \tag{4.24}$$

where i $(1 \leq i \leq n)$ is the index for the bays, k $(k = 1, 2)$ is the index for the two quay cranes. Let $J(i) = \{1, \ldots, i\}$ indicate the set of the leftmost i jobs. $A(i, w_2)$ denotes the minimum latest completion time of jobs in $J(i)$ processed by Crane 1, where the latest completion time of jobs in $J(i)$ processed by Crane 2 is exactly w_2. Initially, $A(0, w_2)$ is set to zero if $w_2 = 0$ and positive infinity, otherwise. The interpretation of the recursive function (4.13) is: if the bay is handled by Crane 1 ($k = 1$), then $A(i, w_2)$ equals to the processing time of Bay i (p_i) adds the maximum value between $A(i-1, w_2)$ and w_2; otherwise, when $k = 2$, $A(i, w_2)$ should be equal to $A(i-1, w_2 - p_i)$. Name the algorithm above QCSNC-2 which can solve the QCSP with Non-crossing Constraints with 2 quay cranes exactly.

The basic idea of EBP is stated in the following: since BP can efficiently generate feasible solutions (one quay crane for a partition) for the QCSP with Non-crossing Constraints and QCSNC-2 can provide an optimal solution for the QCSP with Non-crossing Constraints with 2 quay cranes within acceptable time when the problem size is relatively small, it is possible to combine the two methods to obtain a better scheduling scheme. Observing a BP solution shown in Figure 4.8, in this example, 5 quay cranes are assigned to a berthed vessel and the latest completion time is determined by the job partition handled by Crane 3, i.e., $S_3 = \max_{k=1}^{5} S_k$.

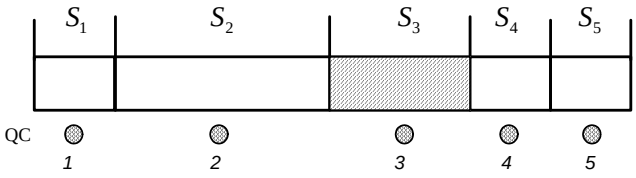


Figure 4.8: An output of BP for a QCSP with Non-crossing Constraints with 5 quay cranes

Hence, how to reduce S_3 is a key consideration in order to shorten the makespan of the vessel. A straightforward idea is to let Crane 2 (or 4) and Crane 3 work together to handle partition S_2 (or S_4) and S_3 while keep other quay cranes working on the previous assigned partitions. Here, the first problem encountered is which Crane (2 or 4) should be chosen to cooperate with Crane 3. To solve this problem, Lemma 4.3.3 can be applied to facilitate the decision making.

Lemma 4.3.3. *Let $J = \{J_1, J_2, \ldots, J_{n-1}, J_n\}$ be the job list for a berthed vessel, then the optimal makespan C^* for the QCSP with Non-crossing Constraints with m QCs should satisfy $C^* \geq {}^{(\sum_{i=1}^{n} J_i)}/_{m}$.*

Proof. Denote JS_k the job set assigned to quay crane k. According to the assumptions of the QCSP with Non-crossing Constraints, $JS_k \cap JS_l$, $\forall 1 \leq k, l \leq m, k \neq l$ and $\bigcup_{1 \leq k \leq m} JS_k = \{1, 2, \ldots, n\}$. Let $Sum_k = \sum_{i \in JS_k} J_i$. Apparently, $C^* \geq Sum_k$, $\forall 1 \leq k \leq m$. Thus, $mC^* \geq \sum_{k=1}^{m} Sum_k$. Because $\sum_{k=1}^{m} Sum_k = \sum_{i=1}^{n} J_i$, C^* should be greater than ${}^{(\sum_{i=1}^{n} J_i)}/_{m}$. □

Lemma 4.3.3 indicates that given a job list, ${}^{(\sum_{i=1}^{n} J_i)}/_{m}$ is a lower bound for the optimal makespan of the QCSP with Non-crossing Constraints with m quay cranes. Back to the previous question, in light of Lemma 4.3.3, if $S_2 \geq S_4$, combining 3rd and 4th partition together (call it P_{3-4}) and assigning Crane 3 and Crane 4 to P_{3-4} is more possible to achieve a lower completion time for the QCSP compared to the plan that assigning Crane 2 and Crane 3 to P_{2-3} . More specifically, if P_{3-4} is adopted, the makespan of this approximation scheme could be shorter than S_2; otherwise, if P_{2-3} is adopted, the makespan of this approximation must be larger than S_2. Therefore, in this case, Crane 4 should be chosen to cooperate with Crane 3. Input P_{3-4} into QCSNC-2, an improved scheduling plan can be generated.

Besides, Lemma 4.3.3 also suggests that to obtain a better approximation scheme, it is not

Table 4.3: The pseudo code for EBP algorithm

Enhanced Best Partition Method EBP$[J, m]$:

1. call BP$[J, m]$, return the best partition pattern and $\{S_k\}_{1 \le k \le m}$
2. calculate $AT = (\sum_{i=1}^{n} J_i)/m$
3. $\Phi = \emptyset$, $\Psi = \emptyset$.
4. **for** $k = 1$ to m **do**
5. **if** $(S_k > AT)$ **then**
6. $\Phi = \Phi \cup \{k\}$
7. **end**
8. **end**
9. **while** $\Phi \neq \emptyset$ **do**
10. find the index of k $(k \in \Phi, S_k = \max_{i \in \Phi} S_i)$
11. search the partition with minimum sum of workload $S_l (l = k - 1$ or $k + 1$,
12. if $k = 1$, $l = 2$; if $k = m$, $l = m - 1$; besides, $l \notin \Psi$).
13. **if** (there is not feasible l) **then**
14. $\Phi = \Phi \setminus \{k\}^{\dagger}$
15. **else**
16. call QCSNC-2(P_{l-k}),
17. $\Phi = \Phi \setminus \{k, l\}$, $\Psi = \Psi \cup \{k, l\}$
18. **end**
19. **if** $(\Phi = \emptyset)$ **then**
20. break
21. **end**
22. **end**
23. **return** the makespan of the approximation scheme.

†: for set A, B, $A \setminus B = \{x \mid x \in A$ and $x \notin B\}$

necessary trying to improve the partition grouping plan P_{k-k+1} $\left(\frac{S_k + S_{k+1}}{2} \le \frac{\sum_{i=1}^{n} J_i}{m}\right)$, $1 \le k \le m - 1$ by QCSNC-2. Based on the aforementioned findings, the pseudo-code for EBP algorithm is shown in Table 4.3.2.

In Table 4.3.2, firstly, EBP calls BP to get the best partition pattern for the input job list. According to Lemma 4.3.3, the lower bound $(\sum_{i=1}^{n} J_i)/m$ is computed and stored in AT. Φ is a index set $\Phi = \{k \mid S_k >$ AT$\}$. Ψ is another index set used to store the partition numbers which have been combined together. Initially, Φ and Ψ are empty sets. From Line 4 to Line 8, all the partitions are examined and their index are added into set Φ if the sum of workload in the partitions are larger than the lower bound AT. In the loop to decide which two partitions will be combined, firstly, Partition k should be identified as long as S_k is the maximum value in $\{S_i\}_{i \in \Phi}$. Next, after examining the neighbor partitions around Partition k, the one with relatively smaller sum of workload will be chosen. Note that the neighbor partition number should not be an element of set Ψ. If there is not a feasible neighbor partition, separate k from

set Φ and if Φ is not empty, execute another search. However, if a feasible partition can be found (denote it as partition l), the combination of Partitions l and k (denote as P_{l-k}) will be fed into QCSNC-2. Then delete l and k from Φ (l may not be an element of Φ, in this case just delete k) and add them into Ψ.

To show the performance of EBP, 24 experiments were conducted. For each instance, the job list was randomly generated. That is, the workload in each bay followed a uniform discrete distribution in the range of $[30, 300]$. The first 20 instances are relatively small size problems compared with last 4 experiments. To assess the effectiveness of BP and EBP, the revised model of the QCSP with Non-crossing Constraints in Section 4.2 was also coded in CPLEX and executed in a Pentium IV CPU 3.00GHz PC with 2.00 GB of RAM. Due to the intractability of the QCSP with Non-crossing Constraints, for large scale problems like the experiment in this study, CPLEX usually failed to provide the final solutions. CPLEX was run for 1 hour for each experiment. Meanwhile, EBP were coded in Matlab. In Table 4.3.2, the computational results of EBP are presented. In the third column, AT defined as $(\sum_{i=1}^{n} J_i)/m$ is shown, which according to Lemma 4.3.3, is a lower bound for the the QCSP with Non-crossing Constraints with m quay cranes. The 4^{th} column is the partial solutions generated by CPLEX after 1 hour searching. For experiment from 21 to 24, CPLEX fails to generate any solution within one hour. Columns 5 and 9 are the results of BP and EBP respectively. Gap^1_{BP} in Column 6 $\text{Gap}^1_{\text{EBP}}$ in Column 10 are the indicators to measure the effectiveness of BP and EBP using lower bound AT as reference. For all the 24 randomly generated instances, the average difference between BP's result and AT is 10% while EBP achieves better outcome since the average $\text{Gap}^1_{\text{EBP}}$ is nearly half of Gap^1_{BP}. In Columns 7 and 11, Gap^2_{BP} and $\text{Gap}^2_{\text{EBP}}$ show the performances of BP and EBP compared with CPLEX in terms of the solutions found. It is clear that for larger instances (from experiment 17 to 24), running CPLEX merely 1 hour is not enough to obtain better solutions compared with EBP even BP. Besides, both BP and EBP are very efficient (see Columns 8 and 12). As a comparison, to finish the search of Instance 2 (16 bays and 5 cranes), it takes 25685 seconds (roughly 7 hours) for CPLEX to find the global optimal value 586.

Figure 4.9 and Figure 4.10 show the Gantt Chart for Instance 2 (16 bays and 5 cranes) scheduled by BP and EBP respectively. In Figure 4.9, it is shown that BP separates the job

list into 5 partitions[1]: {221_1, 174_2, 187_3}, {108_4, 111_5, 239_6}, {33_7, 236_8, 250_9}, {222_{10}, 42_{11}, 142_{12}, 263_{13}}, and {244_{14}, 131_{15}, 290_{16}}. The sum of workload of each partition is: $S_1 = 582$, $S_2 = 458$, $S_3 = 519$, $S_4 = 669$, and $S_5 = 665$. Since AT = 578.6, only Part 3 and Part 4 will be combined for QCSNC-2. In Figure 4.10, Crane 3 and Crane 4 cooperate with each other to handle P_{3-4}. After complete the job in Bay 9, Crane 3 will proceed to Bay 11. While for Crane 4, at time 222, it will move from Bay 10 directly to Bay 12. It can be checked that no quay cranes violate the Non-crossing Constraints at any instant.

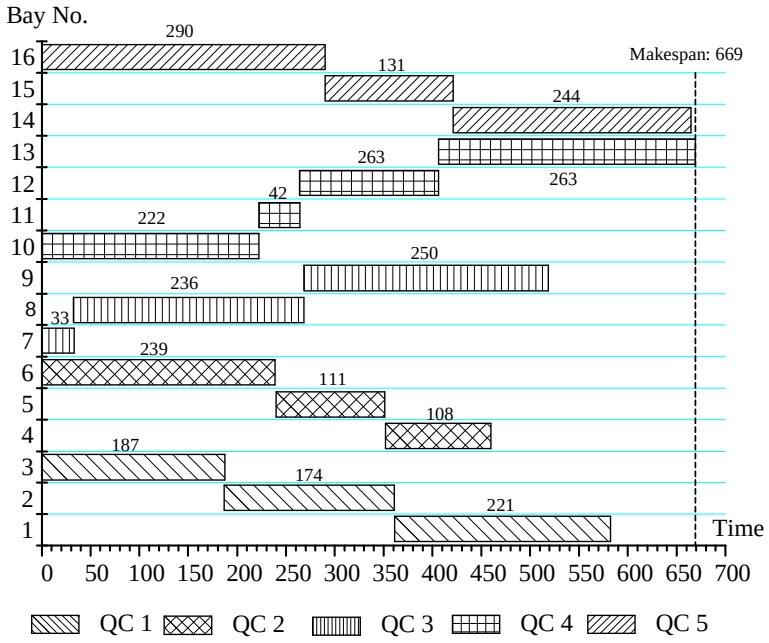


Figure 4.9: Gantt chart for Instance 2 scheduled by BP

[1]In the following, 221_1 means the workload in Bay 1 is 221. The rest can be interpreted by analogy.

Table 4.4: Computational results for BP and EBP

Experiment No.	Size (bays×cranes)	AT†	CPLEX‡	BP				EBP			
				Results	Gap^1_{BP}(%) §	Gap^2_{BP}(%) ¶	Time(s)	Results	Gap^1_{EBP}(%) ††	Gap^2_{EBP}(%) ‡‡	Time(s)
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
1	16 × 4	723.25	726	801	10.75	10.33	<0.1	781	7.98	7.58	1.766
2	16 × 5	578.6	610	669	15.62	9.67	<0.1	665	14.93	9.02	0.828
3	17 × 4	735.25	746	800	8.81	7.24	<0.1	761	3.50	2.01	1.688
4	17 × 5	588.2	604	665	13.06	10.10	<0.1	636	8.13	5.30	1.688
5	18 × 4	714	737	790	10.64	7.19	<0.1	745	4.34	1.09	2.563
6	18 × 5	571.2	595	628	9.94	5.55	<0.1	619	8.37	4.03	0.750
7	19 × 4	701	728	826	17.83	13.46	<0.1	711	1.43	-2.34	2.688
8	19 × 5	560.8	580	611	8.95	5.34	<0.1	584	4.14	0.69	1.578
9	20 × 4	922	949	980	6.29	3.27	<0.1	979	6.18	3.16	2.016
10	20 × 5	737.6	787	803	8.87	2.03	<0.1	781	5.88	-0.76	2.313
11	21 × 4	758.25	806	832	9.73	3.23	<0.1	801	5.64	-0.62	1.969
12	21 × 5	606.6	651	658	8.47	1.08	<0.1	622	2.54	-4.45	2.109
13	22 × 4	756.75	804	824	8.89	2.49	<0.1	766	1.22	-4.73	3.422
14	22 × 5	605.4	636	680	12.32	6.92	<0.1	643	6.21	1.10	1.625
15	23 × 4	886	981	924	4.29	-5.81	<0.1	910	2.71	-7.24	4.109
16	23 × 5	708.8	765	814	14.84	6.41	<0.1	740	4.40	-3.27	2.750
17	24 × 4	857.5	928	921	7.41	-0.75	<0.1	874	1.92	-5.82	4.203
18	24 × 5	686	812	755	10.06	-7.02	<0.1	712	3.79	-12.32	2.875
19	25 × 4	1083.5	1253	1167	7.71	-6.86	<0.1	1129	4.20	-9.90	5.594
20	25 × 5	866.8	987	982	13.29	-0.51	<0.1	921	6.25	-6.69	1.844
21	50 × 8	998.25	§§–	1084	8.59	–	<0.1	1046	4.78	–	7.42
22	50 × 10	798.6	–	898	12.45	–	<0.1	897	12.32	–	14.91
23	100 × 8	2064.88	–	2151	4.17	–	<0.1	2124	2.86	–	4.23
24	100 × 10	1651.9	–	1768	7.03	–	<0.1	1747	5.76	–	10.06
Average					10.00	3.67			5.40	-1.21	

†: AT = $(\sum_{i=1}^{n} J_i)/m$. According to Lemma 4.3.3, AT is a lower bound for the QCSP with Non-crossing Constraints.

‡: The partial solutions obtained by CPLEX after 1 hour.

§: Gap^1_{BP}=(solution obtained from BP-AT)×100/ AT.

¶: Gap^2_{BP}=(solution obtained from BP-CPLEX solution)×100/ CPLEX solution.

††: Gap^1_{EBP}=(solution obtained from EBP-AT)×100/ AT.

‡‡: Gap^2_{EBP}=(solution obtained from EBP-CPLEX solution)×100/ CPLEX solution.

§§: CPLEX fails to generate any solution within 1 hour.

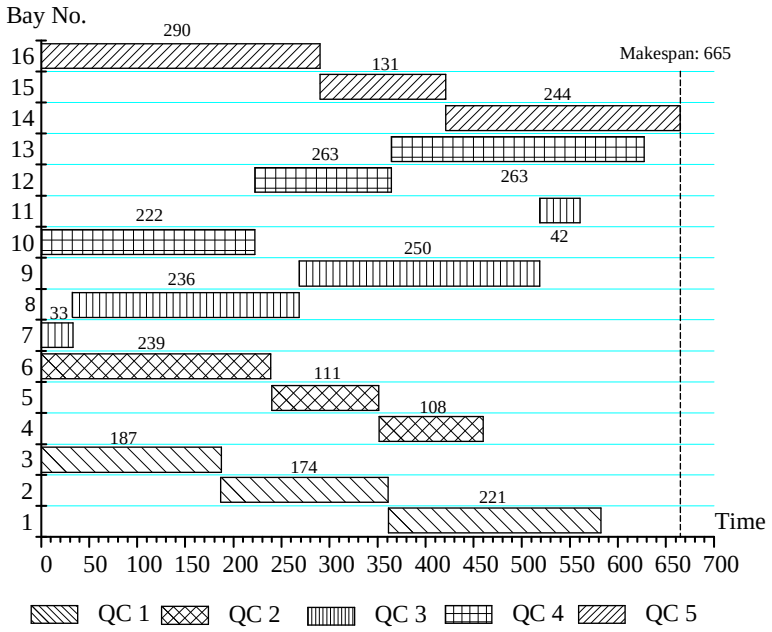


Figure 4.10: Gantt chart for Instance 2 scheduled by EBP

4.4 Summary

The common deficiencies found in the previous studies in modeling the QCSP with Non-crossing Constraints in port container terminals are identified. Remedies for the problems are proposed to prevent the occurrences of unrealistic optimal solutions for the problem. Additionally, two approximation schemes for the QCSP with Non-crossing Constraints are studied. The first one is called BP Method, which can be solved by DP algorithm with time complexity of $\mathcal{O}\left(mn^2\right)$. A more efficient algorithm is designed to improve the original DP by adopting Binary Search. Next, based on BP and an exact algorithm for the QCSP with Non-crossing Constraints (with pseudo-polynomial computational time), the second approximation scheme named EBP Method is proposed to achieve a better estimation of the optimal makespan for the QCSP with Non-crossing Constraints. The results of numerical experiments show that EBP is an effective approximation scheme for the problem.

Chapter 5

Heuristics for Quay Crane Scheduling at Indented Berth

5.1 Background

In a long run, due to the increasing volume of worldwide container traffic and the severe freight rate competition, the number of mega-containerships will continue to rise in this global environment. Nevertheless, to provide a fast service of mega-containerships brings up new and arduous challenges for container terminal operators.

Figure 5.1: Hybrid quay wall proposed for the west terminal of Busan New Port

Being one of the strategies to cope with the looming conundrum, indented berth which is designed to increase the ship-to-shore interface, has been adopted in practice. The world's first indented berth was implemented in the Ceres Paragon terminal at Port of Amsterdam in 2001, which enabled container vessels to be handled simultaneously from both sides. It is reported that through the indented berth, up to as many as nine quay cranes can be deployed to a berthing vessel. Therefore, shorter handling time for the vessel is expected. The recent technical update for the indented berth is the invention of hybrid quay wall. Hybrid quay wall is a floating mobile quay with dimension of 480m×160m×6m (Chae et al., 2008). Equipped with thrusters, a hybrid quay wall can travel the distance of two or three berths. Therefore, in combination with an existing land based berth, a hybrid quay wall can dynamically construct an indented berth and enhance the flexibility and productivity of container terminals. The concept of hybrid quay wall has been proposed for the west terminal of the Busan New Port in Korea (see Figure 5.1).

The indented berth is designed to shorten the time necessary to load and unload vessels by quay cranes. Compared with standard berth, the QCSP at indented berth possesses the following unique features:

1. At indented berth, quay cranes are able to handle a vessel simultaneously from both sides of the vessel. Due to the design of quay cranes (the arm of a quay crane can lift up when it is going to pass the quay cranes at the opposite side), the quay cranes at different sides of the indented berth are free of the non-crossing constraints. However, lifting up of the arm of a crane will introduce an additional idle time for the crane.

2. The handling time of a task (including processing time and transportation time) depends on the quay crane that it is assigned to. Here a task is referred to the loading or unloading operations of a group of containers belonging to the same bay (Kim and Park, 2004). For example, as shown in Figure 5.1, suppose the designated storage yard location for a cluster of containers located at Bay a, is Bay b. In terms of transportation time, assigning the task to Crane 1 is probably more beneficial compared with the alternative plan by assigning it to Crane 2.

In this Chapter, the QCSP at indented berth will be studied by taking its two aforementioned unique features into account. However, it is assumed that the time lost introduced by crane arm lifting up is small enough compared with the operational time of the crane and thus the cost is not considered.

5.2 Problem Description and Mathematical Formulation

As depicted in Figure 5.2, it is a convention in this chapter that: the ship bay of a vessel is numbered increasingly from left to right; the quay cranes at Berth Side A are numbered increasingly from left to right while the quay cranes at Berth Side B are numbered immediately in the same manner thereafter. Let $\Omega = \{1, 2, \ldots, n\}$ denote the set of tasks (i.e., discharging or loading operations for container clusters, Kim and Park (2004)). For each task $i \in \Omega$, it is located at Bay l_i and its processing time is p_{ik} if Task i is assigned to the Crane k. Additionally, define two dummy Tasks 0 and $T = n + 1$ with processing time $p_{0k} = p_{Tk} = 0$ to indicate the beginning and the end of the service of the vessel. Let $\Omega^0 = \Omega \cup \{0\}$, $\Omega^T = \Omega \cup \{T\}$, and $\overline{\Omega} = \Omega \cup \{0, T\}$. Furthermore, let Φ denote the set of precedence constrained task pairs and Ψ be the set of pairs of tasks that cannot be performed simultaneously. Obviously, $\Phi \subseteq \Psi$.

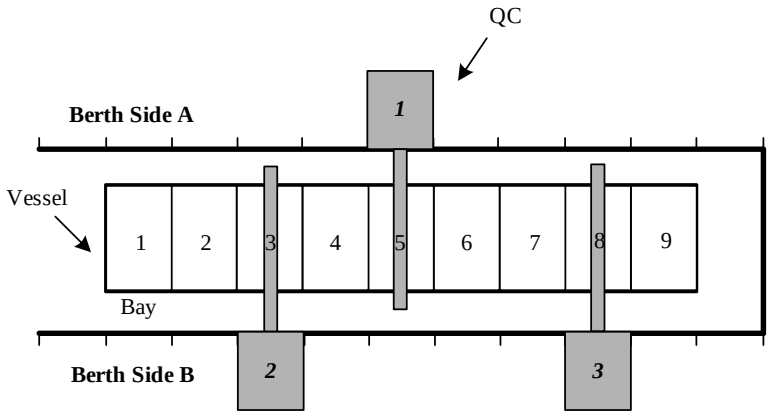


Figure 5.2: Schema for QCSP at indented berth

$Q_a = \{1, \ldots, q_a\}$ and $Q_b = \{q_a + 1, \ldots, q\}$ are the sets of quay cranes at Berth Side A and Berth Side B, respectively. Define the set of all quay cranes $Q = Q_a \cup Q_b$. Each quay crane $k \in Q$, has a starting position l_0^k and a final position l_T^k. It is assumed that all the quay cranes

travel in an identical speed and the travel time for them to move between two adjacent ship bays is $\hat{t}$. The travel time between bay position l_i and l_j $(i, j \in \Omega)$ is defined as $t_{ij} = \hat{t} \cdot |l_i - l_j|$. Let t_{0j}^k and t_{jT}^k respectively represent the travel time from the initial position (l_0^k) of Crane k to location of Task j (l_j), and from location of Task j to the final destination of Crane k (l_T^k). Besides, due to the safety requirement, any two quay cranes should maintain a safety distance δ, measured in units of ship bays.

As pointed out by Bierwirth and Meisel (2009), in order to prevent the crane interference, necessary time span between the execution of tasks by any two quay cranes should be considered. Define Δ_{ij}^{vw} as the minimum time span to elapse between the processing of two Tasks i and j, if processed by Cranes v and w, respectively. Let $\delta_{vw} = (\delta + 1) \cdot |v - w|$ be the smallest allowed difference between the bay positions of Cranes v and w that are at the same side of the berth (i.e., $v, w \in Q_a$ or $v, w \in Q_b$). Then for all combinations of Tasks $i, j \in \Omega$ $(i \neq j)$ and Cranes $v, w \in Q$ $(v \neq w)$, the minimum temporal distance Δ_{ij}^{vw} can be quantified as:

$$\Delta_{ij}^{vw} = \begin{cases} (l_i - l_j + \delta_{vw}) \cdot \hat{t}, & \text{if } v < w,\ v, w \in Q_a \text{ or } Q_b \text{ and } l_i > l_j - \delta_{vw}; \\ (l_j - l_i + \delta_{vw}) \cdot \hat{t}, & \text{if } v > w,\ v, w \in Q_a \text{ or } Q_b \text{ and } l_i < l_j + \delta_{vw}; \\ (l_i - l_j + \delta + 1) \cdot \hat{t}, & \text{if } v, w \text{ are not at the same side of berth and} \\ & l_j \leq l_i \leq l_j + \delta; \\ (l_j - l_i + \delta + 1) \cdot \hat{t}, & \text{if } v, w \text{ are not at the same side of berth and} \\ & l_j - \delta \leq l_i \leq l_j; \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

The first two cases of (5.1) have been explained in Bierwirth and Meisel (2009). The third and forth cases in (5.1) are unique for the QCSP at indented berth. As mentioned before, the quay cranes at different side of indented berth are free from the Non-crossing Constraints. However, in order to reduce the risk of collision, the safety distance requirement remains valid for any two quay cranes. Without loss of generality, suppose Task j is processed prior to Task i and $v \in Q_a$, $w \in Q_b$. If the location of Task i is outside of the safety zone of Task j (i.e., $l_i < l_j - \delta$ or $l_i > l_j + \delta$), since Tasks i and j are possible to be processed simultaneously, $\Delta_{ij}^{vw} = 0$. For

the case when $l_j \leq l_i \leq l_j + \delta$ (as depicted in Figure 5.3), there should be a minimum temporal distance between the execution of Tasks j and i to guarantee the safety distance requirement. However, this minimum temporal distance is a function of the realized quay crane routing plan. For instance, after completion of Task j, if Crane w moves in the direction from Bay l_j to Bay b, then $\Delta_{ij}^{vw} = (l_j - l_i + \delta + 1) \cdot \hat{t}$; if quay crane w moves in the direction from bay l_j to bay a, $\Delta_{ij}^{vw} = (l_i - l_j + \delta + 1) \cdot \hat{t}$ instead. Because it is impossible to know the realized quay crane routing plan beforehand, a compromised strategy is to keep a redundant temporal distance between the execution of the two tasks. That is, $\Delta_{ij}^{vw} = \max\{(l_i - l_j + \delta + 1) \cdot \hat{t}, (l_j - l_i + \delta + 1) \cdot \hat{t}\}$. Since $l_i \geq l_j$, $(l_i - l_j + \delta + 1) \cdot \hat{t} \geq (l_j - l_i + \delta + 1) \cdot \hat{t}$. Therefore, in the case when $l_j \leq l_i \leq l_j + \delta$, Δ_{ij}^{vw} can be set to $(l_i - l_j + \delta + 1) \cdot \hat{t}$.

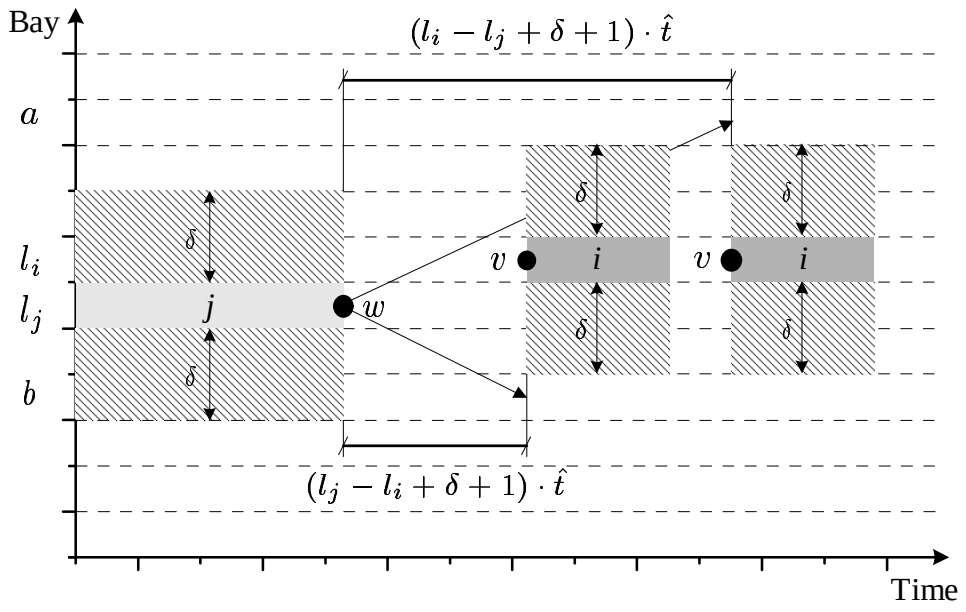


Figure 5.3: Minimum time span between the execution of tasks by quay cranes at different side of berth

When $l_j - \delta \leq l_j \leq l_j$, similar analysis presented above will suggest that $\Delta_{ij}^{vw} = (l_j - l_i + \delta + 1) \cdot \hat{t}$. For the scenarios when task i is processed prior to Task j, it can be shown that the redundant Δ_{ij}^{vw} can also be expressed in the way described in the third and forth cases of (5.1). Note that although for the third and forth cases of (5.1), the choice of Δ_{ij}^{vw} can be conservative, the redundance of Δ_{ij}^{vw} is sufficiently small provided that δ is usually set to 1 or 2 in practice and

$\hat{t}$ is usually within one minute. Hence, the impact of this conservative strategy on the QCSP should be marginal.

After calculating the value of Δ_{ij}^{vw} in advance, the set of all combinations of tasks and quay cranes that potentially lead to crane interference, Θ, can be defined as $\Theta = \{(i,j,v,w) \in \Omega^2 \times Q^2 \,|\, (i<j) \wedge (\Delta_{ij}^{vw} > 0)\}$. The sections below show the mathematical formulation for the proposed problem.

5.2.1 Decision variables

- $x_{ij}^k \in \{0,1\}$ $(i,j \in \overline{\Omega}, k \in Q)$, is 1 if and only if Tasks i and j are performed consecutively by Crane k
- $z_{ij} \in \{0,1\}$ $(i,j \in \Omega)$, is equal to 1 if and only if Task j starts after the completion time of Task i
- c_i $(i \in \overline{\Omega})$, the completion time of Task i
- p_i $(i \in \overline{\Omega})$, the realized handling time for Task i. By default, $p_0 = p_T = 0$.

5.2.2 MIP model

$$\min \quad c_T \tag{5.2}$$

s.t.

$$\sum_{j \in \Omega^T} x_{0j}^k = 1, \qquad (k \in Q) \tag{5.3}$$

$$\sum_{j \in \Omega^0} x_{jT}^k = 1, \qquad (k \in Q) \tag{5.4}$$

$$\sum_{j \in \Omega^0} x_{ji}^k - \sum_{j \in \Omega^T} x_{ij}^k = 0, \qquad (i \in \Omega, k \in Q) \tag{5.5}$$

$$\sum_{k \in Q} \sum_{j \in \Omega^T} x_{ij}^k = 1, \qquad (i \in \Omega) \tag{5.6}$$

$$p_i = \sum_{k \in Q} \sum_{u \in \Omega^0} x_{ui}^k \cdot p_{ik}, \qquad (i \in \Omega) \tag{5.7}$$

$$c_i + t_{ij} + p_{jk} - c_j \leq M(1 - x_{ij}^k), \qquad (i, j \in \Omega, k \in Q) \tag{5.8}$$

$$c_i + p_j - c_j \leq 0, \qquad ((i, j) \in \Phi) \tag{5.9}$$

$$c_i + p_j - c_j \leq M(1 - z_{ij}), \qquad (i, j \in \Omega) \tag{5.10}$$

$$c_j - p_j - c_i \leq M z_{ij}, \qquad (i, j \in \Omega) \tag{5.11}$$

$$z_{ij} + z_{ji} = 1, \qquad ((i, j) \in \Psi) \tag{5.12}$$

$$\sum_{u \in \Omega^0} x_{ui}^v + \sum_{u \in \Omega^0} x_{uj}^w \leq 1 + z_{ij} + z_{ji}, \quad ((i, j, v, w) \in \Theta \wedge v, w \in Q_a) \tag{5.13}$$

$$\sum_{u \in \Omega^0} x_{ui}^v + \sum_{u \in \Omega^0} x_{uj}^w \leq 1 + z_{ij} + z_{ji}, \quad ((i, j, v, w) \in \Theta \wedge v, w \in Q_b) \tag{5.14}$$

$$c_i + \Delta_{ij}^{vw} + p_j - c_j \leq M(3 - z_{ij} - \sum_{u \in \Omega^0} x_{ui}^v - \sum_{u \in \Omega^0} x_{uj}^w), \quad ((i, j, v, w) \in \Theta) \tag{5.15}$$

$$c_j + \Delta_{ij}^{vw} + p_i - c_i \leq M(3 - z_{ji} - \sum_{u \in \Omega^0} x_{ui}^v - \sum_{u \in \Omega^0} x_{uj}^w), \quad ((i, j, v, w) \in \Theta) \tag{5.16}$$

$$t_{0j}^k + p_{jk} - c_j \leq M(1 - x_{0j}^k), \qquad (j \in \Omega, k \in Q) \tag{5.17}$$

$$c_j + t_{jT}^k - c_T \leq M(1 - x_{jT}^k), \qquad (j \in \Omega, k \in Q) \tag{5.18}$$

$$c_i, p_i \in \Re^+, \qquad (i \in \overline{\Omega}) \tag{5.19}$$

$$x_{ij}^k \in \{0, 1\}, \qquad (i, j \in \overline{\Omega}, k \in Q) \tag{5.20}$$

$$z_{ij} \in \{0, 1\}, \qquad (i, j \in \Omega) \tag{5.21}$$

The objective of the QCSP at indented berth is to minimize the completion time of the final Task T (i.e., the makespan). Constraints (5.3-5.5) are the classical flow conservation equations. Constraints (5.6) make sure that each task can be performed by one and only one crane. Constraints (5.7) calculate the real handling time for task i, $i \in \Omega$. Constraints (5.8), (5.17), and (5.18) compute the task completion times for all the tasks in $\overline{\Omega}$. If Task j should be processed prior to Task i, (i.e., $(i, j) \in \Phi$), Constraints (5.9) guarantee the precedence relationship. Constraints (5.10-5.11) define the variable z_{ij}. And Constraints (5.12) ensure

that Tasks i and j cannot be performed simultaneously if $(i,j) \in \Psi$. In Constraints (5.13), $\sum_{u\in\Omega^0} x_{ui}^v = 1$ if and only if Task i is processed by Crane v, and similarly, $\sum_{u\in\Omega^0} x_{uj}^w = 1$ if and only if Task j is processed by Crane w. If both assignments realize, the left-hand side equals to 2 and the inequality guarantees that $z_{ij} + z_{ji}$ cannot be 0, i.e., Tasks i and j are not operated simultaneously. Therefore, Constraints (5.13) are the Non-crossing Constraints for the Cranes v and w, $(v, w \in Q_a)$. Similarly, Constraints (5.14) prevent the crossing between Cranes v and w, for $v, w \in Q_b$. Constraints (5.15) and (5.16) insert the minimum temporal distance between the completion time of one task and the starting time of another task. Finally, Constraints (5.19-5.21) specify the domains for the decision variables.

5.3 Heuristic Solution Procedure

In this section, a new decomposition heuristic framework will be introduced to find satisfactory solution for QCSP at indented berth. Similar to Sammarra et al. (2007), QCSP is broken down into two consecutive subproblems, i.e., assigning subproblem and scheduling subproblem, respectively. As mentioned before, the assigning subproblem is to partition the set of tasks Ω into $|Q|$ mutually exclusive but collectively exhaustive subsets and assign them to the $|Q|$ quay cranes. Consecutively, scheduling subproblem needs to be solved to determine the sequence and starting times for all the tasks in each subset by considering all sorts of constraints and requirements. Compared with the assigning subproblem, scheduling subproblem is essentially more complex. Therefore, in the sequel, a sophisticated but efficient algorithm to solve scheduling subproblem will be introduced at first.

5.3.1 Heuristic for scheduling subproblem

The disjunctive graph is applied to represent the scheduling subproblem by delineating Task i, $i \in \overline{\Omega}$ as a node, known task sequence relationship as a set of conjunctive arcs, and undetermined task sequence relationship as a set of disjunctive arcs. Then to solve a scheduling subproblem for the QCSP at indented berth is equivalent to find a feasible orientation of disjunction arcs to minimize the longest path length of the corresponding disjunctive graph (Sammarra et al.,

2007).

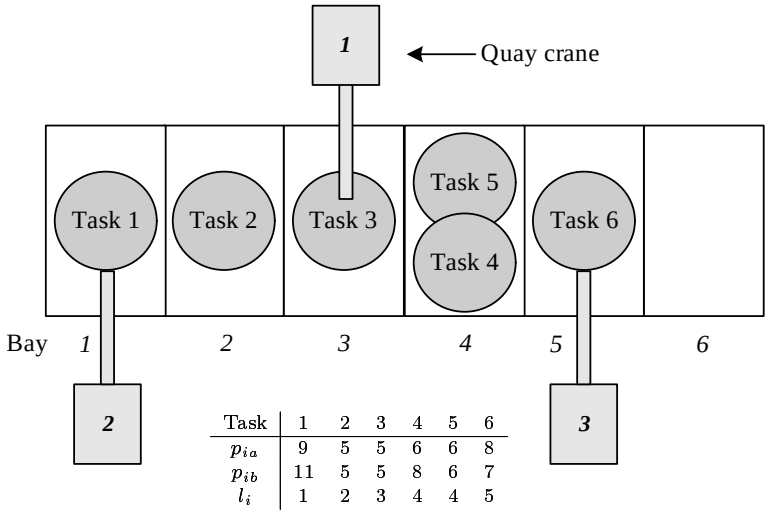
Given a scheduling subproblem, let the disjunctive graph $\mathcal{G} = (\overline{\Omega}, C, D, W_{\mathcal{V}}, W_{\mathcal{E}})$ be its corresponding geometrical representation. Here $\overline{\Omega}$ is the node set for the graph, C and D represent the sets of conjunctive arcs and disjunctive arcs respectively, while $W_{\mathcal{V}} = [w_i]$, $i \in \overline{\Omega}$ represents the node weights and $W_{\mathcal{E}} = [w_{ij}]$, $i \in \Omega^0, j \in \Omega^T, i \neq j$ represents the arc weights. As stated above, scheduling subproblem receives the output of its preceding subproblem, i.e., assigning subproblem. Assume that task subset Ω_v, $v \in Q$ is assigned to Crane v such that $\Omega_v \cap \Omega_w = \emptyset$ $(v, w \in Q, v \neq w)$ and $\bigcup_{v \in Q} \Omega_v = \Omega$. Define a set Υ_v, $v \in Q$, as $\Upsilon_v = \{(i,j) \in \Omega^2 \setminus \Phi \,|\, i,j \in \Omega_v, i < j\}$. Let $\Upsilon = \bigcup_{v \in Q} \Upsilon_v$, which is the set of disjunctive arcs that connect nodes (tasks) belonging to the same task subset. Apparently, $\Upsilon \subseteq D$. There is another type of disjunctive arcs: the arcs for task pairs which are not precedence constrained but can cause crane interference under the given task assignment, $\{\Omega_v\}_{v \in Q}$. Specifically, the latter disjunctive arcs set can be defined as $\Lambda = \{(i,j) \in \Omega^2 \setminus \Phi \,|\, (i,j,v,w) \in \Theta, v, w \in Q, v \neq w, i \in \Omega_v, j \in \Omega_w\}$. Therefore, the set of all disjunctive arcs for graph $\mathcal{G}$, $D = \{(i,j) \in \Omega^2 \,|\, (i,j) \in \Upsilon \cup \Lambda\}$. For the case of conjunctive arcs, they represent the known task sequence relationship. Let $\langle i,j \rangle$, $i \in \Omega^0, j \in \Omega^T, i \neq j$, be an element of C, which means that Node i is the direct predecessor of Node j in graph $\mathcal{G}$. To construct the set of conjunctive arcs C, add $\langle i,j \rangle$ into C if and only if (i,j) belongs to Φ. Note that in a disjunctive graph, the notation $\langle i,j \rangle$ stands for a conjunctive arc with direction from Node i to Node j; while the notation (i,j) represents a disjunctive arc connecting Node i and Node j.

Since the task assignment has been determined beforehand, the realized handling time for Task i, $i \in \Omega$, p_i is known for the scheduling subproblem. The weight for a node in $\mathcal{G}$ is equivalent to its realized handling time (i.e., $w_i = p_i$). Hence, $W_{\mathcal{V}} = [p_i]$, $i \in \overline{\Omega}$. For $W_{\mathcal{E}}$, the weights of arcs are defined in the following:

$$w_{ij} = \begin{cases} \max\{t_{ij}, \Delta_{ij}^{vw}\}, & v, w \in Q,\ i \in \Omega_v,\ j \in \Omega_w; \\ t_{0i}^{v}, & v \in Q,\ i = 0,\ j \in \Omega_v; \\ t_{iT}^{v}, & v \in Q,\ i \in \Omega_v,\ j = T. \end{cases} \tag{5.22}$$

To facilitate the understanding on the transformation from a scheduling subproblem to a disjunctive graph, a simple example will be used for illustration. As shown in Figure 5.4, three quay cranes ($Q_a = \{1\}$, $Q_b = \{2, 3\}$) are designated to serve a vessel with six bays. In Figure 5.4, $p_{ik} = p_{ia}$, $\forall\, k \in Q_a$ and $p_{ik} = p_{ib}$, $\forall\, k \in Q_b$. There are six tasks distributed in the ship bays of the vessel. Suppose the task assignment is $\Omega_1 = \{3, 4\}$, $\Omega_2 = \{1, 2\}$, $\Omega_3 = \{5, 6\}$ and $\hat{t} = 0.1$, $\delta = 1$, $l_0^1 = l_T^1 = 3$, $l_0^2 = l_T^2 = 1$, $l_0^3 = l_T^3 = 5$. Then it can be derived that $p_1 = 11$, $p_2 = 5$, $p_3 = 5$, $p_4 = 6$, $p_5 = 6$, $p_6 = 7$ and according to Formula (5.1) and the realized task assignment plan, except $\Delta_{32}^{12} = \Delta_{23}^{21} = \Delta_{35}^{13} = \Delta_{53}^{31} = \Delta_{46}^{13} = \Delta_{64}^{31} = 0.3$, $\Delta_{45}^{13} = \Delta_{54}^{31} = 0.2$, the rest of $\Delta_{ij}^{vw} = 0$. Additionally, $\Phi = \{(0,1), (0,2), (0,3), (0,4), (0,5), (0,6), (4,5), (1,T), (2,T), (3,T), (4,T), (5,T), (6,T)\}$, $\Psi = \Phi \cup \{(2,3), (3,5), (4,6)\}$, and $\Theta = \{(2,3,2,1), (3,5,1,3), (4,6,1,3), (4,5,1,3)\}$.

Figure 5.5 shows the corresponding disjunctive graph for the scheduling subproblem for the example in Figure 5.4, where the arcs with arrow are the conjunctive arcs and the dotted arcs are disjunctive arcs. The weights on both nodes and arcs are also highlighted in Figure 5.5. The following describes the steps to identify the set of disjunctive arcs. Based on the task assignment, $\Upsilon_1 = \{(3,4)\}$, $\Upsilon_2 = \{(1,2)\}$, and $\Upsilon_3 = \{(5,6)\}$. Therefore, $\Upsilon = \{(1,2), (3,4), (5,6)\}$. Moreover, since $\Lambda = \{(2,3), (3,5), (4,6)\}$, $D = \{(1,2), (2,3), (3,4), (3,5), (4,6), (5,6)\}$.



Task	1	2	3	4	5	6
p_{ia}	9	5	5	6	6	8
p_{ib}	11	5	5	8	6	7
l_i	1	2	3	4	4	5

Figure 5.4: A simple example for illustration

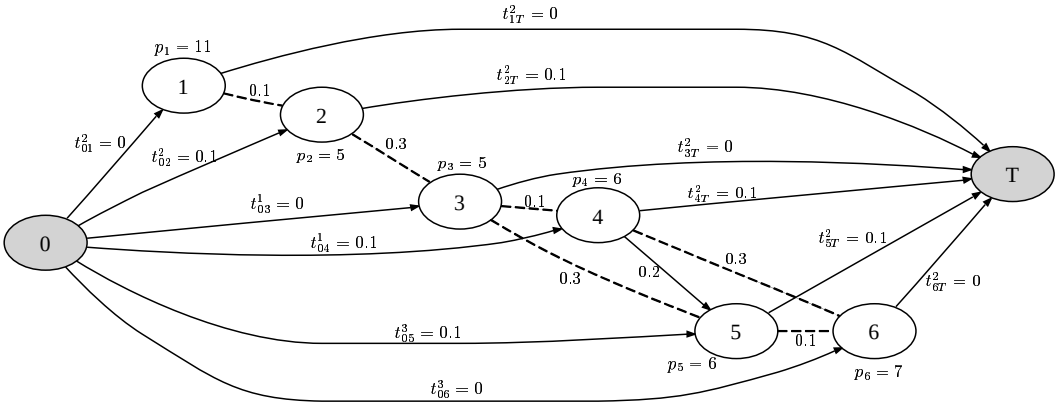


Figure 5.5: Disjunctive graph for the scheduling subproblem for example in Figure 5.4

After constructing the disjunctive graph $\mathcal{G}$, a scheduling subproblem is transformed to a problem of setting the direction of disjunctive arcs in the graph such that the graph becomes directed acyclic and the longest path from Node 0 to Node T is minimized. A heuristic to achieve the goal will be introduced hereafter, whose basic principle is: when fixing the direction of a particular disjunctive arc, choose the direction which might lead to a shorter critical path in the final Directed Acyclic Graph (DAG). In the literature, the basic idea of this heuristic is also applied to solve the berth planning problem in Lim (1998). Before describing the heuristic, some new terms should be defined.

- L^j_{in}, $j \in \Omega$, the longest incoming path of node j;
- L^j_{out}, $j \in \Omega$, the longest outgoing path from node j;
- $\beta_{ij} = L^i_{in} + w_i + w_{ij} + w_j + L^j_{out}$, $(i,j) \in D$;
- $\beta_{ji} = L^j_{in} + w_j + w_{ji} + w_i + L^i_{out}$, $(i,j) \in D$;
- $\alpha_{ij} = \max\{\beta_{ij}, \beta_{ji}\}$, the potential of the disjunctive arc (i,j);
- $\tau(\mathcal{G})$, the longest path from node 0 to node T in a graph $\mathcal{G}$;
- $\overline{D}$, a set of directed arcs;

- Let $\langle i_k, j_k \rangle$ be the k^{th} element of $\overline{D}$, $k \in \{1, \cdots, |\overline{D}|\}$. Define $\overline{D}_k = (\overline{D} \backslash \{\langle i_k, j_k \rangle\}) \cup \{\langle j_k, i_k \rangle\}$. In other words, $\overline{D}_k$ is the same as $\overline{D}$ except that its k^{th} element is $\langle j_k, i_k \rangle$ rather than $\langle i_k, j_k \rangle$.

Scheduling Subproblem Heuristic (SSH):

Step 1: $\forall\, i \in \Omega$, calculate L_{in}^i and L_{out}^i. Let $\overline{D} = \emptyset$.

Step 2: $\forall\, (i,j) \in D$, compute α_{ij}.

Step 3: Find the disjunctive arc (i,j) with the highest potential α_{ij}. If there is a tie in the highest potential, pick the arc with the largest $|\beta_{ij} - \beta_{ji}|$. Let $D \leftarrow D \backslash \{(i,j)\}$.

Step 4: If $(\beta_{ij} < \beta_{ji})$

Then set arc to go from i to j, add $\langle i,j \rangle$ into $\overline{D}$

Else if $(\beta_{ij} > \beta_{ji})$

Then set arc to go from j to i, add $\langle j,i \rangle$ into $\overline{D}$

Else if $(i < j)$

Then set arc to go from i to j, add $\langle i,j \rangle$ into $\overline{D}$

Else set arc to go from j to i, add $\langle j,i \rangle$ into $\overline{D}$

Step 5: Update the affected longest incoming and outgoing paths of nodes.

Step 6: Update the potential of affected disjunctive arcs.

Step 7: If $D \neq \emptyset$, goto Step 3.

Step 8: Calculate $\tau(\mathcal{G}_0) = \max_{i \in \Omega}(L_{in}^i + w_i + L_{out}^i)$ for graph $\mathcal{G}_0 = (\overline{\Omega}, C, \overline{D}, W_{\mathcal{V}}, W_{\mathcal{E}})$.

Step 9: Let $K = \{0\}$.

For $k = 1, \ldots, |\overline{D}|$,

If graph $\mathcal{G}_k = (\overline{\Omega}, C, \overline{D}_k, W_{\mathcal{V}}, W_{\mathcal{E}})$ is a DAG,

Compute the longest path from Node 0 to Node T in graph $\mathcal{G}_k$, i.e., $\tau(\mathcal{G}_k)$. $K \leftarrow K \cup k$.

Step 10: Return $\min_{k\in K}\tau(\mathcal{G}_k)$ and the corresponding graph with minimum longest path, $\mathcal{G}_{k^*}$, $k^* = \text{argmin}_{k\in K}\tau(\mathcal{G}_k)$.

The algorithm of SSH consists of two parts. Firstly, Step 1 to Step 7 try to fix the direction of disjunctive arcs one by one. In each iteration, a particular disjunctive arc is chosen by the criteria of maximum potential α_{ij} (in the event of a tie, $|\beta_{ij} - \beta_{ji}|$). Once the arc is selected, the direction of the arc is determined in the way that the longest path through the arc is as small as possible (Lim, 1998). Secondly, the function of Step 8 to Step 9 is to conduct a local search based on a feasible solution, graph $\mathcal{G}_0$, generated by the procedures beforehand.

Note that topological sorting can be used to determine the longest incoming and outgoing paths of nodes and additionally, to verify whether a graph is a DAG or not. Since a topological sorting has running time linear in the number of nodes plus the number of arcs of a graph $\mathcal{G} = (\overline{\Omega}, C, D, W_{\mathcal{V}}, W_{\mathcal{E}})$ (i.e., $\mathcal{O}(|\overline{\Omega}|+|C|+|D|)$), the time complexity of SSH is $\mathcal{O}(|D|\cdot(|\overline{\Omega}|+|C|+|D|))$.

Lemma 5.3.1. *The output of SSH is a DAG.*

Proof. It is easy to verify that the input for SSH, $\mathcal{G} = (\overline{\Omega}, C, D, W_{\mathcal{V}}, W_{\mathcal{E}})$ does not contain a cycle. Given $\mathcal{G}$, in Lim (1998), it has been proved that the procedures from Step 1 to Step 7 in SSH will create a DAG. That is, $\mathcal{G}_0$ is a DAG. Moreover, since in Step 9 of SSH, the graph $\mathcal{G}_k$, $k \in \{1, \cdots, |\overline{D}|\}$ which is not a DAG, has been skipped, the set of graph $\{\mathcal{G}_k\}_{k\in K}$ is a set of directed acyclic graphs. Therefore, the output of SSH, $\mathcal{G}_{k^*}$, $k^* = \text{argmin}_{k\in K}\tau(\mathcal{G}_k)$ is a DAG. □

Since SSH will generate a DAG, it guarantees that its corresponding scheduling plan should be a feasible solution to the scheduling subproblem of the QCSP at indented berth. Figure 5.6 and 5.7 show the procedures of SSH for the disjunctive graph in Figure 5.5 (Omit nodes 0 and T and the corresponding arcs connected to them). The output of SSH for the disjunctive graph in Figure 5.5 is 19.4 and its corresponding graph is $\mathcal{G}_0$. Providing $\mathcal{G}_{k^*}$ from SSH, it can be projected back to the domain of original scheduling subproblem. For example, Figure 5.8 gives the interpretation of $\mathcal{G}_0$ in terms of quay crane moving trajectories.

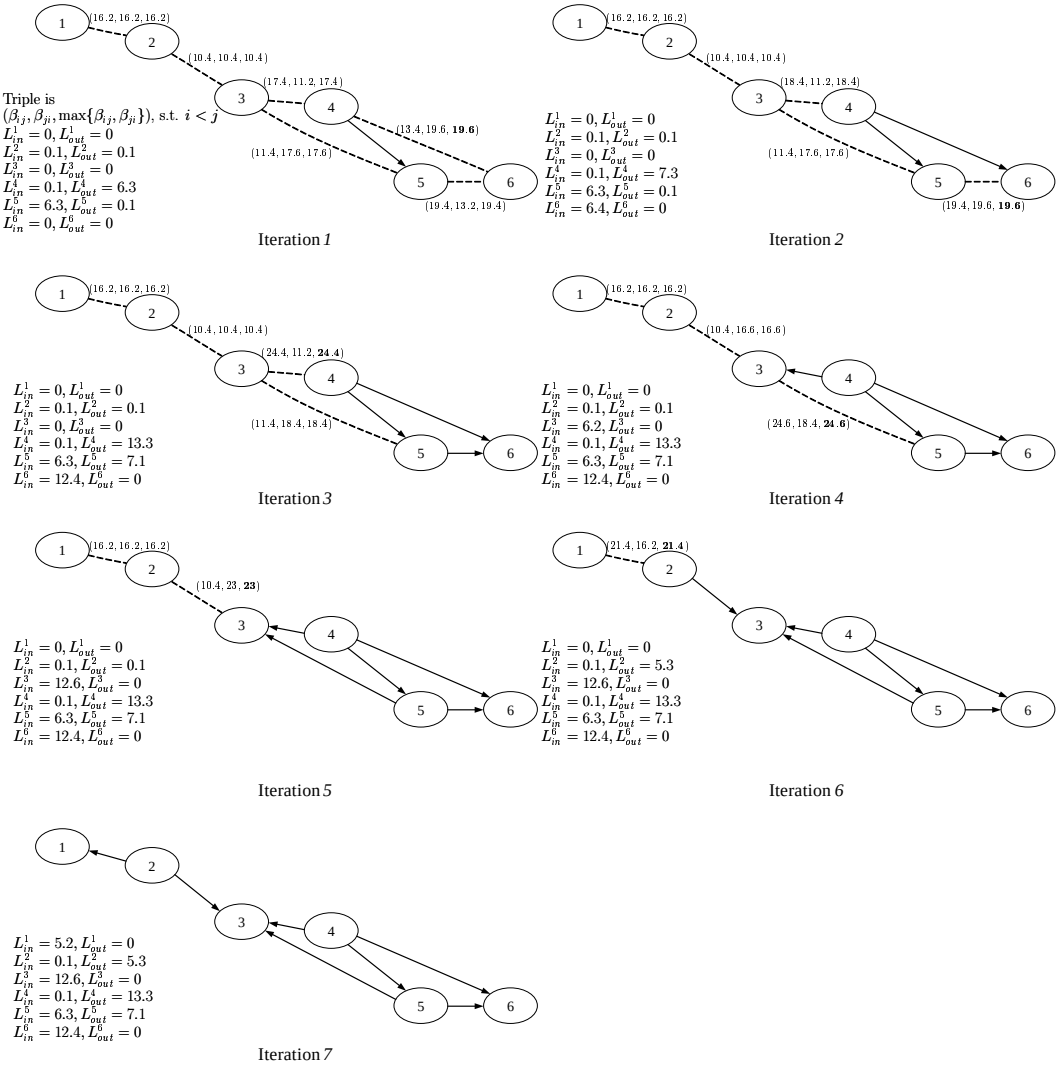


Figure 5.6: Iterations of SSH from Step 1 to Step 7 for the disjunctive graph in Figure 5.5

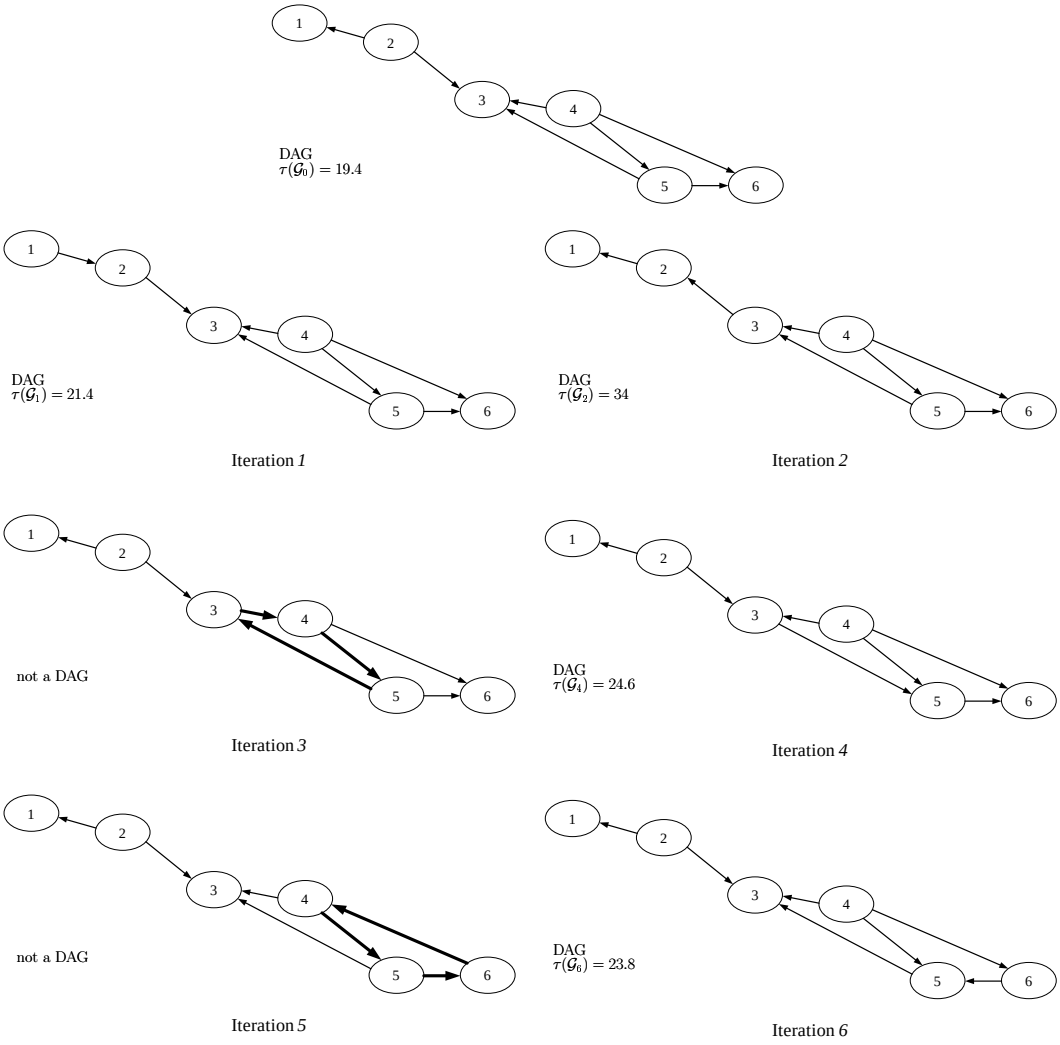


Figure 5.7: Iterations of SSH in Step 9 for the disjunctive graph in Figure 5.5

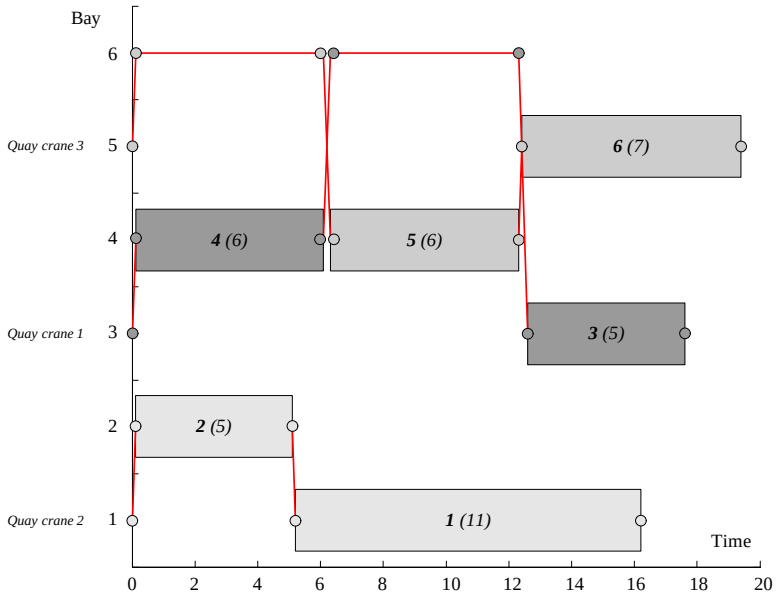


Figure 5.8: Interpretation of the output from SSH for the scheduling subproblem in Figure 5.4

5.3.2 Heuristics for assigning subproblem

The assigning subproblem for the QCSP at indented berth can be treated as a special case for unrelated parallel machines scheduling problem, $R_m \mid \mid c_{\max}$. In an unrelated parallel machines scheduling problem, there are n tasks to be scheduled without preemption on m unrelated parallel machines. Each task is to be assigned to a machine and, at any time, each machine can process at most one task. Task j $(j = 1, \ldots, n)$ requires a positive handling time p_{jk} if it is assigned to Machine k $(k = 1, \ldots, m)$. The objective of an unrelated parallel machines scheduling problem is to assign the tasks to machines so that the maximum completion time $c_{\max}$ is minimized.

Mathematically, the unrelated parallel machines scheduling problem can be formulated as:

$$[\mathcal{P}] \min \quad c_{\max} \tag{5.23}$$

s.t.

$$\sum_{j=1}^{n} p_{jk} u_{kj} \leq c_{\max}, \qquad k = 1, \ldots, m \tag{5.24}$$

$$\sum_{k=1}^{m} u_{kj} = 1, \qquad j = 1, \ldots, n \tag{5.25}$$

$$u_{kj} \in \{0, 1\}, \qquad k = 1, \ldots, m; j = 1, \ldots, n \tag{5.26}$$

$$c_{\max} \in \Re^{+} \tag{5.27}$$

where u_{kj} is a binary assignment variable and equals to 1 if and only if Task j is assigned to Machine k. Unfortunately, the unrelated parallel machines scheduling problem is $\mathcal{NP}$-hard (Hariri and Potts, 1991). Therefore, most researchers resorted to heuristic methods which provide approximation solutions. In this research, two kinds of heuristics will be applied to solve the assigning subproblem.

Assigning Subproblem Heuristic 1 (ASH1):

The heuristic is proposed by Hariri and Potts (1991), which integrates the linear programming relaxation and earliest completion time heuristic. One observation for problem $\mathcal{P}$ is that if constraints (5.26) are replaced by $u_{kj} \geq 0$ $(k = 1, \ldots, m; j = 1, \ldots, n)$, for $n \geq m - 1$, at least $n - m + 1$ assignment variables take the value 1 in a basic optimal solution. Therefore, linear programming relaxation of problem $\mathcal{P}$ will produce a partial schedule in which at least $\max\{n-m+1, 0\}$ tasks are assigned to machines. Let u^*_{kj} $(k = 1, \ldots, m; j = 1, \ldots, n)$ be the optimal solution for $\mathcal{P}$ by linear programming relaxation. Denote $\Omega_k = \{j \mid u^*_{kj} = 1, j = 1, \ldots, n\}$ which stands for the task set that assigned to machine k $(k = 1, \ldots, m)$. Hence, for the set of unscheduled tasks U, $U = \{1, \ldots, n\} \backslash \bigcup_{k=1}^{m} \Omega_k$. Given the partial solution from linear programming relaxation, earliest completion time heuristic can be used to assign the remaining tasks.

Step 1: Solve the linear programming relaxation of problem $\mathcal{P}$ by Simplex method. Based on the optimal solution u^*_{kj} $(k = 1, \ldots, m; j = 1, \ldots, n)$, initialize $\{\Omega_k\}$, $k = 1, \ldots, m$.

Step 2: Determine the set U, $U = \{1, \ldots, n\} \backslash \bigcup_{k=1}^{m} \Omega_k$.

Step 3: $\forall\, k = 1, \ldots, m$, if $\Omega_k \neq \emptyset$, compute $t_k = \sum_{j \in \Omega_k} p_{jk}$; otherwise, let $t_k = 0$.

Step 4: Determine $j^* = \text{argmin}_{j \in U}\{\min_{k=1}^{m}(t_k + p_{jk})\}$ and $k^* = \text{argmin}_{k=1}^{m}(t_k + p_{kj^*})$.

Step 5: Assign Task j^* to Machine k^*. Update U, Ω_{k^*} and t_{k^*}, by $U \leftarrow U \backslash \{j^*\}$, $\Omega_{k^*} \leftarrow \Omega_{k^*} \cup \{j^*\}$, $t_{k^*} \leftarrow t_{k^*} + p_{k^*j^*}$.

Step 6: If $U \neq \emptyset$, goto Step 4.

Step 7: Return $\{\Omega_k\}$, $k = 1, \ldots, m$.

For the example shown in Figure 5.4, ASH1 will create the assignment plan that $\Omega_1 = \{1, 4\}$, $\Omega_2 = \{2, 5\}$, and $\Omega_3 = \{3, 6\}$.

Assigning Subproblem Heuristic 2 (ASH2):

The second heuristic is called ASH2. Compared with ASH1, in this heuristic, all the tasks are assigned to machines purely based on the rule of earliest completion time first. For the example shown in Figure 5.4, ASH2 will generate the assignment plan that $\Omega_1 = \{2, 4\}$, $\Omega_2 = \{3, 6\}$, and $\Omega_3 = \{1, 5\}$.

5.3.3 Tabu search to refine the assigning solution

Although combining ASH1 and SSH or ASH2 and SSH can obtain a feasible solution to the QCSP at indented berth, since the heuristics for assigning subproblem fail to take the vital constraints such as Non-crossing and safety distance into account, sometimes the generated feasible solution may be far from satisfactory. Therefore, Tabu search will be proposed to refine the assigning solution created by ASH1 or ASH2. Figure 5.9 is the flowchart for the proposed Tabu search. Compared with traditional Tabu search, the proposed Tabu search in this research takes advantage of hashing technique to keep track of all the visited solutions so as to prevent cycling. In the literature, the approach to embed hashing technique within the Tabu search framework has been successfully implemented in solving machine scheduling problems such as Srivastava (1998).

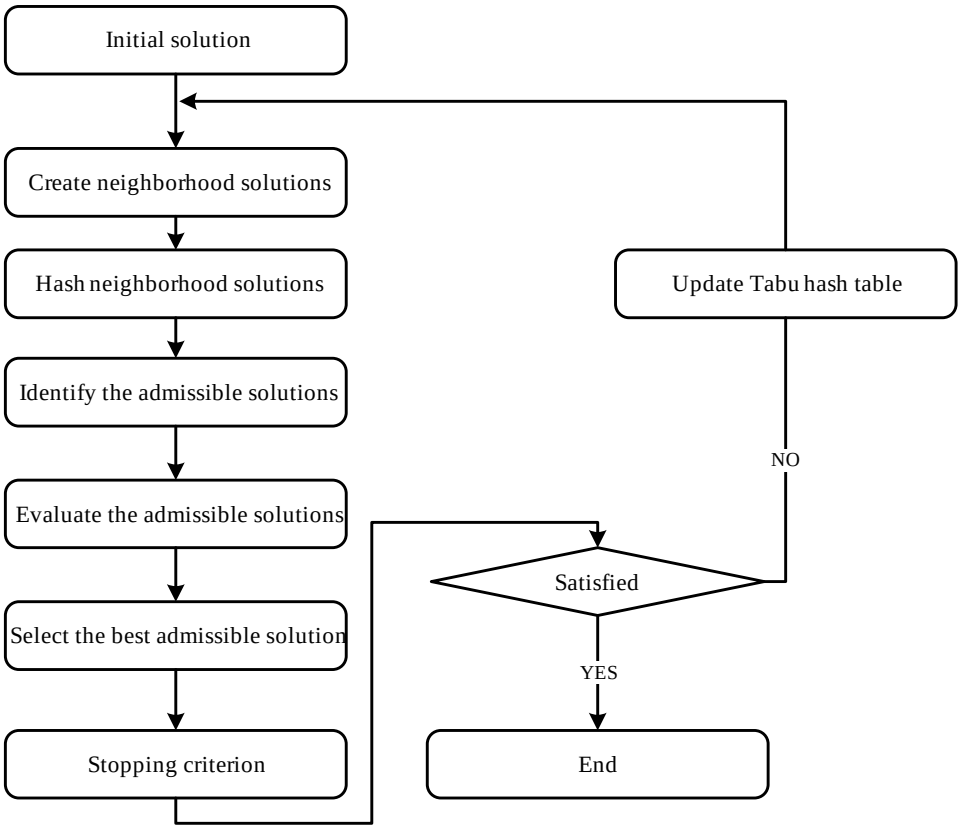


Figure 5.9: Flowchart of proposed Tabu search

Hashing technique is an important method widely used in information retrieval systems. Its basic idea is to map relatively complicated records into integers through a hashing function and use the integers to represent the records for further information manipulation, in light of the fact that keeping and searching a list of integers is comparatively easier. Given a record $\mathbf{s} = (s_1, s_2, \ldots, s_m)$, a hashing function h is served to map the record $\mathbf{s}$ into an integer z. There are several types of hash function. For example, a hashing function can be:

$$z = h(\mathbf{s}) = \sum_{i=1}^{m} z_i \cdot s_i \tag{5.28}$$

where $(z_1, z_2, \ldots, z_m)$ is a pre-computed vector of pseudo-random integers in the range $[1, \rho]$. Woodruff and Zemel (1993) has demonstrated that as long as the value of ρ is relatively large, the probability of collision (i.e., when two different records are encountered with the same hashing value) would be pretty low. In this research, ρ is set as $2^{16} = 65536$.

One intuitive way to code the assigning plan for the QCSP at indented berth is to use a vector $\mathbf{s} = (s_1, s_2, \ldots, s_{|\Omega|})$ with dimension equaling to the size of tasks, whose element s_i $i \in \Omega$ is an integer and bounded in the range $[1, |Q|]$. If $s_i = k$, $i \in \Omega, k \in Q$, it means that Task i is assigned to Crane k.

Initial solution

Let $\mathbf{s}_{\mathrm{I}} = (s_1^{\mathrm{I}}, s_2^{\mathrm{I}}, \ldots, s_{|\Omega|}^{\mathrm{I}})$ be the assigning plan generated by ASH1 and $\mathbf{s}_{\mathrm{II}} = (s_1^{\mathrm{II}}, s_2^{\mathrm{II}}, \ldots, s_{|\Omega|}^{\mathrm{II}})$ be the output of ASH2. After inputting $\mathbf{s}_{\mathrm{I}}$ and $\mathbf{s}_{\mathrm{II}}$ into SSH, let $\tau(\mathcal{G}(\mathbf{s}_{\mathrm{I}}))$ and $\tau(\mathcal{G}(\mathbf{s}_{\mathrm{II}}))$ be the corresponding makespans, respectively. If $\tau(\mathcal{G}(\mathbf{s}_{\mathrm{I}})) \leq \tau(\mathcal{G}(\mathbf{s}_{\mathrm{II}}))$, the initial solution $\mathbf{s}_0 = \mathbf{s}_{\mathrm{I}}$; otherwise, $\mathbf{s}_0 = \mathbf{s}_{\mathrm{II}}$.

Create neighborhood solutions

Given an assigning plan $\mathbf{s} = (s_1, s_2, \ldots, s_{|\Omega|})$, two kinds of operation can be conducted on $\mathbf{s}$ to create its neighborhood solutions, $N(\mathbf{s})$. The first operation is called pairwise interchange. Before pairwise interchange, two random integers i, j $(i, j \in \Omega, i < j, s_i \neq s_j)$ will be generated. Then a neighborhood solution of $\mathbf{s} = (s_1, \ldots, s_i, \ldots, s_j, \ldots, s_{|\Omega|})$, $\mathbf{s}'$, can be constructed as $(s_1, \ldots, s_j, \ldots, s_i, \ldots, s_{|\Omega|})$. The second operation is called flipping which changes a randomly selected element s_i $(i \in \Omega)$ into another integer in the set $Q \backslash \{s_i\}$.

Hash neighborhood solutions and identify the admissible solutions

$\forall\, \mathbf{s}' = (s_1', s_2', \ldots, s_{|\Omega|}')$ in $N(\mathbf{s})$, its corresponding hash value $z' = h(\mathbf{s}') = \sum_{i \in \Omega} z_i \cdot s_i'$. Denote H as the Tabu hash table for all the visited solutions (initially, $H = \{h(\mathbf{s}_0)\}$). Then if $z' \in H$, statistically speaking, it is very safe to assert that the $\mathbf{s}'$ is a previously visited solution given that ρ is relatively large. Therefore, at the current solution $\mathbf{s}$, a neighborhood solution $\mathbf{s}'$ is regarded as an admissible solution if and only if $\mathbf{s}' \in N(\mathbf{s})$ and $h(\mathbf{s}') \notin H$. Define a set for all the admissible moves when the current solution is $\mathbf{s}$, $A(\mathbf{s}) = \{\mathbf{s}' \,|\, \mathbf{s}' \in N(\mathbf{s}) \wedge h(\mathbf{s}') \notin H\}$.

Evaluate the admissible solutions and select the best one

Suppose the current solution is $\mathbf{s}$, for all $\mathbf{s}' \in A(\mathbf{s})$, SSH can be used to evaluate the goodness of the task assigning plan $\mathbf{s}'$. Let $\tau(\mathcal{G}(\mathbf{s}'))$ be the output of SSH when $\mathbf{s}'$ is evaluated. Then the best admissible solution is $\mathbf{s}'^* = \mathrm{argmin}_{\mathbf{s}' \in A(\mathbf{s})} \tau(\mathcal{G}(\mathbf{s}'))$.

Stopping criterion

The Tabu search algorithm will be terminated in the case that one of the following conditions is satisfied:

- The maximum iteration number L_1 is achieved.
- The objective value has never changed from the previous L_2 consecutive iterations.

Update Tabu hash table

In one iteration of the Tabu search algorithm, if both stopping criteria can not be met, the Tabu hash table H will be updated. Simply let $H \leftarrow H \cup \{z' \mid z' = h(\mathbf{s}'), \mathbf{s}' \in A(\mathbf{s})\}$.

Hereafter, the above Tabu search algorithm will be called Tabu1. Actually, there is another strategy to code the information of assigning plan in Tabu search for the QCSP at indented berth. The principle of this version of Tabu search (named Tabu2) is to aggregate all the tasks located in the same ship bay as a batch and assign the batch to a single quay crane. This kind of scheduling scheme is referred to be as bay-based QCSP in Bierwirth and Meisel (2009). Let $\mathbf{b} = (b_1, b_2, \ldots, b_l)$ be a binary string with length l which is the total number of ship bays. The element $b_i \in \{0, 1\}$ $(i = 1, \ldots, l)$, is equal to 1 if and only if the batch of tasks in Bay i is assigned to a Crane k $(k \in Q_a)$. By default, if Bay i does not contain any task, b_i is set to 0.

Initial solution for Tabu2

Without loss of generality, suppose the assigning plan of ASH1 is superior to that of ASH2 (i.e., $\tau(\mathcal{G}(\mathbf{s}_{\mathrm{I}})) \leq \tau(\mathcal{G}(\mathbf{s}_{\mathrm{II}}))$). Here is the rule to construct the initial solution for Tabu2, $\mathbf{b}_0 = (b_1^0, b_2^0, \ldots, b_l^0)$: for Bay i $(i = 1, \ldots, l)$, if there are more than one task located inside, the value of b_i^0 is equal to the value randomly selected from $\{0, 1\}$; if there is only one task (e.g., task j, $j \in \Omega$) located in Bay i, $b_i^0 = 1$ in the case that $s_j^{\mathrm{I}} \in Q_a$ and $b_i^0 = 0$ in the case that $s_j^{\mathrm{I}} \in Q_b$. Taking the case in Figure 5.4 as an example, the output of ASH1 is $\Omega_1 = \{1, 4\}$, $\Omega_2 = \{2, 5\}$, and $\Omega_3 = \{3, 6\}$, in other words, $\mathbf{s}_{\mathrm{I}} = (1, 2, 3, 1, 2, 3)$. Note that both Task 4 and Task 5 are located in Bay 4, therefore, the value of b_4^0 can be ether 0 or 1. Besides, since there is no task in Bay 6, b_6^0 should equal to 0. Hence, $\mathbf{b}_0$ can be $(1, 0, 0, 1, 0, 0)$ or $(1, 0, 0, 0, 0, 0)$.

Create neighborhood solutions for Tabu2

Given a ship bay assigning plan $\mathbf{b} = (b_1, b_2, \ldots, b_l)$, the methods to create its neighborhood solution, $N(\mathbf{b})$ are similar to the two operations introduced before. Note that for flipping operation, if element b_i $(i = 1, \ldots, l)$ is chosen to manipulate, then b_i will be changed to the integer in the set $\{0, 1\} \backslash \{b_i\}$.

Hash neighborhood solutions and identify the admissible solutions for Tabu2

$\forall\, \mathbf{b}' = (b'_1, b'_2, \ldots, b'_l)$ in $N(\mathbf{b})$, its hash value $z' = h(\mathbf{b}') = \sum_{i=1}^{l} z_i \cdot b'_i$. Define the admissible moves when the current solution is $\mathbf{b}$, $A(\mathbf{b}) = \{\mathbf{b}' \,|\, \mathbf{b}' \in N(\mathbf{b}) \wedge h(\mathbf{b}') \notin \overline{H}\}$. Here $\overline{H}$ is the Tabu hash table for Tabu2 and it is initialized as $\overline{H} = \{h(\mathbf{b}_0)\}$ immediately after the construction of the initial solution for Tabu2.

Evaluate the admissible solutions and select the best one for Tabu2

To evaluate the goodness of an admissible solution, $\mathbf{b}$, Tabu2 needs more attempts compared with Tabu1. Before using SSH, the assignment information for individual task should be obtained based on $\mathbf{b} = (b_1, b_2, \ldots, b_l)$. Let $\overline{\mathbf{p}} = (\overline{p}_1, \overline{p}_2, \ldots, \overline{p}_l)$, where $\overline{p}_i$ $(i = 1, \ldots, l)$ is the aggregated handling time for Bay i under the ship bay assignment plan $\mathbf{b}$. Specifically, define the task subset $\Omega(i) = \{j \,|\, l_j = i, j \in \Omega, 1 \leq i \leq l\}$, which is the set for all the tasks located in Bay i. Then $\overline{p}_i = \sum_{j \in \Omega(i)} [b_i \cdot p_{jk \,|\, k \in Q_a} + (1 - b_i) \cdot p_{jk \,|\, k \in Q_b}]$. In the case that $\Omega(i) = \emptyset$ $(1 \leq i \leq l)$, $\overline{p}_i = 0$. Based on $\overline{\mathbf{p}}$, two vectors, $\overline{\mathbf{p}}_\mathrm{a}$ and $\overline{\mathbf{p}}_\mathrm{b}$ can be calculated: $\overline{\mathbf{p}}_\mathrm{a} = (b_1 \cdot \overline{p}_1, b_2 \cdot \overline{p}_2, \ldots, b_l \cdot \overline{p}_l)$, $\overline{\mathbf{p}}_\mathrm{b} = ((1-b_1) \cdot \overline{p}_1, (1-b_2) \cdot \overline{p}_2, \ldots, (1-b_l) \cdot \overline{p}_l)$. The purpose to define $\overline{\mathbf{p}}_\mathrm{a}$ and $\overline{\mathbf{p}}_\mathrm{b}$ is to store the workload list that purely related to the quay cranes in Q_a or Q_b. For instance, let $\mathbf{b} = (1, 0, 0, 1, 0, 0)$ for the example shown in Figure 5.4. Then $\overline{\mathbf{p}} = (9, 5, 5, 12, 7, 0)$, $\overline{\mathbf{p}}_\mathrm{a} = (9, 0, 0, 12, 0, 0)$ and $\overline{\mathbf{p}}_\mathrm{b} = (0, 5, 5, 0, 7, 0)$.

It is worth to mention that given a $\mathbf{b}$, without consideration of safety distance requirement, the bay-based QCSP at indented berth can be treated as two traditional bay-based QCSP (see Figure 5.10). Therefore, the heuristics proposed in the literature for solving bay-based QCSP can be implemented to both traditional bay-based quay crane scheduling problems and finally generate the assignment plan for individual task.

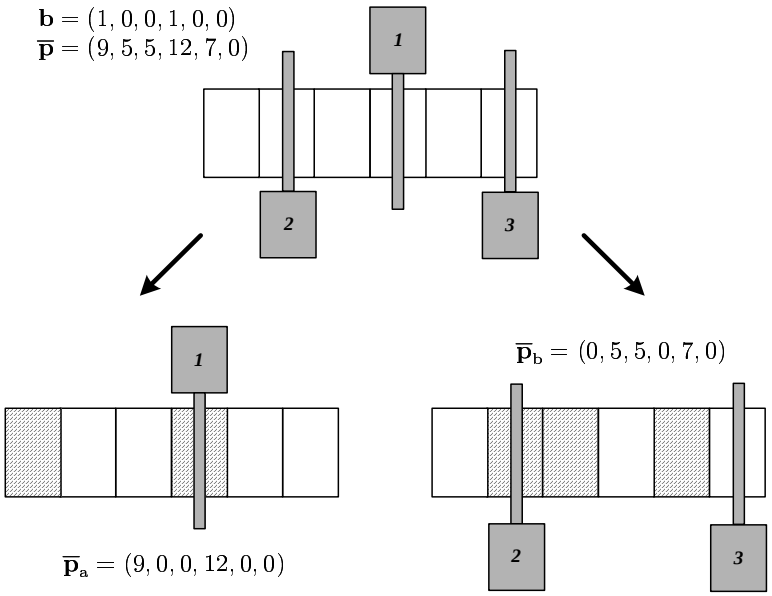


Figure 5.10: Partition the problem for the example in Figure 5.4

The approximation algorithm proposed in Lim et al. (2004b) is adopted to solve both traditional bay-based QCSP. Given the number of quay cranes (m) and the workload list for all the ship bays (e.g., $\hat{\mathbf{p}} = (\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n)$), the approximation algorithm BP introduced in Chapter 4 will try to seek the best partition pattern for the ship bays and assign each quay crane to the corresponding partition orderly through the DP recursive functions of 4.22 and 4.23. For example, in Figure 5.10, if the number of quay crane $m = 2$ and the workload list $\overline{\mathbf{p}}_b = (0, 5, 5, 0, 7, 0)$ are fed into the DP recursive functions, BP will generate two consecutive sections, $(0, 5, 5)$ and $(0, 7, 0)$ and assign them to Cranes 2 and 3, respectively. Obviously, not only does BP distribute the workload as evenly as possible, but also its generated bay-based work assignment naturally eliminates the issue of crossing among quay cranes.

By sequentially inputting ($m = |Q_a|$, $\hat{\mathbf{p}} = \overline{\mathbf{p}}_a$) and ($m = |Q_b|$, $\hat{\mathbf{p}} = \overline{\mathbf{p}}_b$) into BP, $\mathbf{b}$ can be translated to a more detailed individual task assignment $\mathbf{s} = (s_1, s_2, \ldots, s_{|\Omega|})$, where $s_i \in Q$ ($i \in \Omega$) means that Task i is assigned to Crane s_i. For the case shown in Figure 5.10, if $\mathbf{b} = (1, 0, 0, 1, 0, 0)$, after using BP twice, the assignment information for individual task $\mathbf{s} = (1, 2, 2, 1, 1, 3)$ can be obtained. For ease of description, denote this transformation briefly as $\mathbf{s} = \mathrm{BP}(\mathbf{b})$. After the translation, the goodness of $\mathbf{b}$ can be evaluated by SSH. Let $\tau(\mathcal{G}(\mathrm{BP}(\mathbf{b})))$

be the output of SSH when $\mathbf{b}$ is evaluated. Therefore, the best admissible solution for Tabu2 is $\mathbf{b}'^* = \text{argmin}_{\mathbf{b}' \in A(\mathbf{b})} \tau(\mathcal{G}(\text{BP}(\mathbf{b}')))$.

Stopping criterion for Tabu2

The stopping criterion for Tabu2 is exactly the same as Tabu1.

Update Tabu hash table for Tabu2

The rule to update the Tabu hash table is: $\overline{H} \leftarrow \overline{H} \cup \{z' \mid z' = h(\mathbf{b}'), \mathbf{b}' \in A(\mathbf{b})\}$.

5.4 Numerical Experiments

To examine the performance, the proposed heuristic framework is evaluated by a suite of randomly generated problems, which contains 7 instance sets of different problem size with 30 instances each. Table 5.1 summarizes the details for the 7 instance sets. For a Task i ($i \in \Omega$), the handling time p_{ik}, $\forall\, k \in Q_a$ follows a discrete uniform distribution in the range $[30, 100]$; while p_{ik}, $\forall\, k \in Q_b$ is derived by adding p_{ik}, $\forall\, k \in Q_a$ a random integer variable generated from a discrete uniform distribution in the range $[-15, 15]$. The length of a vessel measured by the number of bays is equal to the size of the tasks, $n = |\Omega|$. To construct an instance, first of all, the n tasks are randomly distributed to the n bays one by one. Next for each bay, if the number of tasks that assigned to the bay is greater than 2, then each task in that particular bay is tagged to be loading or discharging task randomly with equal possibility. The information obtained from this part can be used to construct the precedence constrained pair set, Φ. Finally, let the safety distance δ equal to one ship bay and $\hat{t} = 0.1$.

In this research, two criteria are used to assess the effectiveness of the proposed heuristic framework: computation time and solution quality. Because it is challenging to find the optimal solutions in reasonable time for each instance, a compromise strategy is to compare the approximate solutions that the proposed heuristic framework generates to lower bounds. Three kinds of lower bound from unrelated machine scheduling problem are calculated for each instance:

- $LB_1 = \frac{\sum_{i \in \Omega} \min_{k \in Q} p_{ik}}{|Q|}$. LB_1 distributes the sum of the minimal handling times among the quay cranes.

- $LB_2 = \max_{i\in\Omega}\{\min_{k\in Q} p_{ik}\}$.

- The third lower bound LB_3, is found by solving the surrogate dual problem of problem $\mathcal{P}$. Let $\lambda = (\lambda_1, \ldots, \lambda_m)$ be a vector of non-negative multipliers, the surrogate relaxation problem $\mathcal{S}_\lambda$ is obtained by dualizing constraints (5.24) in $\mathcal{P}$:

$$[\mathcal{S}_\lambda] \min \quad \frac{\sum_{k=1}^{m}\sum_{j=1}^{n} \lambda_k \cdot p_{jk} \cdot u_{kj}}{\sum_{k=1}^{m} \lambda_k} \tag{5.29}$$

s.t.

$$\sum_{k=1}^{m} u_{kj} = 1, \qquad j = 1, \ldots, n \tag{5.30}$$

$$u_{kj} \in \{0, 1\}, \qquad k = 1, \ldots, m; j = 1, \ldots, n \tag{5.31}$$

Note that problem $\mathcal{S}_\lambda$ is solvable in $\mathcal{O}(mn)$ (Ghirardi and Potts, 2005). Let $\mathcal{S}(\lambda)$ be the optimal solution of $\mathcal{S}_\lambda$, which is obviously a lower bound for problem $\mathcal{P}$. Then the surrogate dual problem $\mathcal{D}$ for problem $\mathcal{P}$ is:

$$[\mathcal{D}] \max \quad \mathcal{S}(\lambda) \tag{5.32}$$

$$\text{s.t.} \quad \lambda \geq 0 \tag{5.33}$$

Problem $\mathcal{D}$ can be solved by ascent direction algorithm proposed in Van de Velde (1993). Let LB_3 be the optimal solution of problem $\mathcal{D}$.

After calculating the above three lower bounds, the lower bound for each instance LB is equal to $\max\{LB_1, LB_2, LB_3\}$.

Table 5.1: Detail of the 7 instance sets

Instance set	A	B	C	D	E	F	G
Tasks: $\|\Omega\|$	10	15	20	25	30	35	40
Quay cranes: $\|Q_a\|$	1	1	2	2	2	3	3
Quay cranes: $\|Q_b\|$	2	2	3	3	3	4	4

The proposed heuristic framework is coded in Matlab and ran in a PC with 2.4 GHz CPU

and 2.99 GB of RAM. There are 4 solution methods in the framework to find the approximate solutions for the QCSP at indented berth: the combination of ASH1 and SSH (denoted as ASH1+SSH), the combination of ASH2 and SSH (denoted as ASH2+SSH), Tabu1, and Tabu2. The first two solution methods are deterministic heuristics while the latter two are stochastic ones. For Tabu1 and Tabu2, the maximum iteration number L_1 and the number to test the convergence L_2 are set to 200 and 10, respectively. Besides, for each instance, both Tabu search algorithms are executed 3 times and the information of the best solution found in the 3 runs for each algorithm is recorded for evaluation. For a particular instance, let μ be the makespan for the solution found by one of the four solution methods. Then the effectiveness of the corresponding solution method can be measured by:

$$\text{gap} = \frac{\mu - LB}{LB} \times 100\% \tag{5.34}$$

To select the suitable size of neighborhood solutions searched in each iteration (denoted as L_3), Tabu1 with different values of L_3 is executed to solve the 30 instances in Set A. Figure 5.11 depicts the performance of Tabu1 to solve instance Set A with different value of L_3 ranging from 20 to 160 with step size 20. In Figure 5.11, the horizontal axis measures the effectiveness of the algorithm while the vertical axis is the average CPU time consumed by the algorithm to solve the 30 instances. As can be observed, except $L_3 = 20$ and $L_3 = 60$ which lies on the pareto front, other options are dominated. By putting more weight on effectiveness, 20 can be a suitable value for the parameter L_3. For simplicity, in the rest of numerical experiments, the value of L_3 is fixed to 20.

Table 5.2 summarizes the average computational time, the means and standard deviations of gaps for each solution method under different instance set. From the computational results shown in Table 5.2, several facts can be observed: 1) On average, the solutions generated by the two deterministic heuristics (i.e., ASH1+SSH and ASH2+SSH) are not good enough since the means of gap are ranging from 36% to 100%. 2) The difference between the two deterministic heuristics in terms of solution quality is marginal. 3) The improvement of solution by Tabu search is significant especially for Tabu2. Compared with Tabu1, Tabu2 not only can find better

solutions but also achieves that in a faster way. To explain this phenomenon, it is worth noting that the exact size of searching space for Tabu1 is $(|Q|)^{|\Omega|}$ while for Tabu2 is less $2^{|\Omega|}$ (recall that the tasks in the same ship bay are assigned to the same quay crane). Since in practice $|Q| \geq 2$, the exact searching space for Tabu1 is remarkably larger than the one for Tabu2. Besides, the task assignment plan generated by Tabu2 has taken the Non-crossing Constraints among quay cranes into account which will help to reduce the number of disjunctive arcs in graph $\mathcal{G}$. In light of the above facts, Tabu2 should be essentially more efficient than Tabu1.

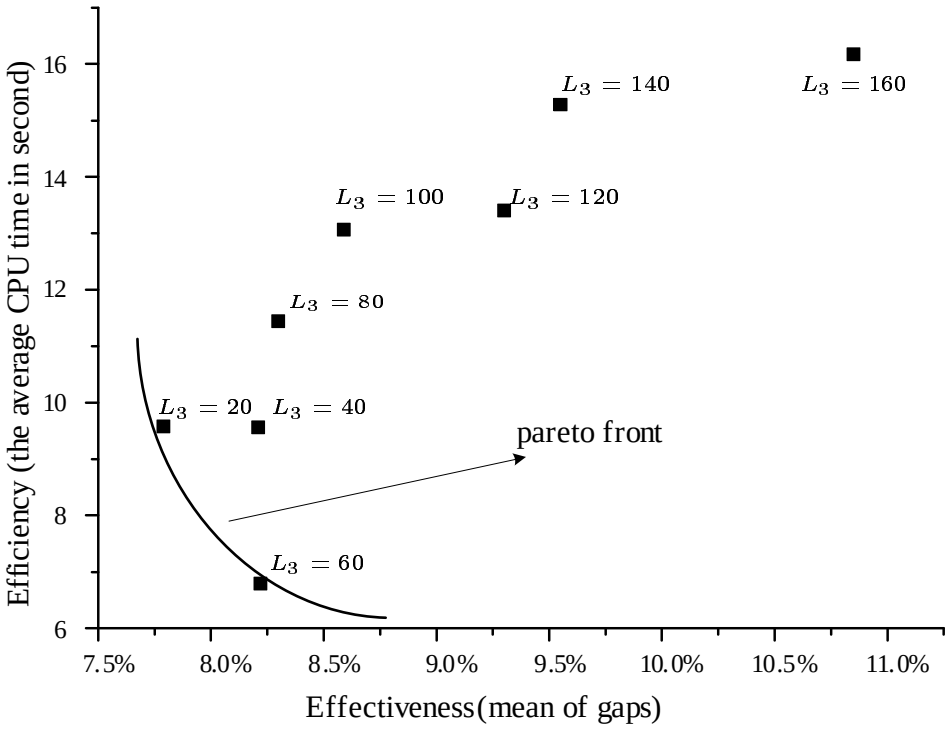


Figure 5.11: The selection of L_3

5.5 Summary

This paper discusses the QCSP at indented berth, an extension to the current QCSP in the field of container terminal operation. An MIP model by considering the unique features of the QCSP at indented berth is formulated. For solution, decomposition heuristic framework is developed and enhanced by Tabu search. To evaluate the performance of the proposed heuristic

Table 5.2: The performance of the proposed heuristic framework

Instance set		A	B	C	D	E	F	G
ASH1+SSH	CPU[a]	0.09	0.21	0.41	0.79	1.49	2.32	3.85
	Mean[b]	48.16	36.28	79.43	68.55	62.48	100.47	95.65
	STD[c]	18.32	18.96	18.86	18.44	23.47	19.03	16.66
ASH2+SSH	CPU	0.07	0.21	0.44	0.82	1.52	2.38	3.95
	Mean	41.43	37.6	79.42	68.55	62.48	100.47	95.65
	STD	16.94	18.39	18.86	18.44	23.47	19.03	16.66
Tabu1	CPU	9.58	40.75	82.38	227.99	453.99	915.13	1434.12
	Mean	7.79	4.22	16.52	11.7	12.38	28.44	29.27
	STD	3.35	1.87	6.77	4.45	4.96	7.24	8.17
Tabu2	CPU	4.38	20.46	22.61	59.79	101	100.41	192.61
	Mean	6.11	3.31	8.02	5.88	4.59	8.29	6.93
	STD	2.55	1.33	2.51	1.78	1.16	1.75	1.49

a: the average computational time in seconds
b: the mean of gaps (%)
c: the standard deviation of gaps (%)

framework, a comprehensive numerical test is carried out and its results show the good quality of the proposed heuristic framework.

Chapter 6

A Combinatorial Benders' Cuts Algorithm for the Quayside Operation Problem at Container Terminals

6.1 Background

Based on the literature review of the integrated model for the quayside operation problem presented in Chapter 2, it can be observed that the framework proposed in Liu et al. (2006) is advantageous since it is capable to veritably reveal the exact influence of the quay crane interference. That is, the QCSP is truly included into the integrated model of the quayside operation problem. Nevertheless, both Vessel-level and Berth-level models in Liu et al. (2006) are solved by the off-the-shelf optimization solver. In this Chapter, a more efficient and problem-oriented algorithm, called the Combinatorial Benders' Cuts (CBC) algorithm, will be proposed for the Berth-level model in Liu et al. (2006).

6.2 Mathematical Model

In this section, the mathematical model provided by Liu et al. (2006) is presented. Subsequently, some observations on the Liu, Wan, and Wang formulation (Liu et al., 2006) is introduced as well.

6.2.1 Notation

Let V denote the set of vessels to be processed in the planning horizon and Q be the set of all quay cranes available along the berth. Number the quay cranes one by one increasingly along the increasing direction of the berth. The followings are the input data for the model:

- r_i, the arrival time of Vessel i
- d_i, the scheduled departure time of Vessel i
- l_i, the (mid-point) position of Vessel i
- $\underline{q}^i$, the minimum number of quay cranes that can be deployed to Vessel i
- $\overline{q}^i$, the maximum number of quay cranes that can be deployed to Vessel i
- $\Theta(i) = \{q \,|\, \underline{q}^i \leq q \leq \overline{q}^i\}, \forall\, i \in V$, the set of legitimate quay crane numbers for Vessel i
- p_{iq}, the processing time when q quay cranes are assigned to process Vessel i
- τ, the total preparation time on the quayside for a docking vessel to depart and for a waiting vessel to dock

Additionally, in order to present the precedence relationships among the vessels, three kinds of vessel pair sets need to be defined.

- $\mathbb{R}$, the set of vessel pairs (i, j) such that Vessel j is the next vessel that docks along some berth position occupied by Vessel i
- $\mathbb{R}'$, the set of vessel pairs that include all the pairs in $\mathbb{R}$ and pairs (i, j) such that $\{(i, i_1), (i_1, i_2), \ldots, (i_e, j)\} \in \mathbb{R}$ for some Vessels $i_1, i_2, \ldots, i_e$

- $\mathbb{U}$, the set of all vessel pairs (i, j) such that both $(i, j) \notin \mathbb{R}'$ and $(j, i) \notin \mathbb{R}'$

6.2.2 The Liu, Wan, and Wang formulation

In Liu et al. (2006), the ultimate objective for the proposed problem is to minimize the maximum relative tardiness of all vessels in a planning horizon. The relative tardiness of a particular Vessel i is defined as:

$$L_i = \frac{\max(0, c_i - d_i)}{d_i - r_i}$$

where L_i is the relative tardiness of Vessel i and c_i is the processing completion time of Vessel i. The list below shows all the decision variables used by Liu et al. (2006):

- L_i, the relative tardiness of Vessel i
- L, the maximum relative tardiness of all vessels
- s_i, the service start time of Vessel i
- c_i, the service completion time of Vessel i
- x^s_{ki}, the position of quay crane k at the start time of Vessel i
- x^c_{ki}, the position of quay crane k at the completion time of Vessel i
- $y^{ss}_{ij} \in \{0, 1\}$, 1, if s_i is before s_j, $(i, j) \in \mathbb{U}$; 0, otherwise
- $y^{sc}_{ij} \in \{0, 1\}$, 1, if s_i is before c_j, $(i, j) \in \mathbb{U}$; 0, otherwise
- $y^{cs}_{ij} \in \{0, 1\}$, 1, if c_i is before s_j, $(i, j) \in \mathbb{U}$; 0, otherwise
- $y^{cc}_{ij} \in \{0, 1\}$, 1, if c_i is before c_j, $(i, j) \in \mathbb{U}$; 0, otherwise
- $z_{ki} \in \{0, 1\}$, 1, if Crane k is assigned to process Vessel i; 0, otherwise
- $\psi_{iq} \in \{0, 1\}$, 1, if q quay cranes are assigned to process Vessel i; 0, otherwise

The Liu, Wan, and Wang formulation, denoted by $[\mathcal{F}]$ hereafter, is shown as follows:

$$[\mathcal{F}] \quad \min \quad L \tag{6.1}$$

s.t.

$$\overline{q}^i \geq \sum_{k \in Q} z_{ki} \geq \underline{q}^i, \qquad \forall\, i \in V \tag{6.2}$$

$$\sum_{q \in \Theta(i)} q \cdot \psi_{iq} = \sum_{k \in Q} z_{ki}, \qquad \forall\, i \in V \tag{6.3}$$

$$s_j \geq s_i + M \cdot (y_{ij}^{ss} - 1), \qquad \forall\, (i,j) \in \mathbb{U} \tag{6.4}$$

$$c_j \geq s_i + M \cdot (y_{ij}^{sc} - 1), \qquad \forall\, (i,j) \in \mathbb{U} \tag{6.5}$$

$$s_j \geq c_i + M \cdot (y_{ij}^{cs} - 1), \qquad \forall\, (i,j) \in \mathbb{U} \tag{6.6}$$

$$c_j \geq c_i + M \cdot (y_{ij}^{cc} - 1), \qquad \forall\, (i,j) \in \mathbb{U} \tag{6.7}$$

$$y_{ij}^{ss} + y_{ji}^{ss} = 1, \qquad \forall\, (i,j) \in \mathbb{U} \tag{6.8}$$

$$y_{ij}^{sc} + y_{ji}^{cs} = 1, \qquad \forall\, (i,j) \in \mathbb{U} \tag{6.9}$$

$$y_{ij}^{cs} + y_{ji}^{sc} = 1, \qquad \forall\, (i,j) \in \mathbb{U} \tag{6.10}$$

$$y_{ij}^{cc} + y_{ji}^{cc} = 1, \qquad \forall\, (i,j) \in \mathbb{U} \tag{6.11}$$

$$x_{ki}^{s} \geq l_i + M \cdot (z_{ki} - 1), \qquad \forall\, i \in V, k \in Q \tag{6.12}$$

$$x_{ki}^{s} \leq l_i + M \cdot (1 - z_{ki}), \qquad \forall\, i \in V, k \in Q \tag{6.13}$$

$$x_{ki}^{c} \geq l_i + M \cdot (z_{ki} - 1), \qquad \forall\, i \in V, k \in Q \tag{6.14}$$

$$x_{ki}^{c} \leq l_i + M \cdot (1 - z_{ki}), \qquad \forall\, i \in V, k \in Q \tag{6.15}$$

$$x_{kj}^{s} \leq l_i + M \cdot (3 - z_{ki} - y_{ij}^{ss} - y_{ji}^{sc}), \qquad \forall\, (i,j) \in \mathbb{U}, k \in Q \tag{6.16}$$

$$x_{kj}^{s} \geq l_i + M \cdot (z_{ki} + y_{ij}^{ss} + y_{ji}^{sc} - 3), \qquad \forall\, (i,j) \in \mathbb{U}, k \in Q \tag{6.17}$$

$$x_{kj}^{c} \leq l_i + M \cdot (3 - z_{ki} - y_{ij}^{sc} - y_{ji}^{cc}), \qquad \forall\, (i,j) \in \mathbb{U}, k \in Q \tag{6.18}$$

$$x_{kj}^{c} \geq l_i + M \cdot (z_{ki} + y_{ij}^{sc} + y_{ji}^{cc} - 3), \qquad \forall\, (i,j) \in \mathbb{U}, k \in Q \tag{6.19}$$

$$x_{ki}^{s} - x_{k-1,i}^{s} \geq 0, \qquad \forall\, i \in V, k \in Q \backslash \{1\} \tag{6.20}$$

$$x_{ki}^{c} - x_{k-1,i}^{c} \geq 0, \qquad \forall\, i \in V, k \in Q \backslash \{1\} \tag{6.21}$$

$$s_i \geq r_i, \qquad \forall\, i \in V \tag{6.22}$$

$$c_i \geq s_i + \sum_{q \in \Theta(i)} p_{iq} \cdot \psi_{iq}, \qquad \forall\, i \in V \tag{6.23}$$

$$s_j - c_i \geq \tau, \qquad \forall\, (i,j) \in \mathbb{R} \tag{6.24}$$

$$L_i \geq (c_i - d_i)/(d_i - r_i), \qquad \forall\, i \in V \tag{6.25}$$

$$L \geq L_i, \qquad \forall\, i \in V \tag{6.26}$$

$$s_i, c_i, L_i, L, x^s_{ki}, x^c_{ki} \in \Re^+, \qquad \forall\, i \in V, k \in Q \tag{6.27}$$

$$y^{ss}_{ij}, y^{sc}_{ij}, y^{cs}_{ij}, y^{cc}_{ij}, z_{ki}, \psi_{iq} \in \{0,1\}, \forall\, (i,j) \in \mathbb{U}, k \in Q, q \in \Theta(i) \tag{6.28}$$

The objective function (6.1) is to minimize L, which is defined as $\max_{i \in V} L_i$. Constraints (6.2) make sure that the total number quay cranes assigned to a particular vessel should be in the legitimate range. Constraints (6.3) link both the decision variables, ψ_{iq} and z_{ki}. Constraints (6.4) to (6.11) show the properties of $y^{ss}_{ij}, y^{sc}_{ij}, y^{cs}_{ij}, y^{cc}_{ij}$. Constraints (6.12) to (6.19) indicate that once a quay crane is assigned to a certain vessel, the quay crane can only work on that specific vessel until it leaves port. Constraints (6.20) and (6.21) state that quay cranes can not cross each other in the planning horizon. In Constraints (6.22) and (6.23), they ensure that the service start time of Vessel i should be later than the arrival time of the vessel and meanwhile, the completion time of Vessel i should be greater than the sum of start time, s_i, and realized processing time, $\sum_{q \in \Theta(i)} p_{iq} \cdot \psi_{iq}$. For the vessel pair (i,j) in $\mathbb{R}$, there should be a sufficiently large time gap (i.e., τ) between the completion time of Vessel i and the start time of Vessel j. Constraints (6.25) and (6.26) define the decision variables, L_i and L, respectively. Finally, Constraints (6.27) to (6.28) declare the domains for all the decision variables.

6.2.3 Observations and valid inequalities

Based on $[\mathcal{F}]$ originally developed in Liu et al. (2006), in this subsection, two observations can be made on $[\mathcal{F}]$ and additionally, several valid inequalities which are redundant for $[\mathcal{F}]$ but can be used to strengthen the master problem of the proposed algorithm (introduced in the next section), is presented.

First of all, according to the definition of vessel pair set $\mathbb{U}$, as long as (i,j) is an element of $\mathbb{U}$, (j,i) should also be included in $\mathbb{U}$. In other words, the set $\mathbb{U}$ is sort of "symmetric". In light

of this observation, actually, constraints (6.9) and (6.10) are the same. Therefore, one of them can be dropped from $[\mathcal{F}]$ to avoid the repetition. Moreover, in the original formulation, there is lacking a very important restriction on the decision variable ψ_{iq}:

$$\sum_{q\in\Theta(i)} \psi_{iq} = 1, \qquad \forall\, i \in V \tag{6.29}$$

The validness of equations (6.29) is evident since they make sure that there is only one kind of quay crane assignment plan for a certain vessel. And without the enforcement of constraints (6.29), it is possible that ψ_{iq_1} and ψ_{iq_2} can both take 1 simultaneously while $\overline{q}^i \geq q_1 + q_2 \geq \underline{q}^i$. In this case, the term $\sum_{q\in\Theta(i)} p_{iq} \cdot \psi_{iq}$ in constraints (6.23) is equal to $p_{iq_1} + p_{iq_2}$ and fails to represent a valid processing time for Vessel i.

Next, 3 kinds of valid inequalities for $[\mathcal{F}]$ are described. For ease of expression, it is necessary to define a new vessel pair set, $\mathbb{U}'$, i.e., the set of vessel pairs (i,j) such that $(i,j) \in \mathbb{U}$ and additionally $l_i > l_j$.

The first kind of valid inequalities is about the connections among y_{ij}^{ss}, y_{ij}^{sc}, y_{ij}^{cs} and y_{ij}^{cc}. By definition, for $i,j \in V, i \neq j$, $y_{ij}^{ss} = 1$ if and only if $s_i \leq s_j$ and $y_{ij}^{sc} = 1$ if and only if $s_i \leq c_j$. Since $s_j \leq c_j$, if $y_{ij}^{ss} = 1$, it follows that $s_i \leq c_j$ which implies that $y_{ij}^{sc} = 1$. In other words, if $y_{ij}^{ss} = 1$, then $y_{ij}^{sc} = 1$. This relationship between y_{ij}^{ss} and y_{ij}^{sc} can be expressed as the inequalities shown in (6.30), i.e., $y_{ij}^{sc} \geq y_{ij}^{ss}$. For the case of y_{ij}^{cc} and y_{ij}^{cs}, the analogous analysis can be applied to shown that $y_{ij}^{cc} \geq y_{ij}^{cs}$.

$$y_{ij}^{sc} \geq y_{ij}^{ss}, y_{ij}^{cc} \geq y_{ij}^{cs}, \qquad \forall\, i,j \in V, i \neq j \tag{6.30}$$

The next type of valid inequalities summarized in (6.31) is based on the fact that if Cranes k and k' (say, $k < k' - 1$) are assigned to Vessel i at the same time, the quay crane located at the interval between these two quay cranes (i.e., quay crane k_e, $k < k_e < k'$) should also be assigned to Vessel i due to the physical constraint of Non-crossing.

$$z_{k_e,i} \geq z_{k,i} + z_{k',i} - 1, \qquad \forall\, i \in V, k, k_e, k' \in Q, k < k_e < k' \tag{6.31}$$

Mathematically speaking, it claims that if both $z_{k,i}$ and $z_{k',i}$ take 1 simultaneously, then $z_{k_e,i}$ should be 1 as well. It can be observed that inequalities (6.31) are able to serve the function mentioned above.

Finally, valid inequalities (6.32) link the decision variables y_{ij}^{cs} and z_{ki} in order to forbid the assignment plan shown in Figure (6.1) and Figure (6.2). In both cases, Vessel i and Vessel j are processed simultaneously at certain point of time. In Figure (6.1), assume that Cranes k and k' ($k < k'$) are assigned to Vessel j and Vessel i, respectively, where Vessel i is moored above Vessel j (i.e., $(i,j) \in \mathbb{U}'$). Since at certain time point, Vessel i and Vessel j are processed simultaneously, such kind of quay crane assignment is doomed to violate the non-crossing requirement. In the case shown in Figure (6.2), identical quay crane is assigned to two vessels which can be handled simultaneously. Since a quay crane assigned to a vessel stays at the berthing location of the vessel until its completion, the assignment plan shown in Figure (6.2) is also infeasible for real-life practice. For prevention, it should make sure that Vessels i and j, $(i,j) \in \mathbb{U}'$, can not be processed simultaneously (i.e., $y_{ij}^{cs} + y_{ji}^{cs} = 0$) if Cranes k and k' are assigned to Vessel i and Vessel j, respectively (i.e., $z_{ki} + z_{k'j} = 2$).

$$y_{ij}^{cs} + y_{ji}^{cs} \geq z_{ki} + z_{k'j} - 1, \qquad \forall\, (i,j) \in \mathbb{U}', k, k' \in Q, k \leq k' \tag{6.32}$$

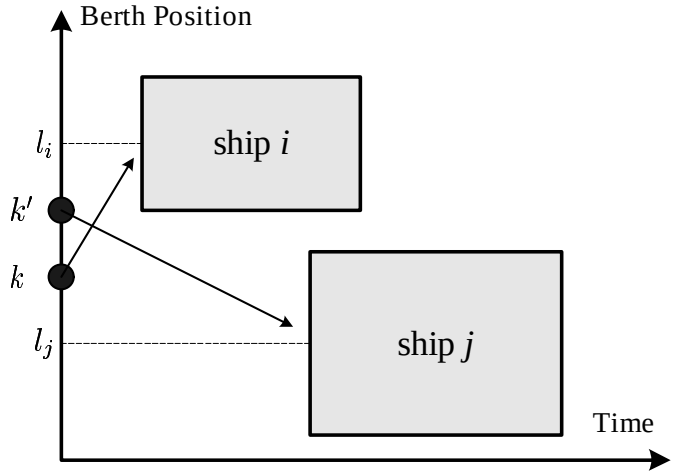


Figure 6.1: An illustration for Valid inequalities (6.32) when $k < k'$

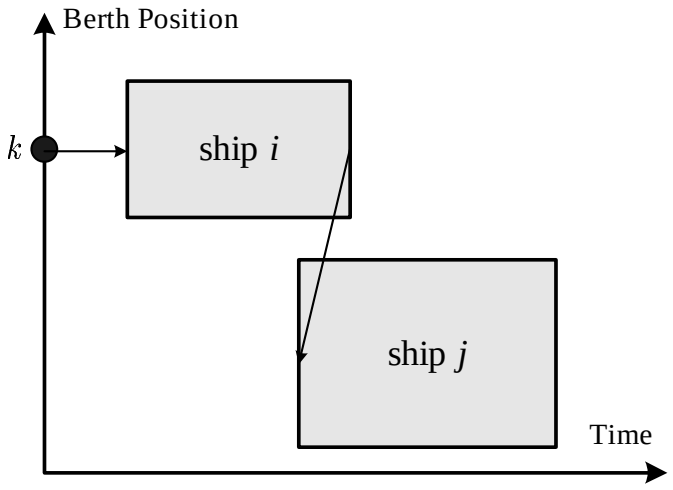


Figure 6.2: An illustration for Valid inequalities (6.32) when $k = k'$

6.3 CBC Algorithm

Based on the aforementioned observations, the formulation of Liu, Wan, and Wang should be updated as below:

$$
\begin{aligned}
[\mathcal{F}] \quad & \min \quad L \\
& s.t. \\
& \qquad \text{Constraints (6.2) to (6.9)} \\
& \qquad \text{Constraints (6.11) to (6.29)}
\end{aligned}
$$

In this section, the CBC algorithm will be developed to solve $[\mathcal{F}]$ (assuming that $[\mathcal{F}]$ is a feasible problem). The CBC algorithm was firstly proposed in Hooker (2000) and extended by Codato and Fischetti (2006) to solve mixed integer programming with special structures. In the literature of port container terminal management, the CBC algorithm has been successfully adopted to find the optimal solution for integrated yard truck and yard crane scheduling problem (Cao et al., 2010). Compared with traditional Benders' Cuts method, the CBC algorithm tries

to derive Benders' Cuts from the minimal set of inconsistencies of subproblem itself rather than the dual information of the original subproblem.

Note that the decision process of $[\mathcal{F}]$ can be naturally decomposed into two phases. The first phase of the proposed problem is called [Master], which is the decision to determine the service start time, the total number of quay crane assigned, and the service completion time of each vessel. Sequenctially, the decision process of the second phase (denoted as [Slave]) starts and takes the output from the first phase as input with the objective to check whether the output from the previous phase is feasible in the sense that the Non-crossing requirement among quay cranes can be satisfied or not. Specifically, the two-phase decision process for $[\mathcal{F}]$ is summarized in the sequel.

For conciseness, vector notations are defined:

- $\mathbf{x}$, the vector of $\{x^s_{ki}, x^c_{ki}\}_{i\in V, k\in Q}$
- $\mathbf{y}$, the vector of $\{y^{ss}_{ij}, y^{sc}_{ij}, y^{cs}_{ij}, y^{cc}_{ij}\}_{(i,j)\in\mathbb{U}}$
- $\mathbf{z}$, the vector of $\{z_{ki}\}_{i\in V, k\in Q}$

$$
\begin{aligned}
[\text{Master}] \quad \min \quad & L \\
s.t. \quad & \\
& \text{Constraints (6.2) to (6.9)} \\
& \text{Constraints (6.11)} \\
& \text{Constraints (6.22) to (6.26)} \\
& \text{Constraints (6.28) to (6.29)} \\
& s_i, c_i, L_i, L \in \Re^+, \qquad \forall\, i \in V, k \in Q
\end{aligned} \tag{6.33}
$$

Let $(\mathbf{y}^*, \mathbf{z}^*)$ denote the output by solving the current [Master]. Then the second phase problem [Slave] is,

[Slave] min $\mathbf{0} \cdot \mathbf{x}$

s.t.

$$x^s_{ki} \geq l_i + M \cdot (z^*_{ki} - 1), \qquad \forall\, i \in V, k \in Q \tag{6.34}$$

$$x^s_{ki} \leq l_i + M \cdot (1 - z^*_{ki}), \qquad \forall\, i \in V, k \in Q \tag{6.35}$$

$$x^c_{ki} \geq l_i + M \cdot (z^*_{ki} - 1), \qquad \forall\, i \in V, k \in Q \tag{6.36}$$

$$x^c_{ki} \leq l_i + M \cdot (1 - z^*_{ki}), \qquad \forall\, i \in V, k \in Q \tag{6.37}$$

$$x^s_{kj} \leq l_i + M \cdot (3 - z^*_{ki} - y^{ss*}_{ij} - y^{sc*}_{ji}), \forall\, (i,j) \in \mathbb{U}, k \in Q \tag{6.38}$$

$$x^s_{kj} \geq l_i + M \cdot (z^*_{ki} + y^{ss*}_{ij} + y^{sc*}_{ji} - 3), \forall\, (i,j) \in \mathbb{U}, k \in Q \tag{6.39}$$

$$x^c_{kj} \leq l_i + M \cdot (3 - z^*_{ki} - y^{sc*}_{ij} - y^{cc*}_{ji}), \forall\, (i,j) \in \mathbb{U}, k \in Q \tag{6.40}$$

$$x^c_{kj} \geq l_i + M \cdot (z^*_{ki} + y^{sc*}_{ij} + y^{cc*}_{ji} - 3), \forall\, (i,j) \in \mathbb{U}, k \in Q \tag{6.41}$$

$$x^s_{ki} - x^s_{k-1,i} \geq 0, \qquad \forall\, i \in V, k \in Q \backslash \{1\} \tag{6.42}$$

$$x^c_{ki} - x^c_{k-1,i} \geq 0, \qquad \forall\, i \in V, k \in Q \backslash \{1\} \tag{6.43}$$

$$x^s_{ki}, x^c_{ki} \in \Re^+, \qquad \forall\, i \in V, k \in Q \tag{6.44}$$

Given a $(\mathbf{y}^*, \mathbf{z}^*)$, if there exists a feasible solution for [Slave], say $\mathbf{x}^*$, it can be concluded that $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ must also be an optimal solution for $[\mathcal{F}]$. However, if [Slave] happens to be infeasible, it provides a hint that useful information can be explored from the Minimal Infeasible Subsystem (MIS) of the polyhedron $P = \{\mathbf{x} \,|\, \text{Constraints}\,(6.34) - (6.44)\}$. Here the MIS of P is defined as any inclusion-minimal set of row indices of P such that the linear system has no feasible solution.

As pointed out in Codato and Fischetti (2006), in order to derive at least one combinatorial benders' cut from [Slave], one needs to convert the polyhedron P into an equivalent but special form, i.e., a polyhedron comprised of a set of "conditional" linear constraints plus a set of "unconditional" constraints. Here the set of "conditional" linear constraints should follow the

prototype,

$$m \cdot \alpha^* + \mathbf{a}^T \boldsymbol{\beta} \geq b \tag{6.45}$$

where α^* is the realized value of a certain binary decision variable from [Master], $\boldsymbol{\beta}$ is a vector for continuous variables, and m ($m \neq 0$), b, $\mathbf{a}$ are constants. And the set of "unconditional" constraints only involves the continuous variables $\boldsymbol{\beta}$. That is, a set of linear constraints similar to Inequality (6.45) but m takes the value 0.

To achieve the conversion for the current P, one possible way is to introduce continuous copies of decision variables $\mathbf{y}$ and $\mathbf{z}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$, into the problem [Slave]. After the conversion, the new but equivalent [Slave] problem is given below. Denote the row indices for constraints (6.56) to (6.60) in the current [Slave] by set Δ and the row indices for the MIS of current [Slave] by set Λ if the input $(\mathbf{y}^*, \mathbf{z}^*)$ makes [Slave] infeasible. Note that the set $\Gamma = \Delta \cap \Lambda$ can not be an empty set, otherwise it implies that the original problem $[\mathcal{F}]$ is an infeasible problem which contradicts the previous assumption. Additionally, the set Γ can be decomposed into two mutually exclusive subsets Γ_y and Γ_z, where Γ_y contains the row indices related to Constraints (6.56) to (6.59) and Γ_z contains the row indices related to Constraints (6.60). For $t \in \Gamma_y$, there is an one to one mapping relationship between t and a particular element of $\hat{\mathbf{y}}$ (ultimately $\mathbf{y}$, since $\hat{\mathbf{y}}$ is a continuous copy of $\mathbf{y}$). Denote this binary decision variable as $\mathbf{y}(t)$. By the same token, for each $t \in \Gamma_z$, it links with an unique binary variable in $\mathbf{z}$ called $\mathbf{z}(t)$.

[Slave] $\min \quad \mathbf{0} \cdot \mathbf{x} + \mathbf{0} \cdot \hat{\mathbf{y}} + \mathbf{0} \cdot \hat{\mathbf{z}}$

s.t.

$$x^s_{ki} \geq l_i + M \cdot (\hat{z}_{ki} - 1), \qquad \forall\, i \in V, k \in Q \tag{6.46}$$

$$x^s_{ki} \leq l_i + M \cdot (1 - \hat{z}_{ki}), \qquad \forall\, i \in V, k \in Q \tag{6.47}$$

$$x^c_{ki} \geq l_i + M \cdot (\hat{z}_{ki} - 1), \qquad \forall\, i \in V, k \in Q \tag{6.48}$$

$$x^c_{ki} \leq l_i + M \cdot (1 - \hat{z}_{ki}), \qquad \forall\, i \in V, k \in Q \tag{6.49}$$

$$x^s_{kj} \leq l_i + M \cdot (3 - \hat{z}_{ki} - \hat{y}^{ss}_{ij} - \hat{y}^{sc}_{ji}), \forall\, (i,j) \in \mathbb{U}, k \in Q \tag{6.50}$$

$$x^s_{kj} \geq l_i + M \cdot (\hat{z}_{ki} + \hat{y}^{ss}_{ij} + \hat{y}^{sc}_{ji} - 3), \forall\,(i,j) \in \mathbb{U}, k \in Q \tag{6.51}$$

$$x^c_{kj} \leq l_i + M \cdot (3 - \hat{z}_{ki} - \hat{y}^{sc}_{ij} - \hat{y}^{cc}_{ji}), \forall\,(i,j) \in \mathbb{U}, k \in Q \tag{6.52}$$

$$x^c_{kj} \geq l_i + M \cdot (\hat{z}_{ki} + \hat{y}^{sc}_{ij} + \hat{y}^{cc}_{ji} - 3), \forall\,(i,j) \in \mathbb{U}, k \in Q \tag{6.53}$$

$$x^s_{ki} - x^s_{k-1,i} \geq 0, \qquad \forall\, i \in V, k \in Q\backslash\{1\} \tag{6.54}$$

$$x^c_{ki} - x^c_{k-1,i} \geq 0, \qquad \forall\, i \in V, k \in Q\backslash\{1\} \tag{6.55}$$

$$\hat{y}^{ss}_{ij} = y^{ss*}_{ij}, \qquad \forall\,(i,j) \in \mathbb{U} \tag{6.56}$$

$$\hat{y}^{sc}_{ij} = y^{sc*}_{ij}, \qquad \forall\,(i,j) \in \mathbb{U} \tag{6.57}$$

$$\hat{y}^{cs}_{ij} = y^{cs*}_{ij}, \qquad \forall\,(i,j) \in \mathbb{U} \tag{6.58}$$

$$\hat{y}^{cc}_{ij} = y^{cc*}_{ij}, \qquad \forall\,(i,j) \in \mathbb{U} \tag{6.59}$$

$$\hat{z}_{ki} = z^*_{ki}, \qquad \forall\, i \in V, k \in Q \tag{6.60}$$

$$x^s_{ki}, x^c_{ki}, \hat{z}_{ki}, \hat{y}^{ss}_{ij}, \hat{y}^{sc}_{ij}, \hat{y}^{cs}_{ij}, \hat{y}^{cc}_{ij} \in \Re^+, \quad \forall\,(i,j) \in \mathbb{U}, k \in Q \tag{6.61}$$

The fundamental rationale of the CBC algorithm is: as long as $(\mathbf{y}^*, \mathbf{z}^*)$ makes [Slave] infeasible, it indicates that at least one binary variable in $\mathbf{y}$ and $\mathbf{z}$ has to be changed to break the infeasibility. This statement can be translated by a linear inequality called the Combinatorial Benders' Cut:

$$\sum_{t \in \Gamma_y\,:\,\mathbf{y}^*(t)=0} \mathbf{y}(t) + \sum_{t \in \Gamma_y\,:\,\mathbf{y}^*(t)=1} (1 - \mathbf{y}(t)) + \sum_{t \in \Gamma_z\,:\,\mathbf{z}^*(t)=0} \mathbf{z}(t) + \sum_{t \in \Gamma_z\,:\,\mathbf{z}^*(t)=1} (1 - \mathbf{z}(t)) \geq 1 \tag{6.62}$$

One or more Combinatorial Benders' Cuts of this type can be derived in correspondence to a given infeasible $(\mathbf{y}^*, \mathbf{z}^*)$ and be added back to [Master]. By iterating the above procedure, an exact solution of $[\mathcal{F}]$ can be found in the spirit of Benders' decomposition (Codato and Fischetti, 2006).

In the application of the CBC algorithm for $[\mathcal{F}]$, as pointed out in the previous section, constraints (6.30) to (6.32) are valid inequalities for $[\mathcal{F}]$ and they can be added to the [Master] for the purposes:

1. help to solve the [Master] subproblem as quickly as possible.

2. make sure that the solution generated by the [Master] subproblem will violate the quay crane Non-crossing requirement as least as possible and satisfy the constraints of the [Slave] subproblem. Usually, by decomposing the original problem into the [Master] subproblem and the [Slave] subproblem, the connection between the [Master] subproblem and the [Slave] subproblem would become weak in the sense that the solution output from the [Master] subproblem "tends to" make the [Slave] subproblem infeasible. Therefore, adding these valid inequalities of the original problem to the [Master] subproblem can guide the [Master] subproblem to seek suitable solutions which are superior in the domain of the [Master] subproblem but also enhance the capability to fit the constraints of the [Slave] subproblem.

Such a strategy to add these inequalities is considered to be a contribution to the classic CBC algorithm. In sum, the following summarizes the procedures of the proposed CBC algorithm:

Step 1: Decompose the $[\mathcal{F}]$ into [Master] and [Slave]. Add the Valid inequalities (6.30) to (6.32) to strengthen [Master]. Transform the [Slave] by introducing continuous copies of binary decision variables, $\mathbf{y}$ and $\mathbf{z}$. Let iteration counter ι equal to 1. Initialize the row index set Δ according to constraints (6.56) to (6.60) and let row index set Γ equal to $\emptyset$.

Step 2: At Iteration ι, solve the [Master] problem to optimality and obtain $(\mathbf{y}^*_\iota, \mathbf{z}^*_\iota)$.

Step 3: With $(\mathbf{y}^*_\iota, \mathbf{z}^*_\iota)$, solve [Slave]. If there is one feasible solution for [Slave], say $\mathbf{x}^*{}_\iota$, then stop the CBC algorithm and return the optimal solution $(\mathbf{x}^*_\iota, \mathbf{y}^*_\iota, \mathbf{z}^*_\iota)$. However, if [Slave] is infeasible, examine the MIS of [Slave] and get the row index set Λ. Update Γ, $\Gamma \leftarrow \Delta \cap \Lambda$. Through Γ, generate Combinatorial Benders' Cuts in the form of (6.62) and add them back to [Master].

Step 4: $\iota \leftarrow \iota + 1$. Go to Step 2.

6.4 Computational Experiments

Numerical experiments are designed to evaluate the performance of the proposed CBC algorithm for $[\mathcal{F}]$. For problem generation, the strategy adopted in Liu et al. (2006) is used. The planning

horizon is set to one week and the berth length is fixed to be 40 (bay lengths). The numbers of incoming vessels and quay cranes considered for experiments are $\{20, 24, 26\}$ and $\{8, 10, 12\}$, respectively. Therefore, there are 9 vessel-quay crane combinations. For each vessel-quay crane combination, 30 instances are randomly generated. Totally, 270 instances are used to test the performance of the proposed CBC algorithm.

For the generation of each particular instance, the vessel arrival times are generated uniformly within the planning horizon. The size of each vessel is chosen randomly in the range $[8, 20]$. The workload of each bay is randomly generated from the uniform distribution between 1 hour and 6 hours. To estimate the scheduled departure time for each vessel, first of all, the processing time for each vessel needs to be estimated beforehand. Following the tactics of Liu et al., the number of quay cranes assigned to a vessel is assumed proportional to the size of the vessel (a quay crane could be allocated for every 4 bays). Hence, the planned processing time of the vessel is calculated by dividing the total workload of the vessel by the number of quay cranes that the vessel is assumed to take. Given the information of both the arrival time and the estimated processing time for each vessel, the corresponding scheduled departure time could be obtained accordingly. Next, the key input, p_{iq}, the processing time when q quay cranes are assigned to process Vessel i, can be estimated by heuristics such as the ones in Bierwirth and Meisel (2009) or the heuristics introduced in Chapter 4. Furthermore, in order to initialize the three vessel pair sets, $\mathbb{R}$, $\mathbb{R}'$, and $\mathbb{U}$, the berth allocation problem for all the incoming vessels are solved by GRASP proposed in Chapter 3. Finally, let $\underline{q}^i = 1$ and $\overline{q}^i = 5$, $i \in V$, $\tau = 1$ hour.

The CBC algorithm is coded in C++ under the ILOG Concert Technology framework (version 12.1). As the benchmark method, the branch and cut method in CPLEX is adopted for comparison. All experiments are executed in a PC with 2.40 GHz CPU and 3 GB RAM. Table 6.1 summarizes the computational results for the 270 instances. For example, when the number of vessels is 20 and the number of quay cranes is 8, the average computational time consumed by CBC algorithm to find the optimal solutions is 28.3 seconds while for CPLEX, the average computational time is 378.3 seconds. It can be observed that by fixing the number of quay cranes, as the number of vessels increases, $[\mathcal{F}]$ becomes more and more challenging to solve; however, conversely, when the number of vessels is fixed, as the number of quay cranes

increases, $[\mathcal{F}]$ seems much easier to solve. One possible explanation for such phenomena is that: since the berth length is set to 40 and the average size for a vessel is around 14 $(0.5 \times (8+20))$, therefore, the average number of vessels that can be processed simultaneously is 2 to 3; if in most of the instances, the average number of vessels that can be processed simultaneously is 2 and note that $\overline{q}^i$=5, than when the number of quay cranes along the berth is greater or equal to 10, the part of quay crane assignment in $[\mathcal{F}]$ would become trivial and thus it reduces the complexity of the entire problem. Throughout the 270 instances, it is evident that the proposed CBC algorithm outperforms the B&C method in CPLEX. For instance, when the number of vessels is 26 and the number of quay cranes is 8, on average, CPLEX would consume 14 hours to complete the search of the optimal solution while only 9 minutes is required for the CBC algorithm, which is 95 times faster.

Table 6.1: Computational results (average computation time for CBC in seconds, average computation time for CPLEX in seconds)

	Number of vessels		
Number of QCs	20	24	26
8	(28.3,378.3)	(138.0,6103.5)	(539.7,51174.1)
10	(3.6,105.0)	(12.0,465.5)	(28.3,859.5)
12	(2.9,22.9)	(9.4,114.3)	(12.9,121.1)

Table 6.2: The incumbent solution found by CPLEX when CBC hits the optimal one

Instance	CBC Time(s)	CBC Optimal	CPLEX Incumbent	Gap(%)
1	49	1.88	2.69	43.59
2	34	1.43	1.60	12.00
3	38	3.22	4.90	52.07
4	714	4.62	4.85	5.00
5	65	2.23	3.11	39.46
6	21	1.64	2.36	44.44
7	424	3.73	3.93	5.36
8	21	2.06	3.07	48.92
9	233	2.83	2.92	2.94
10	328	3.18	3.18	0.00
11	61	1.56	3.73	138.55
12	16	1.54	3.55	130.45
13	12	1.85	4.62	150.00
14	19	2.29	4.17	82.29
15	22	2.00	5.60	180.00
16	25	1.69	5.57	229.22
17	222	4.60	5.46	18.73
18	11	3.69	4.85	31.25
19	61	2.20	2.53	15.15
20	294	2.67	3.42	28.12
21	112	4.00	4.80	20.00
22	12	1.79	6.00	236.00
23	69	1.89	13.11	594.12
24	12	3.27	4.55	38.89
25	229	2.33	2.53	8.57
26	35	4.00	4.63	15.63
27	83	1.92	2.38	24.00
28	97	2.07	2.71	31.03
29	808	2.92	3.18	8.85
30	12	4.42	6.50	47.17
			Average	76.06

Additionally, it is well know that CPLEX solver is usually capable of finding good solutions very quickly at the beginning of search. Therefore, for fair comparison, the 30 instances when the number of vessels is 24 and the number of quay cranes is 8 are used to demonstrate the true potential of the proposed algorithm. When these 30 instances are solved by CPLEX, the search will be forced to terminate if the current running time of CPLEX exceeds the corresponding overall computation time of the CBC algorithm. Table 6.2 lists the incumbent solutions found by CPLEX when CBC has already finished the search. Here the Gap is defined as:

$$\text{Gap} = \frac{\text{CPLEX Incumbent} - \text{CBC Optimal}}{\text{CBC Optimal}} \times 100\%$$

For all the 30 instances, except Instance 10, the incumbent solutions found by CPLEX are not optimal and the average gap is 76%.

6.5 Summary

The quayside operation problem is one of the key components in the management system for a port container terminal. In this Chapter, the integrated models proposed in the previous studies to address the quayside operation problem are examined and one of the potential frameworks is identified, i.e., the model presented in Liu et al. (2006). A new method called the CBC algorithm is developed to solve the Berth-level model in the framework. The computational experiment conducted in this research shows that the proposed approach is more efficient than the B&C algorithm embedded in CPLEX.

Chapter 7

An Integrated Model for the Quayside Operation Problem with Discrete Berths

7.1 Background

Based on the literature review presented in Chapter 2, it is clear that there is still a lack of more reasonable or comprehensive integrated models for the quayside operation problem. In this chapter, a new integrated quayside operation model, an extension of Liu et al. (2006), will be proposed in the context of discrete berths.

7.2 Problem Description and Mathematical Model

In this section, the description of the proposed problem and its mathematical formulation will be presented. For a port container terminal, suppose the quay line is partitioned into m berths (denote the set of the berths as B; the cardinality of B, $|B| = m$) and only one vessel can be moored at each berth at a time. There are q quay cranes equipped along the quay and they are mounted on the same tracks, which forbids them from crossing each other at any instance. During the planning horizon, given the information of the n incoming vessels (denote the set

of the vessels as V and $|V| = n$), the decision for the proposed quayside operation problem is two-fold: firstly, the port operators need to assign the q available quay cranes to each berth and keep the assigned number intact throughout the planning horizon such that it becomes possible to evaluate the length of the processing time for a particular vessel when it moors at a certain berth; secondly, based on the quay crane assignment plan for each berth, the port operators also need to determine the exact berthing location and berthing time for each incoming vessel. The objective of the proposed problem is to minimize the total turnaround time for all vessels by following the convention of the researches on discrete berth allocation problem such as (Imai et al., 2001; Monaco and Sammarra, 2007).

At the first glance, it may seem that the assumption for the quay crane assignment, i.e., the number of quay crane assigned to a particular berth should not be changed during the planning horizon, is somewhat strong. However, since the length of the planning horizon is artificially determined by port operators, if they shorten the length to one shift (i.e., 4 or 6 hours depending on the policy of the port container terminal) or two shifts, then the quay crane assignment mode will vary frequently and appear dynamic. Besides, on the solution part, the algorithms developed in this research (described in Sections 7.4 and 7.4) are capable to solve all relatively large scale problems within one hour (on average). Therefore, such assumption is considered to be reasonable.

The following lists the parameters and decision variables for the proposed problem. Note that for discrete berths, usually there are a minimum number and a maximum number of quay cranes that can be deployed to Berth i (denote as $\underline{c}^i$ and $\overline{c}^i$, respectively). The rationale for the upper limit of the quay cranes for a particular berth is to take the available quay crane resource and operation limits of the berth into account. Moreover, in order to achieve a feasible quay crane assignment plan, it is reasonable to assume that for input data, $q \geq \sum_{i \in B} \underline{c}^i$ should always hold. There are two key input data for the proposed problem, viz., p_{vj} and t_{ij}. p_{vj} is the processing time when v quay cranes are assigned to Vessel j. Actually, given v and j, to obtain the value p_{vj} is equivalent to solve a QCSP which is $\mathcal{NP}$-hard. Fortunately, in the literature, several fast but effective heuristics for the QCSP have been proposed like the ones in Sammarra et al. (2007) and Bierwirth and Meisel (2009). With the facilitation of these

heuristics, the impact of the quay crane scheduling on the overall quayside operation can be well estimated. Another key input, t_{ij}, is the approximate extra transportation time for the loading or unloading the containers in Vessel j if Vessel j is moored at Berth i. The parameter is used to account for the fact that for Vessel j, its extra transportation cost might not be the same for a different berth allocation plan.

7.2.1 Parameters

- V, the set of vessels to be berthed
- B, the set of berths
- Υ, an ordered set of indexes
- n, $n = |\Upsilon| = |V|$
- m, $m = |B|$
- s_i, the time when Berth i becomes available for berthing
- a_j, the arrival time of Vessel j
- q, the number of available quay cranes
- $\underline{c}^i$, the minimum number of quay cranes that can be deployed to Berth i
- $\overline{c}^i$, the maximum number of quay cranes that can be deployed to Berth i
- $V(i) = \{j \in V \mid a_j > s_i\}, \forall i \in B$, the set of vessels that arrive at the port after s_i
- $\Upsilon(k) = \{k^{'} \in \Upsilon \mid k^{'} < k\}, \forall k \in \Upsilon$, the set of position indexes preceding k
- $\Theta(i) = \{v \mid \underline{c}^i \leq v \leq \overline{c}^i\}, \forall i \in B$, the set of legitimate quay crane numbers for Berth i
- p_{vj}, the processing time when v quay cranes are assigned to process Vessel j
- t_{ij}, the extra transportation time for the loading/unloading containers in Vessel j if Vessel j is moored at Berth i

7.2.2 Decision variables

- $x_{ijk} \in \{0,1\}$ is equal to 1 if and only if Vessel j is the k^{th} vessel moored at Berth i
- y_{ik}, the idle time at Berth i between the departure of the $(k-1)^{\text{th}}$ vessel and the arrival of the k^{th} vessel
- $z_{iv} \in \{0,1\}$ is equal to 1 if and only if v quay cranes are assigned to Berth i during the planning horizon
- h_{ij}, the handling time (including processing time and the extra transportation time) of Vessel j in the case that it is moored at Berth i

7.2.3 Formulations

Based on the compact formulation of the discrete BAP proposed in Monaco and Sammarra (2007), the proposed integrated model for the quayside operation problem can be directly derived. The following formulation $[\mathcal{P}_0]$ is the mathematical model for the proposed problem.

$$[\mathcal{P}_0] \min \quad \sum_{i\in B}\sum_{j\in V}\sum_{k\in\Upsilon}\{(n-k+1)h_{ij}+s_i-a_j\}x_{ijk} + \sum_{i\in B}\sum_{k\in\Upsilon}(n-k+1)y_{ik} \tag{7.1}$$

$$s.t.$$

$$\sum_{j\in V(i)}(a_j-s_i)x_{ijk} - \sum_{l\in\Upsilon(k)}(y_{il}+\sum_{j\in V}h_{ij}x_{ijl}) - y_{ik} \le 0, \quad \forall\, i\in B, k\in\Upsilon \tag{7.2}$$

$$h_{ij} = \sum_{v\in\Theta(i)} p_{vj}\cdot z_{iv} + t_{ij}, \qquad \forall\, i\in B, j\in V \tag{7.3}$$

$$\sum_{i\in B}\sum_{k\in\Upsilon} x_{ijk} = 1, \qquad \forall\, j\in V \tag{7.4}$$

$$\sum_{j\in V} x_{ijk} \le 1, \qquad \forall\, i\in B, k\in\Upsilon \tag{7.5}$$

$$\sum_{v\in\Theta(i)} z_{iv} = 1, \qquad \forall\, i\in B \tag{7.6}$$

$$\sum_{i\in B}\sum_{v\in\Theta(i)} v\cdot z_{iv} \le q \tag{7.7}$$

$$x_{ijk} \in \{0,1\}, \qquad \forall\, i \in B, j \in V, k \in \Upsilon \tag{7.8}$$

$$y_{ik} \geq 0, \qquad \forall\, i \in B, k \in \Upsilon \tag{7.9}$$

$$z_{iv} \in \{0,1\}, \qquad \forall\, i \in B, v \in \Theta(i) \tag{7.10}$$

The objective (7.1) is to minimize the sum of the turnaround time for all vessels. Constraints (7.2) and (7.9) define the values of the idle times. Constraints (7.3) show that the value of h_{ij} is the sum of the realized processing time for Vessel j (i.e., $\sum_{v\in\Theta(i)} p_{vj} \cdot z_{iv}$) and the extra transportation time t_{ij}. Constraints (7.4) restrict that each Vessel i is served in exactly one position k at Berth i. Constraints (7.5) ensure that each berth serves no more than one vessel at a time. Constraints (7.6) and (7.7) are served for quay crane assignment. Constraints (7.6) make sure that during the planning horizon only a fixed number of quay cranes are deployed to a certain berth. While Constraints (7.7) guarantee that the total number of deployed quay cranes should not excel the number of available quay cranes. Finally, Constraints (7.8) and (7.10) define the domains for decision variables x_{ijk} and z_{iv}.

Although $[\mathcal{P}_0]$ is straightforward, it is nonlinear. To transform it to an equivalent linear counterpart, there are two steps. First of all, by substituting h_{ij} with $\sum_{v\in\Theta(i)} p_{vj} \cdot z_{iv} + t_{ij}$ and introducing an auxiliary variable w_{ivjk}, $w_{ivjk} = z_{iv} \cdot x_{ijk}$, $[\mathcal{P}_0]$ can be transformed to $[\mathcal{P}_1]$. Note that in $[\mathcal{P}_1]$, the nonlinear part is confined to Constraints (7.13). Therefore, if Constraints (7.13) can be replaced by linear counterparts, then the original nonlinear formulation will be converted to an linear one. Next, by changing Constraints (7.13) and (7.14) to Constraints (7.17) and (7.18), it ends up to a new formulation denoted as $[\mathcal{P}_2]$.

$$\begin{aligned}
[\mathcal{P}_1] \min \quad & \sum_{i\in B}\sum_{j\in V}\sum_{k\in\Upsilon}\sum_{v\in\Theta(i)} (n-k+1)p_{vj} \cdot w_{ivjk} \\
& + \sum_{i\in B}\sum_{j\in V}\sum_{k\in\Upsilon}\{(n-k+1)t_{ij} + s_i - a_j\}x_{ijk} \\
& + \sum_{i\in B}\sum_{k\in\Upsilon}(n-k+1)y_{ik} \qquad (7.11) \\
s.t. \quad & \\
& \sum_{j\in V(i)}(a_j - s_i)x_{ijk} - \sum_{l\in\Upsilon(k)}(y_{il} + \sum_{j\in V} t_{ij} \cdot x_{ijl})
\end{aligned}$$

$$-y_{ik} - \sum_{l\in\Upsilon(k)}\sum_{j\in V}\sum_{v\in\Theta(i)} p_{vj}\cdot w_{ivjl} \le 0 \quad \forall\, i\in B, k\in\Upsilon \tag{7.12}$$

$$w_{ivjk} = z_{iv}\cdot x_{ijk}, \qquad \forall\, i\in B, j\in V, v\in\Theta(i), k\in\Upsilon \tag{7.13}$$

$$w_{ivjk}\in\{0,1\}, \qquad \forall\, i\in B, j\in V, v\in\Theta(i), k\in\Upsilon \tag{7.14}$$

$$\text{Constraints (7.4) to (7.10)} \tag{7.15}$$

$$\begin{aligned}[\mathcal{P}_2]\min \quad & \sum_{i\in B}\sum_{j\in V}\sum_{k\in\Upsilon}\sum_{v\in\Theta(i)} (n-k+1)p_{vj}\cdot w_{ivjk} \\ & + \sum_{i\in B}\sum_{j\in V}\sum_{k\in\Upsilon}\{(n-k+1)t_{ij}+s_i-a_j\}x_{ijk} \\ & + \sum_{i\in B}\sum_{k\in\Upsilon}(n-k+1)y_{ik} \\ s.t. \quad &\end{aligned}$$

$$w_{ivjk} \ge z_{iv} + x_{ijk} - 1, \qquad \forall\, i\in B, j\in V, v\in\Theta(i), k\in\Upsilon \tag{7.16}$$

$$0 \le w_{ivjk} \le 1, \qquad \forall\, i\in B, j\in V, v\in\Theta(i), k\in\Upsilon \tag{7.17}$$

$$\text{Constraints (7.4) to (7.10), (7.12)} \tag{7.18}$$

In the sequel, it would prove that the MIP formulation $[\mathcal{P}_2]$ is equivalent to $[\mathcal{P}_1]$ in the sense that there exists an optimal solution at which both problems achieve the same objective value. Note that due to the relaxation of Constraints (7.16) with regarding to Constraints (7.13) and the observation that the coefficient of w_{ivjk}, i.e., $(n-k+1)\cdot p_{vj}$, is positive appeared in the objective function, by minimization, someone might think that the equivalence between $[\mathcal{P}_1]$ and $[\mathcal{P}_2]$ is trivial. However, according to Constraints (7.12), the decrease of w_{ivjk} will bring about the increase of at least one y_{ik}, $i\in B$, $k\in\Upsilon$ and meanwhile the coefficient of y_{ik} is also positive. Therefore, decrease of w_{ivjk} can not guarantee the decreasing of the objective value and without in-depth examination, it is not clear to state that $[\mathcal{P}_1]$ and $[\mathcal{P}_2]$ are equivalent.

Lemma 7.2.1. *Formulations* $[\mathcal{P}_1]$ *and* $[\mathcal{P}_2]$ *are equivalent.*

Proof. For the sake of conciseness, let $\mathbf{x} = (x_{ijk})_{i\in B, j\in V, k\in\Upsilon}$, $\mathbf{y} = (y_{ik})_{i\in B, k\in\Upsilon}$, $\mathbf{z} = (z_{iv})_{i\in B, v\in\Theta(i)}$,

$\mathbf{w} = (w_{ivjk})_{i\in B, j\in V, v\in\Theta(i), k\in\Upsilon}$ and $\boldsymbol{\alpha} = (\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w})$. Denote the sets of feasible solution for both $[\mathcal{P}_1]$ and $[\mathcal{P}_2]$, P_1 and P_2, respectively. Obviously, $\forall\, \boldsymbol{\alpha} \in P_1$, $\boldsymbol{\alpha}$ is still an element for set P_2. Therefore, $P_1 \subseteq P_2$. Additionally, since $[\mathcal{P}_1]$ and $[\mathcal{P}_2]$ have the same objective function f, it can be concluded that $f(\boldsymbol{\alpha}_1^*) \geq f(\boldsymbol{\alpha}_2^*)$, where $\boldsymbol{\alpha}_1^*$ and $\boldsymbol{\alpha}_2^*$ are the optimal solutions of $[\mathcal{P}_1]$ and $[\mathcal{P}_2]$, respectively.

For an optimal solution of $[\mathcal{P}_2]$, $\boldsymbol{\alpha}_2^* = (\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*, \mathbf{w}^*)$, for all triples of $(x_{ijk}^*, z_{iv}^*, w_{ivjk}^*)$, if they fulfill $x_{ijk}^* \cdot z_{iv}^* = 0$, $w_{ivjk}^* = 0$ or $x_{ijk}^* \cdot z_{iv}^* = 1$, $w_{ivjk}^* = 1$, then it can be checked that $\boldsymbol{\alpha}_2^*$ is also a feasible solution of $[\mathcal{P}_1]$, i.e., $\boldsymbol{\alpha}_2^* \in P_1$. Since $f(\boldsymbol{\alpha}_2^*) \leq f(\boldsymbol{\alpha}_1^*)$, $\boldsymbol{\alpha}_2^*$ is an optimal solution for $[\mathcal{P}_1]$.

If there exists a triple $(x_{i_0j_0k_0}^*, z_{i_0v_0}^*, w_{i_0v_0j_0k_0}^*)$, $k_0 < |\Upsilon|$, such that $x_{i_0j_0k_0}^* \cdot z_{i_0v_0}^* = 0$, $0 < w_{i_0v_0j_0k_0}^* \leq 1$, it can be proved that $\boldsymbol{\alpha}_2 = (\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w})$ is also an optimal solution for $[\mathcal{P}_2]$, where $\boldsymbol{\alpha}_2$ is identical to $\boldsymbol{\alpha}_2^*$, except for the elements $y_{i_0k_0}$ and $w_{i_0v_0j_0k_0}$, with $y_{i_0k_0} = y_{i_0k_0}^* + p_{v_0j_0} \cdot w_{i_0v_0j_0k_0}^*$ and $w_{i_0v_0j_0k_0} = 0$. Firstly, check the feasibility of $\boldsymbol{\alpha}_2$. Because the value of $\boldsymbol{\alpha}_2$ is identical to $\boldsymbol{\alpha}_2^*$ except for the two particular elements, $\boldsymbol{\alpha}_2$ should immediately fulfill the Constraints (7.16-7.18) except for Constraints (7.9, 7.12, 7.16, 7.17). $y_{i_0k_0} = y_{i_0k_0}^* + p_{v_0j_0} \cdot w_{i_0v_0j_0k_0}^* > y_{i_0k_0}^* \geq 0$ and $0 \leq w_{i_0v_0j_0k_0} \leq 1$, therefore, $\boldsymbol{\alpha}_2$ satisfies Constraints (7.9) and (7.17). Furthermore, $x_{i_0j_0k_0} \cdot z_{i_0v_0} = 0$, so $w_{i_0v_0j_0k_0} = 0 \geq z_{i_0v_0} + x_{i_0j_0k_0} - 1$ (i.e., constriants (7.16)). For Constraints (7.12) , since $\boldsymbol{\alpha}_2^*$ is a feasible solution of $[\mathcal{P}_1]$, $\sum_{j\in V(i_0)}(a_j - s_{i_0})x_{i_0jk_0}^* - \sum_{l\in\Upsilon(k_0)}(y_{i_0l}^* + \sum_{j\in V} t_{i_0j} \cdot x_{i_0jl}^*) - \sum_{l\in\Upsilon(k_0)}\sum_{j\in V}\sum_{v\in\Theta(i_0)} p_{vj} \cdot w_{i_0vjl}^* \leq y_{i_0k_0}^*$. For the case of $\boldsymbol{\alpha}_2$, $\sum_{j\in V(i_0)}(a_j - s_{i_0})x_{i_0jk_0} - \sum_{l\in\Upsilon(k_0)}(y_{i_0l} + \sum_{j\in V} t_{i_0j} \cdot x_{i_0jl}) - \sum_{l\in\Upsilon(k_0)}\sum_{j\in V}\sum_{v\in\Theta(i_0)} p_{vj} \cdot w_{i_0vjl} = \sum_{j\in V(i_0)}(a_j - s_{i_0})x_{i_0jk_0}^* - \sum_{l\in\Upsilon(k_0)}(y_{i_0l}^* + \sum_{j\in V} t_{i_0j} \cdot x_{i_0jl}^*) - \sum_{l\in\Upsilon(k_0)}\sum_{j\in V}\sum_{v\in\Theta(i_0)} p_{vj} \cdot w_{i_0vjl}^* \leq y_{i_0k_0}^* < y_{i_0k_0}$, therefore, $\boldsymbol{\alpha}_2$ also fulfills Constraints (7.12). Hence, $\boldsymbol{\alpha}_2 \in P_2$. Note that $f(\boldsymbol{\alpha}_2) = f(\boldsymbol{\alpha}_2^*)$, so $\boldsymbol{\alpha}_2$ is also an optimal solution for $[\mathcal{P}_2]$. Besides, it can be observed that $\boldsymbol{\alpha}_2 \in P_1$, therefore, $f(\boldsymbol{\alpha}_1^*) = f(\boldsymbol{\alpha}_2) = f(\boldsymbol{\alpha}_2^*)$.

If there exists a triple $(x_{i_0j_0k_0}^*, z_{i_0v_0}^*, w_{i_0v_0j_0k_0}^*)$, $k_0 = |\Upsilon|$, such that $x_{i_0j_0k_0}^* \cdot z_{i_0v_0}^* = 0$, $0 < w_{i_0v_0j_0k_0}^* \leq 1$, it can construct a $\boldsymbol{\alpha}_2$ which is identical to $\boldsymbol{\alpha}_2^*$ except for element $w_{i_0v_0j_0k_0}$, $w_{i_0v_0j_0k_0} = 0$. It is easy to check that $\boldsymbol{\alpha}_2$ is a feasible solution for $[\mathcal{P}_2]$. However, $f(\boldsymbol{\alpha}_2) < f(\boldsymbol{\alpha}_2^*)$, which contradict the assumption that $\boldsymbol{\alpha}_2^*$ is an optimal solution for $[\mathcal{P}_2]$. Therefore, no such triple $(x_{i_0j_0k_0}^*, z_{i_0v_0}^*, w_{i_0v_0j_0k_0}^*)$, $k_0 = |\Upsilon|$ can exist.

In sum, it has been proved that if both $[\mathcal{P}_1]$ and $[\mathcal{P}_2]$ are feasible, there exist a feasible solution $\boldsymbol{\alpha}^*$ ($\boldsymbol{\alpha}^* \in P_1, P_2$), at which $[\mathcal{P}_1]$ and $[\mathcal{P}_2]$ achieve the same optimal value. Thus, formulations $[\mathcal{P}_1]$ and $[\mathcal{P}_2]$ are equivalent. □

7.2.4 Strengthening the formulation of $[\mathcal{P}_2]$

Although $[\mathcal{P}_1]$ and $[\mathcal{P}_2]$ are equivalent, $[\mathcal{P}_1]$ is a stronger formulation than $[\mathcal{P}_2]$ since $P_1 \subseteq P_2$. Actually, there are three kinds of cuts that can be used to tighten the formulation of $[\mathcal{P}_2]$. The first two are the valid inequalities for $[\mathcal{P}_1]$. In $[\mathcal{P}_1]$, according to Constraints (7.13), $w_{ivjk} = z_{iv} \cdot x_{ijk}$. Hence, $\sum_{j\in V} w_{ivjk} = \sum_{j\in V} z_{iv} \cdot x_{ijk} = z_{iv} \cdot \sum_{j\in V} x_{ijk} \leq z_{iv}$. The last inequality is valid because of Constraints (7.5). Analogously, based on Constraints (7.6) and (7.13), it follows that $\sum_{v\in\Theta(i)} w_{ivjk} = \sum_{v\in\Theta(i)} z_{iv} \cdot x_{ijk} = x_{ijk} \cdot \sum_{v\in\Theta(i)} z_{iv} = x_{ijk}$. Therefore, the following two sorts of constraints can be added to $[\mathcal{P}_2]$ for strengthening.

$$\sum_{j\in V} w_{ivjk} \leq z_{iv}, \qquad \forall\, i \in B, v \in \Theta(i), k \in \Upsilon \tag{7.19}$$

$$\sum_{v\in\Theta(i)} w_{ivjk} = x_{ijk}, \qquad \forall\, i \in B, j \in V, k \in \Upsilon \tag{7.20}$$

The rest of cuts to strengthen the formulation of $[\mathcal{P}_2]$ can be derived from Lemma 1 in Monaco and Sammarra (2007). Let n^i be the total number of vessels assigned to Berth i. Monaco and Sammarra prove that for the discrete BAP, at optimality, n^i vessels assigned to a specific Berth i are scheduled to be served consecutively as the $(n - n^i + 1)^{\text{th}}$ to the n^{th} vessels. For instance, in Figure 7.1, according to the Lemma 1 in Monaco and Sammarra (2007), it can be concluded that the shown berth allocation plan can not be an optimal one since the vessels assigned to Berth m are not served consecutively as stated by the lemma. Thus, in light of the lemma, the berth allocation plan depicted in Figure 7.1 can be ignored without the loss of optimality for the original problem.

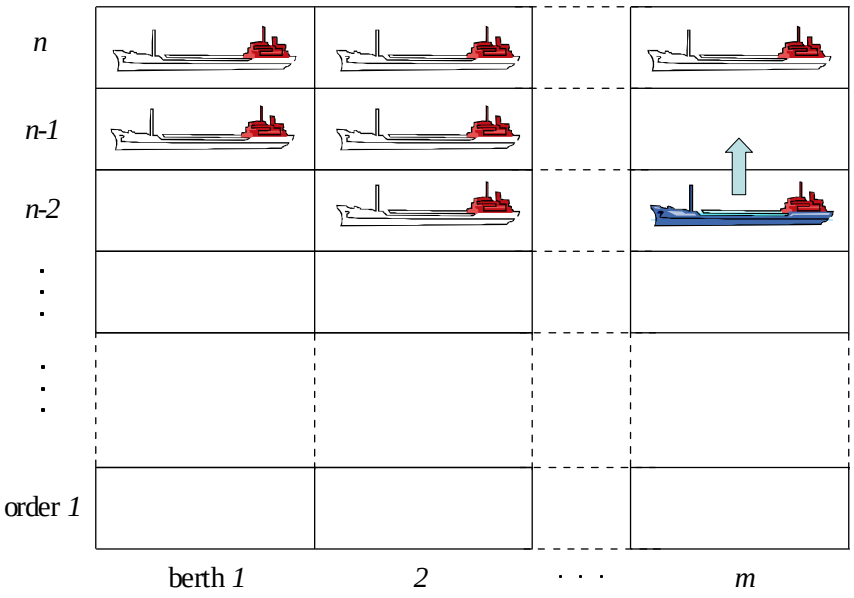


Figure 7.1: An illustration for the Lemma 1 in Monaco and Sammarra (2007)

The Lemma 1 in Monaco and Sammarra (2007) can be translated to mathematical language as follows: $\forall i \in B, l \in \Upsilon$, if $l < n - \sum_{j \in V} \sum_{k \in \Upsilon} x_{ijk} + 1$, then $\sum_{j \in V} x_{ijl} = 0$, where $\sum_{j \in V} \sum_{k \in \Upsilon} x_{ijk}$ is equal to the total number of vessels assigned to Berth i and $\sum_{j \in V} x_{ijl}$ is the indicator for whether the l^{th} order of Berth i is occupied by a vessel or not. For example, in Figure 7.1, there are 3 vessels assigned to berth 2 (i.e., $\sum_{j \in V} \sum_{k \in \Upsilon} x_{ijk} = 3$). Therefore, for order $l\,(1 \leq l < n-2)$, there should be no vessel (i.e., $\sum_{j \in V} x_{ijl} = 0$). The relationship between $\sum_{j \in V} \sum_{k \in \Upsilon} x_{ijk}$ and $\sum_{j \in V} x_{ijl}$ for the lemma is equivalent to the following cuts.

$$n - \sum_{j \in V} \sum_{k \in \Upsilon} x_{ijk} + 1 - l \leq n \cdot (1 - \sum_{j \in V} x_{ijl}), \qquad \forall i \in B, l \in \Upsilon$$

Or more simplified,

$$n \cdot \sum_{j \in V} x_{ijl} \leq \sum_{j \in V} \sum_{k \in \Upsilon} x_{ijk} - 1 + l, \qquad \forall i \in B, l \in \Upsilon \tag{7.21}$$

7.3 Heuristic

This section aims to develop a heuristic for the proposed problem. It is worth noting that for the discrete BAP, it can be regarded as an unrelated parallel machines scheduling problem by associating berths to parallel machines and vessels to jobs. Specifically, a discrete BAP actually can be considered as a machine scheduling problem, $(Rm \,|\, r_j \,|\, \sum c_j)$ in the standard three-field scheduling notation. Therefore, if the information of the berth position allocated for each vessel is regarded as a priori knowledge but the decision of the berthing time for each vessel still needs to be made, a discrete BAP can be treated as m single machine scheduling problems, i.e., $(1 \,|\, r_j \,|\, \sum c_j)$. However, even for $(1 \,|\, r_j \,|\, \sum c_j)$ problem, it has been proved to be $\mathcal{NP}$-hard (**?**). In contrast with $(1 \,|\, r_j \,|\, \sum c_j)$ problem, there exists a polynomial algorithm to solve $(1 \,|\, r_j \,|\, c_{\max})$ problem to optimality: sort all the jobs in non-decreasing order of their release dates and assign all the jobs to the machine in accordance with the job sequence. This optimal rule lays the foundation for constructing the heuristic for the proposed problem. Its rationale is to add cuts to the formulation $[\mathcal{P}_2]$ with the objective to generate a berth allocation plan such that: for each individual berth, all the vessels assigned to the berth are served by the rule of first-come-first-serve.

To achieve this, the following constraints can be added to $[\mathcal{P}_2]$. For all $i \in B, k \in \Upsilon\,(k \neq n)$ and $j_1, j_2 \in V$ while $(a_{j_2} > a_{j_1})$:

$$\sum_{l \in \Gamma(k)} x_{ij_2l} - x_{ij_1k} \geq \sum_{l \in \Upsilon} (x_{ij_1l} + x_{ij_2l}) - 2 \tag{7.22}$$

Here $\Gamma(k) = \{k' \in \Upsilon \,|\, k' > k\}$, $\forall\, k \in \Upsilon$, is the set of position indexes greater than k. For Vessels j_1 and j_2, if they are allocated to the same berth, say Berth i, then $\sum_{l \in \Upsilon}(x_{ij_1l} + x_{ij_2l}) = 2$ and by Constraints (7.22), $\sum_{l \in \Gamma(k)} x_{ij_2l} \geq x_{ij_1k}$ which forces the berthing sequence of Vessel j_1 to precede Vessel j_2. Alternatively, if Vessels j_1 and j_2 are not allocated to the same berth, then Constraints (7.22) become redundant. By adding Constraints (7.19) to (7.22) to the formulation

$[\mathcal{P}_2]$, a much easier problem $[\mathcal{P}_3]$ can be obtained.

$$
\begin{aligned}
[\mathcal{P}_3] \min \quad & \sum_{i\in B}\sum_{j\in V}\sum_{k\in\Upsilon}\sum_{v\in\Theta(i)}(n-k+1)p_{vj}\cdot w_{ivjk} \\
& + \sum_{i\in B}\sum_{j\in V}\sum_{k\in\Upsilon}\{(n-k+1)t_{ij}+s_i-a_j\}x_{ijk} \\
& + \sum_{i\in B}\sum_{k\in\Upsilon}(n-k+1)y_{ik} \\
s.t. \quad & \\
& \text{Constraints (7.16) to (7.22)} \qquad (7.23)
\end{aligned}
$$

Clearly, the optimal solution of $[\mathcal{P}_3]$ can not guarantee to be the optimal solution for the problem $[\mathcal{P}_2]$ as well. To seek better solutions, Tabu search can be adopted for improvement. The coding of a solution for the proposed Tabu search includes two parts: the first $(n+m-1)$ cells represent the berth allocation plan and the following $(q+m-1)$ cells represent the quay crane assignment. Figure 7.2 shows a feasible solution for the proposed quayside operation problem with 2 berths, 4 vessels, and 4 quay cranes (i.e., $m=2, n=4, q=4$).

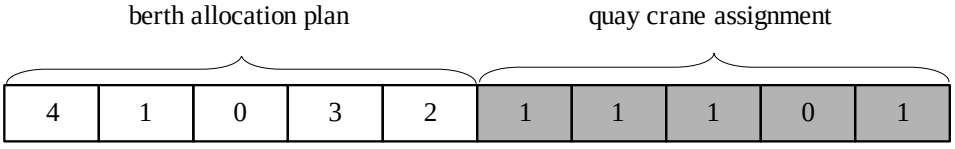


Figure 7.2: An illustration of solution coding for the proposed Tabu search

For each part of the coding string, $(m-1)$ zeros divide the corresponding part into m substrings. For the part representing the berth allocation plan, each substring contains the vessel berthing information for a particular berth. For example, in Figure 7.2, there are two substrings, i.e., '41' and '32', which means that Vessels 4 and 1 are berthed sequentially at Berth 1 and Vessels 3 and 2 are allocated to Berth 2 one after another. For the part that represents the quay crane assignment, it is a binary string with '0' and '1' only. For the i^{th} substring of this part, if the number of element '1's is v, then it means that v quay cranes is assigned to Berth i. In Figure 7.2, the second part of the string implies that Cranes 1 to 3 are assigned to Berth 1 while Crane 4 is assigned to Berth 2.

To generate the neighborhood solutions, the pairwise interchange operation is applied to either part of the current solution with equal probability. If the first part is chosen for operation, two different cells are randomly selected and the values in both cells are exchanged (see Figure 7.3). Otherwise, if the second part is chosen, first of all, denote the set of location indexes for Element '0' as Λ_0 and for the case of Element '1' as Λ_1.

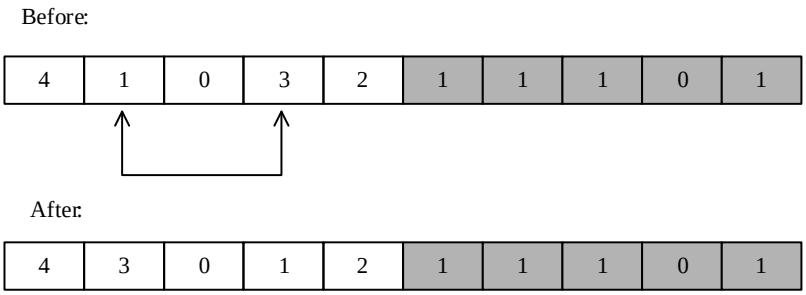


Figure 7.3: Neighborhood solution generation method for the first part

Next, as depicted in Figure 7.4, randomly select two elements from sets Λ_0 and Λ_1 respectively and assign them to l_1 and l_2. Exchange the values in Cells l_1 and l_2. As an illustration, in Figure 7.4, after the pairwise interchange operation, the second part of the new solution stands for that Crane 1 is assigned to Berth 1 and Cranes 2 to 4 is assigned to Berth 2.

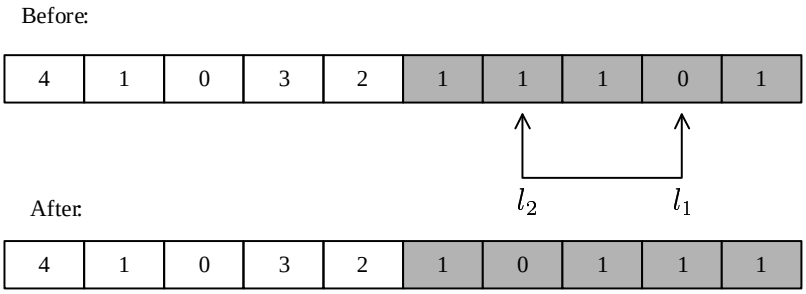


Figure 7.4: Neighborhood solution generation method for the second part

The evaluation of the solution string is straightforward. Both substrings are read from left to right and the the information of berth allocation plan and quay crane assignment for each berth is extracted from the string. Assume that the berth allocation plan for Berth i is $(u_1, u_2, \cdots, u_{n^i})$. Since the number of quay cranes assigned to the berth has been determined, the handling time for each vessel can be calculated. Berth i serves the n^i vessels in the sequence

of $(u_1, u_2, \cdots, u_{n^i})$. For any two consecutive Vessels u_j and u_{j+1}, if the completion time of Vessel u_j, c_{u_j}, is later than the arrival time of Vessel u_{j+1}, $a_{u_{j+1}}$, then the service starting time for Vessel u_{j+1}, $t_{u_{j+1}}$, is equal to c_{u_j}. Otherwise, let $t_{u_{j+1}} = a_{u_{j+1}}$. The evaluation is terminated after the berthing times for all the vessels have been determined.

Compared with traditional Tabu search algorithm, in this research, the hashing technique from computer science is adopted to store the tabu information. For more detail on how to embed the hashing technique into the Tabu search framework, the authors refer to the work in Srivastava (1998).

The following summarizes the procedure of the proposed heuristic designed for the integration problem on quayside operation.

Step 1: Solve the problem $[\mathcal{P}_3]$ and code the optimal solution as a string, which is the initial solution, $\boldsymbol{s}_0$, of the Tabu search. Initialize the current solution $\boldsymbol{s} = \boldsymbol{s}_0$, the set of hashed tabu moves, $H = \{\boldsymbol{s}_0\}$, and the maximum iteration number $\iota_{\max}$. Let the iteration counter $\iota = 1$.

Step 2: Generate the neighborhood solutions of $\boldsymbol{s}$, $N(\boldsymbol{s})$, by the pairwise interchange operation. Find out the admissible moves for the current solution $\boldsymbol{s}$, $A(\boldsymbol{s}) = \{\boldsymbol{s}' \mid \boldsymbol{s}' \in N(\boldsymbol{s}) \wedge \boldsymbol{s}' \notin H\}$.

Step 3: Evaluate all the solutions in $A(\boldsymbol{s})$ and select the one with the minimal total turnaround time. Denote the chosen solution $\boldsymbol{s}^*$, $\boldsymbol{s}^* \in A(\boldsymbol{s})$.

Step 4: If $\iota > \iota_{\max}$, stop the Tabu search. Otherwise, let $\boldsymbol{s} = \boldsymbol{s}^*$, $\iota = \iota + 1$ and go to Step 2.

7.4 Local Branching (LB) Method

In the previous section, a heuristic is developed to obtain an upper bound for the proposed problem. In this section, the LB Method is applied to solve the integrated model for the quayside operation problem aiming to get the exact optimal solution. The LB Method is developed in **?** for solving the MIP problems. Its philosophy is similar to the local search meta-heuristics, but the neighborhoods are obtained through the introduction in the MIP model of linear inequalities known as LB Cuts and solved by a general-purpose MIP solver.

Under the conventional local branching scheme, the k-OPT neighborhood boundary cuts are used for local branching. Given the searching space P for the MIP problem, an incumbent solution $\overline{x}$, $\overline{x} \in P$ and a positive number k, then the k-OPT neighborhood around the incumbent solution $\overline{x}$ is:

$$\Delta(x, \overline{x}) \leq k$$

Here $\Delta(x, \overline{x})$ is defined as $\sum_{j \in \overline{J}}(1 - x_j) + \sum_{j \in J \setminus \overline{J}} x_j$ and the set J is the index set for x and the set $\overline{J} := \{j \in J \,|\, \overline{x}_j = 1\}$. Therefore, by adding the cut representing the boundary of the k-OPT neighborhood, the whole searching space P can be partitioned into two branches: $\Delta(x, \overline{x}) \leq k$ (left branch) and $\Delta(x, \overline{x}) \geq k+1$ (right branch). Actually, for the proposed integrated quayside operation problem, there is another problem-oriented branching scheme. As illustrated in Figure 7.1, intuitively, the searching space of the proposed problem can be separated into two mutually exclusive subspaces: the one where there is at least one vessel served at the Order 1 of some berth (left branch); the other where there is no vessel served at the Order 1 of some berth (right branch). Note that for each right branch, a similar partition scheme can be adopted for further branching. More specifically, if the current right branch is that there is no vessel served at the Order k (also for the Order l, $1 \leq l < k$) of a particular berth, then the right branch can be divided into two sub-branches: there is at least one vessel served at the Order $k+1$ of a particular berth (but no vessel served at the Order l, $1 \leq l \leq k$) and there is no vessel served at the Order $k+1$ (also for the Order l, $1 \leq l \leq k$). Since, Constraints (7.21) enforce that vessels assigned to a particular berth should be served in a consecutive manner, the neighborhood boundary cuts for the left and right branches at level k are Constraints (7.24) and (7.25), respectively.

$$\sum_{i \in B} \sum_{j \in V} x_{ijk} \geq 1 \tag{7.24}$$

$$\sum_{i \in B} \sum_{j \in V} x_{ijk} = 0 \tag{7.25}$$

Figure 7.5 shows the local branching scheme for the proposed problem. There are $(k+1)$ levels (numbered from 0 to k) under the scheme, where $k = n - \lceil \frac{n}{m} \rceil$ and the function $\lceil x \rceil = \max\{z \in \mathbb{Z} \,|\, z \geq x\}$. The maximum number of levels is bounded by $n - \lceil \frac{n}{m} \rceil$ since for further

branching after the level, the right branch will become infeasible. In Level 0, it contains only one node, Node 0, which represents the original problem $[\mathcal{P}_2]$. In Level l, $1 \leq l \leq k$, there are two nodes: Node l and Node l', which stand for the left branch and right branch for the current level, respectively. From Figure 7.5, it is clear that the original problem is branched into $(n+1-\lceil \frac{n}{m} \rceil)$ mutually exclusive but collectively exhaustive sub-problems and they are Nodes l, $1 \leq l \leq k$ and Node k'.

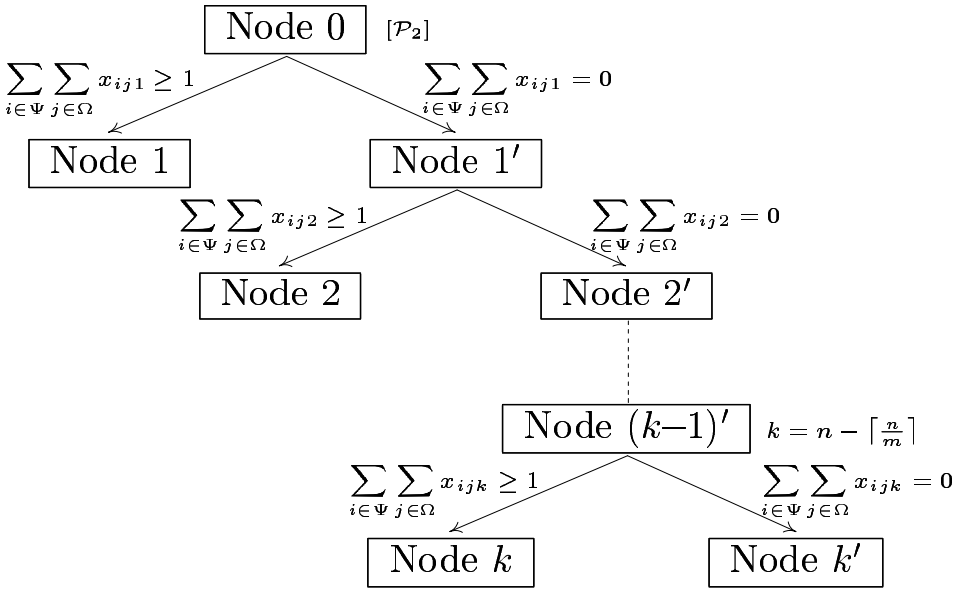


Figure 7.5: The local branching scheme for the proposed problem

Also different from the conventional LB Method, in this research, since an upper bound for the proposed problem can be obtained through the heuristic developed in the previous section, the bounding procedure can be added into the framework to prune dominated branches if at certain point of time, their lower bounds excel the upper bound.

The following summarizes the procedure of the proposed local branching method for the studied problem. For conciseness, denote Node k' in Figure 7.5 as Node $(k+1)$.

Step 1: Let U be an upper bound for the proposed problem. Initially, $U = +\infty$. Let the iteration counter $\iota = 0$.

Step 2: If $\iota = 0$, go to Step 3; If $1 \leq \iota \leq n+1-\lceil \frac{n}{m} \rceil$, go to Step 4; Otherwise, stop the local branching procedure.

Step 3: Solve the problem $[\mathcal{P}_2]$ by the heuristic developed in the previous section. Update the upper bound U to be the output of the heuristic. Let $\iota = \iota + 1$ and go to Step 2.

Step 4: Construct Node ι by adding the corresponding LB Cuts into the formulation $[\mathcal{P}_2]$ shown in Figure 7.5. Solve the branch by the MIP solver, CPLEX. If at certain point of time, the lower bound for Node ι is greater than U, stop solving the node; Otherwise, Node ι is solved to optimality and update the upper bound U as the optimal value of Node ι. Let $\iota = \iota + 1$ and go to Step 2.

7.5 Computational Experiments

The aim of this section is to analyze the performance of the proposed heuristic and LB Method for the integrated quayside operation problem. There are two sets of instances used for evaluation.

- Set A: the number of vessels $n = 20$, the number of berths m ranges from 5 to 12, and the number of quay cranes $q = 3m$.
- Set B: the number of vessels $n = 30$, the number of berths m ranges from 5 to 6, and the number of quay cranes $q = 3m$.

To construct an instance, first of all, let the minimum and the maximum numbers of quay cranes that can be deployed to any berth be 1 and 5 (i.e., $\underline{c}^i = 1$ and $\overline{c}^i = 5$, $\forall\, i \in B$). Then let the available time for Berth i, s_i, follows the uniform distribution in the range $[0, 100]$. Consequently, generate the information of the n vessels randomly following the procedures below: the expected arrival time and length of the workload list for any vessel (say, Vessel j) follow the discrete uniform distribution within the intervals $[0, 2000]$ and $[16, 25]$, respectively; given the length of the workload list for the vessel, the size of the individual task in each bay of the vessel ranges from 30 to 50; by inputting the workload list of Vessel j to one of the heuristics to solve quay crane scheduling like the ones introduced in Chapter 4, the processing time p_{vj} $(1 \leq v \leq 5)$ can be initialized one by one; finally, the extra transportation if Vessel j is moored at Berth i is randomly generated in the range $[0, 100]$.

For each combination of n and m in the instance sets, 5 random copies of instances are generated and they are denoted as m_n_i, $1 \leq i \leq 5$. In other words, there are $8 \times 1 \times 5 = 40$ instances in set A and $2 \times 1 \times 5 = 10$ instances in set B.

The proposed heuristic and LB Method are coded in C++ and used the ILOG Concert Technology (version 12.1). For the heuristic, the maximum iteration number $\iota_{\max}$ is set to 100 and the size of the neighborhood solutions for each iteration is 100. As a benchmark, the B&C algorithm in CPLEX is used for comparison. The Tables 7.1 and 7.2 summarize the computational results for the 40 instance in Set A, which are tested in a PC with 2.40 GHz CPU and 3 GB RAM. In these two tables, Columns (1) and (2) store the optimal objective value and running time for each instance solved by CPLEX; Column (3) is the list of computational times consumed by the LB Method; Columns (4) and (5) are the computational results for the proposed heuristic; The gaps in Columns (6) and (7) are used to measure the efficiency of the LB Method and the effectiveness of the proposed heuristic, respectively. Here,

$$\text{Gap}_1 = \frac{\text{value in Column (3)}}{\text{value in Column (2)}} \times 100$$

$$\text{Gap}_2 = \frac{\text{value in Column (4)} - \text{value in Column (1)}}{\text{value in Column (1)}} \times 100$$

Over the 40 instances, the LB Method outperforms the B&C in CPLEX except for the Instances 11_20_4 and 12_20_5. The average computational time saving is up to 51%. On the other hand, the performance of the heuristic is also remarkable. With only 1.39% of average gap, the heuristic is considerably faster than the other two exact algorithms.

As pointed out by **?**, one of the merits of the LB Method is its capability to be straightforwardly implemented in a parallel computing architecture. The 10 instances in Set B are adopted to evaluate the performance of the parallel version of the proposed LB Method. Two computers are used for the parallel computing and they are connected through a Message Passing Interface written by C++. The first PC with 3 GHz CPU and 4 GB RAM is chosen as the master computer and the second PC with 2.40 GHz CPU and 3 GB RAM is selected as the slave computer. For comparison, the B&C algorithm of CPLEX, the LB Method for single

computer, and the heuristic are also run on the master computer individually for all the 10 instances. Note that the time limit for the 4 algorithms is set to 3 hours (i.e., 10800 seconds). Table 7.3 is the computational results for the test. Columns (1) and (2) are the computational results for the B&C. For Instances 6_30_3, 6_30_4, and 6_30_5, the B&C fails to generate the optimal solution within the time limit. Therefore, the values with a star symbol are the incumbent solutions for these 3 instances. Columns (3) to (5) keep the optimal value for each instance, the computational time for the single computer version of the LB Method, and the computational time of the two-computer version of the LB Method, respectively. Both two versions of the LB method outperform the B&C algorithm in terms of the computational time. The average running times for the two versions of the LB Method are 2184.3 seconds and 1226.3 seconds. Hence, roughly speaking, adding one computer can double the efficiency of the algorithm. Columns (6) and (7) store the computational results for the heuristic. The Gap_3 in column (8) is defined as:

$$\text{Gap}_3 = \frac{\text{value in Column (6)} - \text{value in Column (3)}}{\text{value in Column (3)}} \times 100$$

Note that average Gap_3 for the 10 instances is 6.11%. Based on the computational times in Column (7) and the gaps in Column (8) for the heuristic, it is clear that the heuristic, which achieves quite a nice balance between effectiveness and efficiency, is a good alternative algorithm to solve the proposed integrated quayside operation problem as well.

Table 7.1: The computational results for the instance set A

	CPLEX		Local Branching	Heuristic		Gap	
Instance	Obj	Time(s)	Time(s)	Obj	Time(s)	Gap_1(%)	Gap_2(%)
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
5_20_1	20291	940	315	20644	39	33.51	1.74
5_20_2	19580	212	181	19580	29	85.38	0.00
5_20_3	18826	320	218	19120	33	68.13	1.56
5_20_4	15731	1103	401	16140	40	36.36	2.60
5_20_5	22975	231	205	22976	25	88.74	0.00
6_20_1	14490	812	424	14490	67	52.22	0.00
6_20_2	14656	1621	468	14656	63	28.87	0.00
6_20_3	16193	570	333	16370	65	58.42	1.09
6_20_4	15457	3002	1000	15457	119	33.31	0.00
6_20_5	14798	2837	764	14798	103	26.93	0.00
7_20_1	13599	2165	595	13659	48	27.48	0.44
7_20_2	14266	2212	565	14406	51	25.54	0.98
7_20_3	14556	1593	736	14921	57	46.20	2.51
7_20_4	10354	515	456	10382	11	88.54	0.27
7_20_5	12400	3537	963	12568	50	27.23	1.35
8_20_1	10620	4326	1120	10685	73	25.89	0.61
8_20_2	11541	5717	1396	11645	156	24.42	0.90
8_20_3	8282	3715	1743	8787	46	46.92	6.10
8_20_4	12898	5654	605	12898	115	10.70	0.00
8_20_5	11469	5588	2017	11469	116	36.10	0.00

Table 7.2: Continuation for Table 7.1

	CPLEX		Local Branching	Heuristic		Gap	
Instance	Obj	Time(s)	Time(s)	Obj	Time(s)	Gap_1(%)	Gap_2(%)
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
9_20_1	9100	823	545	9100	39	66.22	0.00
9_20_2	10252	1510	503	10252	34	33.31	0.00
9_20_3	10620	4894	1068	10675	155	21.82	0.52
9_20_4	10299	766	524	10299	34	68.41	0.00
9_20_5	10020	2476	848	10350	99	34.25	3.29
10_20_1	9303	5538	1374	9428	18	24.81	1.34
10_20_2	8739	3528	1418	8899	23	40.19	1.83
10_20_3	8294	1815	740	8473	12	40.77	2.16
10_20_4	7566	9066	3182	7610	17	35.10	0.58
10_20_5	8376	8013	3752	8929	26	46.82	6.60
11_20_1	8645	2977	1221	8725	30	41.01	0.93
11_20_2	7939	2258	793	8077	19	35.12	1.74
11_20_3	9817	15394	5438	10111	46	35.33	2.99
11_20_4	8624	2438	2996	8808	23	122.89	2.13
11_20_5	8062	2625	1972	8510	26	75.12	5.56
12_20_1	7732	3827	1716	7806	22	44.84	0.96
12_20_2	9068	15791	1805	9169	24	11.43	1.11
12_20_3	7236	3032	2539	7354	16	83.74	1.63
12_20_4	7618	3165	1858	7710	23	58.70	1.21
12_20_5	7111	2594	3629	7184	21	139.90	1.03
Average						48.27	1.39

Table 7.3: The computational results for the instance set B

	CPLEX		Local Branching			Heuristic		
Instance	Obj	Time(s)	Obj	Time1(s)	Time2(s)	Obj	Time(s)	Gap_3(%)
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
5_30_1	34169	5977	34169	1542	859	35966	22	5.26
5_30_2	31679	1183	31679	1113	521	35243	6	11.25
5_30_3	40047	6356	40047	1177	636	40732	7	1.71
5_30_4	33232	4786	33232	1064	512	35613	7	7.16
5_30_5	36357	6442	36357	1276	660	38123	6	4.86
6_30_1	35466	9951	35466	2211	1187	38968	6	9.87
6_30_2	33624	10470	33624	2241	1331	35765	4	6.37
6_30_3	32499*	10800	32417	3181	2044	34207	10	5.52
6_30_4	28978*	10800	28740	3477	2458	29423	20	2.38
6_30_5	25624*	10800	25582	4561	2055	27308	5	6.75
Average				2184.3	1226.3			6.11

7.6 Summary

A new integrated model for the quayside operation problem, one of the most critical logistics problems arising in the management of port container terminals, is provided in this Chapter. The problem is formulated as an MIP and its properties are analyzed. For solution, a heuristic and a LB Method are developed and their performance is evaluated through a comprehensive numerical experiment.

Chapter 8

An Integrated Model for the Quayside Operation Problem with Continuous Berths

8.1 Background

Based on the literature review, it can be observed that despite the aforementioned deficiency, the idea proposed in Liu et al. (2006) is significant since it is capable to veritably embrace all the information of the quayside operation under one umbrella. In this chapter, the Berth-level model in Liu et al. (2006) will be extended by overcoming the deficiency mentioned in Chapter 2.

8.2 Problem Description and Mathematical Formulation

Let the length of the quay space be L and there are m quay cranes (denote the set of quay cranes, Q and number them increasingly according to their relative positions) deployed along the quay. Let V denote the set of container vessels with known characteristics such as the length, the expected arrival time, and the workload. For each incoming vessel, it is allowed to moor at anywhere along the quay so as to sufficiently utilize the quay resource. In other

words, the BAP under the proposed framework is the so called continuous BAP appeared in the literature. Besides, for Vessel $i \in V$, let $\underline{q}^i$ and $\overline{q}^i$ be the minimum and maximum number of quay cranes that can be assigned to Vessel i. The rationale to set the range of quay crane number for each vessel is explained in Liu et al. (2006).

As mentioned in the review of Liu et al. (2006), the author introduced the parameter p_{iq}, $i \in V$, $\underline{q}^i \leq q \leq \overline{q}^i$, the processing time when q quay cranes are assigned to process Vessel i, to avoid the solving of the QCSP within the integrated framework. However, such a strategy comes with the price that in order to obtain the series of parameters p_{iq}, the QCSP has to be solved $\sum_{i \in V}(\overline{q}^i - \underline{q}^i + 1)$ times beforehand. Although the QCSP has been proven to be $\mathcal{NP}$-complete (Lee et al., 2008), the computational results from Liu et al. (2006) showed that for real application (i.e., Hong Kong terminals), the off-the-shelf optimization solver is capable to solve the QCSP in seconds for a given i and q. Besides, in the literature, several heuristics or approximation algorithms have been developed to provide satisfactory approximate solutions for the QCSP, e.g., Bierwirth and Meisel (2009). Therefore, the sacrifice to parameterize p_{iq} can be reduced to the minimum.

The proposed problem for the quayside operation can be stated as followings: given the aforementioned parameters, the decision makers need to decide how many and which quay cranes will be assigned to a particular vessel and meanwhile the docking position and service start time for each incoming vessel, aiming to minimize the total weighted flow time. Actually, the proposed problem can be viewed as the combination of the ones presented in Guan and Cheung (2004) and Liu et al. (2006). Therefore, some basic decision variables are borrowed from the two works.

8.2.1 Parameters

Other parameters are listed below.

- a_i, the arrival time of Vessel i
- l_i, the length of Vessel i
- w_i, the weight assigned for Vessel i

- $\Theta(i) = \{q \mid \underline{q}^i \leq q \leq \overline{q}^i\}, \forall\, i \in V$, the set of legitimate quay crane numbers for Vessel i

8.2.2 Decision Variables

- s_i, the service start time of Vessel i
- v_i, the starting berth position of Vessel i
- c_i, the service completion time of Vessel i
- $x_{ij} \in \{0,1\}$, 1, if the berthing position of Vessel i is completely lower than the berthing position of Vessel j; 0, otherwise
- $y_{ij}^{ss} \in \{0,1\}$, 1, if s_i is before s_j; 0, otherwise
- $y_{ij}^{sc} \in \{0,1\}$, 1, if s_i is before c_j; 0, otherwise
- $y_{ij}^{cs} \in \{0,1\}$, 1, if c_i is before s_j; 0, otherwise
- $y_{ij}^{cc} \in \{0,1\}$, 1, if c_i is before c_j; 0, otherwise
- $z_{ki} \in \{0,1\}$, 1, if Crane k is assigned to process Vessel i; 0, otherwise
- $\psi_{iq} \in \{0,1\}$, 1, if q quay cranes are assigned to process Vessel i; 0, otherwise
- ρ_{ki}^{s}, the position of Crane k at the start time of Vessel i
- ρ_{ki}^{c}, the position of Crane k at the completion time of Vessel i

8.2.3 Formulation

The formulation $[\mathcal{F}]$ is the synthesis of both the MIP models developed in Guan and Cheung (2004) and Liu et al. (2006).

$$[\mathcal{F}] \min \quad \sum_{i \in V} w_i (c_i - a_i) \tag{8.1}$$

$$s.t.$$

$$v_j \geq v_i + l_i + L \cdot (x_{ij} - 1), \qquad \forall\, i, j \in V, i \neq j \tag{8.2}$$

$$s_j \geq s_i + M \cdot (y_{ij}^{ss} - 1), \qquad \forall\, i, j \in V, i \neq j \tag{8.3}$$

$$c_j \geq s_i + M \cdot (y_{ij}^{sc} - 1), \qquad \forall\, i, j \in V, i \neq j \tag{8.4}$$

$$s_j \geq c_i + M \cdot (y_{ij}^{cs} - 1), \qquad \forall\, i, j \in V, i \neq j \tag{8.5}$$

$$c_j \geq c_i + M \cdot (y_{ij}^{cc} - 1), \qquad \forall\, i, j \in V, i \neq j \tag{8.6}$$

$$y_{ij}^{ss} + y_{ji}^{ss} = 1, \qquad \forall\, i, j \in V, i \neq j \tag{8.7}$$

$$y_{ij}^{sc} + y_{ji}^{cs} = 1, \qquad \forall\, i, j \in V, i \neq j \tag{8.8}$$

$$y_{ij}^{cc} + y_{ji}^{cc} = 1, \qquad \forall\, i, j \in V, i \neq j \tag{8.9}$$

$$x_{ij} + x_{ji} + y_{ij}^{cs} + y_{ji}^{cs} \geq 1, \qquad \forall\, i, j \in V, i \neq j \tag{8.10}$$

$$x_{ij} + x_{ji} \leq 1, \qquad \forall\, i, j \in V, i \neq j \tag{8.11}$$

$$y_{ij}^{cs} + y_{ji}^{cs} \leq 1, \qquad \forall\, i, j \in V, i \neq j \tag{8.12}$$

$$\sum_{q \in \Theta(i)} \psi_{iq} = 1, \qquad \forall\, i \in V \tag{8.13}$$

$$\sum_{q \in \Theta(i)} q \cdot \psi_{iq} = \sum_{k \in Q} z_{ki}, \qquad \forall\, i \in V \tag{8.14}$$

$$\rho_{ki}^{s} \geq v_i + 0.5 l_i + M \cdot (z_{ki} - 1), \qquad \forall\, i \in V, k \in Q \tag{8.15}$$

$$\rho_{ki}^{s} \leq v_i + 0.5 l_i + M \cdot (1 - z_{ki}), \qquad \forall\, i \in V, k \in Q \tag{8.16}$$

$$\rho_{ki}^{c} \geq v_i + 0.5 l_i + M \cdot (z_{ki} - 1), \qquad \forall\, i \in V, k \in Q \tag{8.17}$$

$$\rho_{ki}^{c} \leq v_i + 0.5 l_i + M \cdot (1 - z_{ki}), \qquad \forall\, i \in V, k \in Q \tag{8.18}$$

$$\rho_{kj}^{s} \leq v_i + 0.5 l_i + M \cdot (4 - z_{ki} - y_{ij}^{ss} - y_{ji}^{sc} - x_{ij} - x_{ji}), \qquad \forall\, i, j \in V, i \neq j, k \in Q \tag{8.19}$$

$$\rho_{kj}^{s} \geq v_i + 0.5 l_i + M \cdot (z_{ki} + y_{ij}^{ss} + y_{ji}^{sc} + x_{ij} + x_{ji} - 4), \qquad \forall\, i, j \in V, i \neq j, k \in Q \tag{8.20}$$

$$\rho_{kj}^{c} \leq v_i + 0.5 l_i + M \cdot (4 - z_{ki} - y_{ij}^{sc} - y_{ji}^{cc} - x_{ij} - x_{ji}), \qquad \forall\, i, j \in V, i \neq j, k \in Q \tag{8.21}$$

$$\rho_{kj}^{c} \geq v_i + 0.5 l_i + M \cdot (z_{ki} + y_{ij}^{sc} + y_{ji}^{cc} + x_{ij} + x_{ji} - 4), \qquad \forall\, i, j \in V, i \neq j, k \in Q \tag{8.22}$$

$$\rho_{ki}^{s} - \rho_{k-1,i}^{s} \geq 0, \qquad \forall\, i \in V, k \in Q \backslash \{1\} \tag{8.23}$$

$$\rho^c_{ki} - \rho^c_{k-1,i} \geq 0, \qquad \forall\, i \in V, k \in Q \backslash \{1\} \tag{8.24}$$

$$s_i \geq a_i, \qquad \forall\, i \in V \tag{8.25}$$

$$v_i \leq L - l_i, \qquad \forall\, i \in V \tag{8.26}$$

$$c_i = s_i + \sum_{q \in \Theta(i)} p_{iq} \cdot \psi_{iq}, \qquad \forall\, i \in V \tag{8.27}$$

$$s_i, v_i, c_i, \rho^s_{ki}, \rho^c_{ki} \in \Re^+, \qquad \forall\, i \in V, k \in Q \tag{8.28}$$

$$x_{ij}, y^{ss}_{ij}, y^{sc}_{ij}, y^{cs}_{ij}, y^{cc}_{ij}, z_{ki}, \psi_{iq} \in \{0,1\}, \qquad \forall\, i, j \in V, i \neq j, k \in Q, q \in \Theta(i) \tag{8.29}$$

The objective of this model is to minimize the total weighted flow time as in Guan and Cheung (2004). Constraints (8.2) enforce the definition of x_{ij}. Constraints (8.3) to (8.9) work together to define the binary decision variables y^{ss}_{ij}, y^{sc}_{ij}, y^{cs}_{ij}, and y^{cc}_{ij}. Constraints (8.10) to (8.12) are very important since they ensure that there is no conflict between any two berth plans (with respect to both berthing position and service time) of different vessels. Constraints (8.13) and (8.14) are used to link ψ_{iq} and z_{ki} and make sure that the total number of quay cranes assigned to a vessel is in the legitimate range. Constraints (8.15) to (8.22) are used to track the movement of all quay cranes during the planning horizon through variables ρ^s_{ki} and ρ^c_{ki}. Constraints (8.23) and (8.24) serve as the Non-crossing Constraints for any two quay cranes. That is, the quay cranes should keep relative positions and can not cross over each other during the planning horizon. Constraints (8.25) and (8.26) state that the service start time of Vessel i should be later than its expected arrival time and the entire vessel is docked within the available quay space. Constraints (8.27) define the service completion time for Vessel i. For example, if q^* quay cranes are assigned to Vessel i, then only $\psi_{iq^*} = 1$ and $c_i = s_i + p_{iq^*}$. Finally, Constraints (8.28) and (8.29) declare the domains for all the decision variables.

8.2.4 Improved formulation

Actually, the formulation of $[\mathcal{F}]$ can be simplified in accordance with the proposition below.

Proposition 8.2.1. *In an optimal solution of $[\mathcal{F}]$, there is no a quay crane assignment plan*

such that Cranes k^d and k^u are assigned to Vessel i while Crane k ($k^d \leq k \leq k^u$) is not assigned to Vessel i.

Proof. In an optimal solution of $[\mathcal{F}]$ (called current solution), suppose there exists a quay crane assignment plan that Cranes k^d and k^u are assigned to Vessel i and Crane k is not assigned to Vessel i. Since $k^d \leq k \leq k^u$ and the fact that quay cranes should keep relative positions, it is easy to conclude that during the processing of Vessel i, Crane k has to stay idle. Compared with another solution of $[\mathcal{F}]$ (called new solution) such that quay crane assignment plan for other quay cranes and berthing plan for vessels are intact while Crane k is also assigned to Vessel i, due to the monotonically increasing property of parameter p_{iq} with $q \in \Theta(i)$, the objective value of the current solution should be higher because the service completion time of Vessel i is greater than the one for the new solution, which contradicts the assumption that the current solution is an optimal solution. □

In light of the proposition, to solve the proposed problem, in fact, it is not necessary to introduce z_{ki} in order to obtain the information of the assignment plan for individual quay cranes. Instead, only two decision variables for Vessel i are required in order to capture the quay crane assignment plan for Vessel i. Let z_i^u be the quay crane assigned to Vessel i with the highest numbering and z_i^d be the quay crane assigned to Vessel i with the lowest numbering. Based on proposition 8.2.1, any Crane k, $z_i^d \leq k \leq z_i^u$ should be assigned to Vessel i as well. By introducing z_i^u and z_i^d, $i \in V$, the improved formulation denoted as $[\mathcal{G}]$ is shown below.

$$
\begin{aligned}
[\mathcal{G}] \min \quad & \sum_{i \in V} w_i (c_i - a_i) \\
s.t. \quad & \\
& v_j \geq v_i + l_i + L \cdot (x_{ij} - 1), \qquad \forall\, i, j \in V, i \neq j \\
& s_j \geq c_i + M \cdot (y_{ij}^{cs} - 1), \qquad \forall\, i, j \in V, i \neq j \\
& x_{ij} + x_{ji} + y_{ij}^{cs} + y_{ji}^{cs} \geq 1, \qquad \forall\, i, j \in V, i \neq j \\
& x_{ij} + x_{ji} \leq 1, \qquad \forall\, i, j \in V, i \neq j \\
& y_{ij}^{cs} + y_{ji}^{cs} \leq 1, \qquad \forall\, i, j \in V, i \neq j
\end{aligned}
$$

$$\sum_{q\in\Theta(i)} \psi_{iq} = 1, \qquad \forall\, i \in V$$

$$\sum_{q\in\Theta(i)} q \cdot \psi_{iq} = z_i^u - z_i^d + 1, \qquad \forall\, i \in V \tag{8.30}$$

$$z_i^u - z_j^d + 1 \leq Q \cdot (1 + y_{ij}^{cs} + y_{ji}^{cs} - x_{ij}), \quad \forall\, i, j \in V, i \neq j \tag{8.31}$$

$$s_i \geq a_i, \qquad \forall\, i \in V$$

$$v_i \leq L - l_i, \qquad \forall\, i \in V$$

$$c_i = s_i + \sum_{q\in\Theta(i)} p_{iq} \cdot \psi_{iq}, \qquad \forall\, i \in V$$

$$s_i, v_i, c_i \in \Re^+, \qquad \forall\, i \in V$$

$$x_{ij}, y_{ij}^{cs}, \psi_{iq} \in \{0, 1\}, \qquad \forall\, i, j \in V, i \neq j, q \in \Theta(i)$$

$$z_i^u, z_i^d \in Q, \qquad \forall\, i \in V \tag{8.32}$$

Formulation $[\mathcal{G}]$ is a simplified version of formulation $[\mathcal{F}]$. Compared with $[\mathcal{F}]$, $[\mathcal{G}]$ adds in two new integer decision variables z_i^u and z_i^d for each vessel but eliminates the decision variables ρ_{ki}^s, ρ_{ki}^c, y_{ij}^{ss}, y_{ij}^{sc}, y_{ij}^{cc}, and z_{ki} in $[\mathcal{F}]$. Besides, on the constraint part, the new Constraints (8.30) to (8.32) in formulation $[\mathcal{G}]$ are used to replace the Constraints (8.14) to (8.24) in formulation $[\mathcal{F}]$. In Constraints (8.30), according to the definitions of both z_i^u and z_i^d, the term $(z_i^u - z_i^d + 1)$ is the number of quay cranes assigned to Vessel i. Therefore, Constraints (8.30) are equivalent to Constraints (8.14). In formulation $[\mathcal{F}]$, the decision variables ρ_{ki}^s and ρ_{ki}^c are defined by Liu et al. to monitor the movement of all quay cranes during the planning horizon (i.e., Constraints (8.15) to (8.22)) are ultimately to guarantee the Non-crossing requirement for the quay crane assignment plan (i.e., Constraints (8.23) and (8.24)). In fact, to achieve such a goal, imposing Constraints (8.31) is necessary.

Figure 8.1 illustrates how Constraints (8.31) work. In the Time-space Diagram for the proposed problem, suppose Vessel i is completely below Vessel j, i.e, $x_{ij} = 1$ and at certain period of time, Vessel i and Vessel j are processed simultaneously, i.e., $y_{ij}^{cs} + y_{ji}^{cs} = 0$. Hence, to ensure that the quay crane assignment plan follows the Non-crossing requirement, it is only necessary to guarantee that the numbering of Crane z_i^u is strictly smaller than the numbering of Crane z_j^d, i.e., $z_i^u + 1 \leq z_j^d$.

Formulation $[\mathcal{G}]$ is expected to be an effective formulation compared with formulation $[\mathcal{F}]$ since $[\mathcal{G}]$ truncates a great portion of decision variables introduced in the original formulation and is naturally much smaller in size. Although the solution of formulation $[\mathcal{G}]$ does not contain the information for quay crane moving trajectory, such information can be obtained easily after the berthing plan for vessels and quay crane assignment plan are settled. In the section of numerical experiment, the performance of both formulations $[\mathcal{F}]$ and $[\mathcal{G}]$ will be tested.

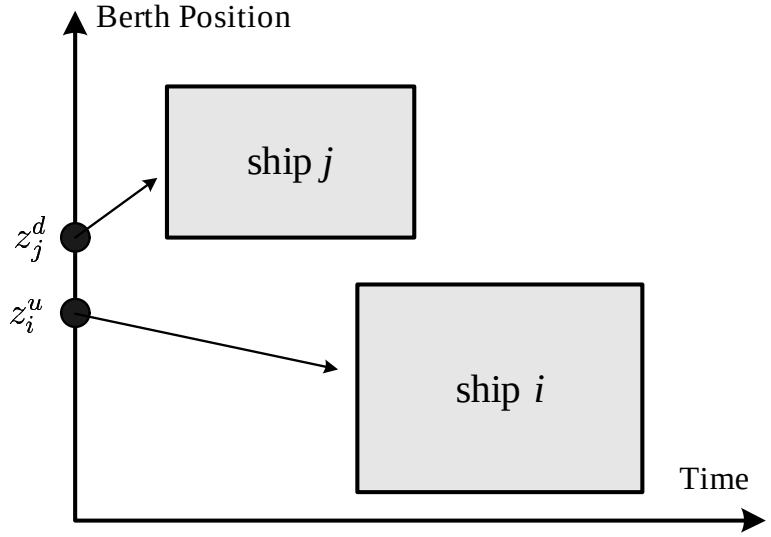


Figure 8.1: An illustration for Constraints (8.31)

8.3 Heuristic Solution

In this section, a greedy search approach will be constructed to solve the proposed quayside operation problem. Compared with most of the greedy search approaches, the heuristic developed in this chapter takes advantage of the current sophisticated MIP solvers, by using them as a black box tool to find satisfactory feasible solutions as early as possible during the computation. Such a strategy is inspired by the work of **?**. For a general MIP problem,

$$\begin{aligned} \min \quad & \mathbf{c}'\mathbf{x} + \mathbf{d}'\mathbf{y} + \mathbf{e}'\mathbf{z} \\ s.t. \quad & (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in H \\ & \mathbf{x}, \mathbf{y} \quad \text{integer} \end{aligned}$$

$\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, $\mathbf{c}$, $\mathbf{d}$, and $\mathbf{e}$ are vectors and H is the convex feasible domain for the decision variables. Denote P as the projection of H onto the space of $\mathbf{x}$ and $\mathbf{y}$. Suppose an initial solution $(\mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0)$ for the MIP problem has been found. The proposed greedy search is applied in the space P (see Figure 8.2) and starts from solution $(\mathbf{x}_0, \mathbf{y}_0)$ attempting to find a better solution in the following manner. First of all, a line search will be conducted by fixing either $\mathbf{x}$ to $\mathbf{x}_0$ or $\mathbf{y}$ to $\mathbf{y}_0$. For illustration, let $\mathbf{y}$ be fixed to $\mathbf{y}_0$ and solve the problem $\min\{\mathbf{c}'\mathbf{x}+\mathbf{d}'\mathbf{y_0}+\mathbf{e}'\mathbf{z} : (\mathbf{x}, \mathbf{y}_0, \mathbf{z}) \in H, \mathbf{x} \text{ integer}\}$, which is equivalent to find a better solution for the original problem along the line of L_1 in the space of P. Suppose the solution for $\min\{\mathbf{c}'\mathbf{x}+\mathbf{d}'\mathbf{y_0}+\mathbf{e}'\mathbf{z} : (\mathbf{x}, \mathbf{y}_0, \mathbf{z}) \in H, \mathbf{x} \text{ integer}\}$ is $(\mathbf{x}_1, \mathbf{y}_0, \mathbf{z}_1)$. The next step for the greedy search is to explore the neighborhood subspace around the point $(\mathbf{x}_1, \mathbf{y}_0)$ in P which is denoted as R_1. Let the solution $(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_2)$ be the optimal solution for the problem $\min\{\mathbf{c}'\mathbf{x} + \mathbf{d}'\mathbf{y} + \mathbf{e}'\mathbf{z} : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in H, (\mathbf{x}, \mathbf{y}) \in R_1, \mathbf{x}, \mathbf{y} \text{ integer}\}$. Plainly, after one line search and one neighborhood search (note that both of them are solved by a generic MIP solver), the heuristic approach has found a better solution $(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_2)$ based on the initial solution $(\mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0)$. By repeating the procedure described previously, the greedy search will find a serial of solutions and finally be terminated once some termination conditions are satisfied.

To adopt the greedy search approach to the proposed quayside operation problem, the decision variables in formulation $[\mathcal{G}]$ need to be classified. Let $\mathbf{x} = \{x_{ij}, y^{cs}_{ij}\}_{i,j \in V, i \neq j}$ which is the integer vector representing the berthing information for vessels. Meanwhile, let $\mathbf{y} = \{\psi_{iq}, z^u_i, z^d_i\}_{i \in V, q \in \Theta(i)}$, the integer vector contains the quay crane assigning plan for every vessel. Finally, let $\mathbf{z} = \{s_i, v_i, c_i\}_{i \in V}$ which is the vector for the rest of continuous decision variables in formulation $[\mathcal{G}]$.

To construct an initial solution for $[\mathcal{G}]$, the following Constraints will be added into $[\mathcal{G}]$ and the new problem will be solved by a MIP solver.

$$y^{cs}_{ji} = 0, \qquad \forall i, j \in V, a_i \leq a_j \tag{8.33}$$

The purpose to adding such cuts is to seek an solution complying with the rule that if the expected arrival time of Vessel i is earlier than the one of Vessel j, then the service start time

of Vessel i should not later than the service completion time of Vessel j (i.e., $s_i \leq c_j$ and thus $y_{ji}^{cs} = 0$). By forcing $\{y_{ji}^{cs}\}_{i,j \in V, a_i \leq a_j}$ equals to 0, the original problem $[\mathcal{G}]$ becomes much easier to solve.

Another issue to apply the greedy search approach for the proposed problem is to define the neighborhood subspace around a known solution. Suppose for the known solution those binary decision variables relating to berthing plan are $\{\hat{x}_{ij}\}_{i,j \in V, i \neq j}$ and $\{\hat{y}_{ij}^{cs}\}_{i,j \in V, i \neq j}$. Then one possible neighborhood subspace around the known solution can be defined by the following boundary cut.

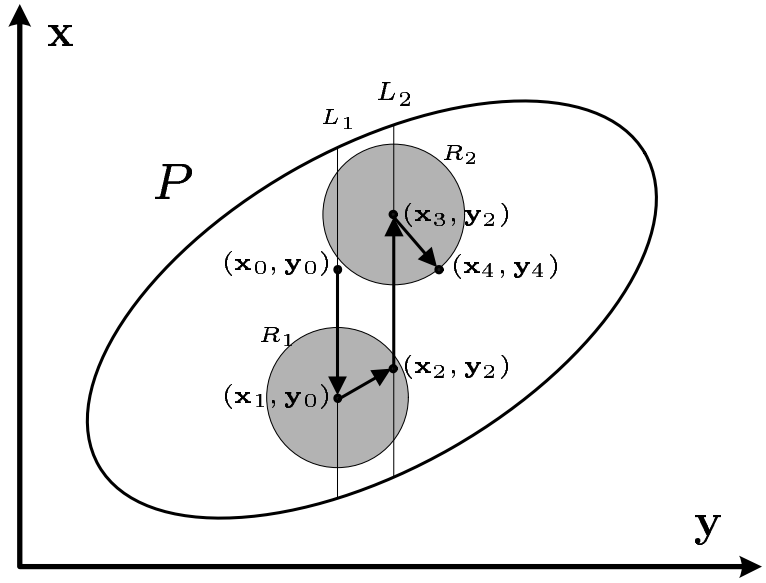


Figure 8.2: An illustration for the proposed heuristic approach

$$\sum_{i,j \in V} (\hat{x}_{ij} \cdot x_{ij} + \hat{y}_{ij}^{cs} \cdot y_{ij}^{cs}) \geq \lceil 0.95 \sum_{i,j \in V} (\hat{x}_{ij} + \hat{y}_{ij}^{cs}) \rceil \tag{8.34}$$

The rationale for such a boundary cut is: let the number of non-zero variables in $\{\hat{x}_{ij}\}_{i,j \in V, i \neq j}$ and $\{\hat{y}_{ij}^{cs}\}_{i,j \in V, i \neq j}$ be n; then for any point in the neighborhood subspace, the sum of variables x_{ij} (if $\hat{x}_{ij} = 1$) and y_{ij}^{cs} (if $\hat{y}_{ij}^{cs} = 1$) should be greater than 95% (say) of the number n. The form of such boundary cut is also referred to as the linear soft fixing constraint in **?**.

To sum up, the procedure for the greedy search approach for the proposed quayside operation problem includes the following steps:

Step 1: Solve the problem $[\mathcal{G}]$ with Constraints (8.33) to obtain the initial solution $\{x_{ij}, y_{ij}^{cs}\}_{i,j\in V,i\neq j}^{0}$, $\{\psi_{iq}, z_i^u, z_i^d\}_{i\in V,q\in\Theta(i)}^{0}$, and $\{s_i, v_i, c_i\}_{i\in V}^{0}$. Initialize the maximum iteration number $k_{\max}$ and let the iteration counter $k = 0$.

Step 2: if $k > k_{\max}$, stop the greedy search approach. Otherwise, proceed to Step 3.

Step 3: Conduct the line search by fixing $\{\psi_{iq}, z_i^u, z_i^d\}_{i\in V,q\in\Theta(i)}$ in the formulation $[\mathcal{G}]$ to $\{\psi_{iq}, z_i^u, z_i^d\}_{i\in V,q\in\Theta(i)}^{k}$. Denote the optimal solution after the line search as $\{\hat{x}_{ij}, \hat{y}_{ij}^{cs}\}_{i,j\in V,i\neq j}$, $\{\psi_{iq}, z_i^u, z_i^d\}_{i\in V,q\in\Theta(i)}^{k}$, and $\{\hat{s}_i, \hat{v}_i, \hat{c}_i\}_{i\in V}$.

Step 4: Conduct the neighborhood search around the solution $\{\hat{x}_{ij}, \hat{y}_{ij}^{cs}\}_{i,j\in V,i\neq j}$, $\{\hat{s}_i, \hat{v}_i, \hat{c}_i\}_{i\in V}$, and $\{\psi_{iq}, z_i^u, z_i^d\}_{i\in V,q\in\Theta(i)}^{k}$. Use the MIP solver to obtain the optimal solution of $[\mathcal{G}]$ within the subspace confined by boundary (8.34). Denote the new solution as $\{x_{ij}, y_{ij}^{cs}\}_{i,j\in V,i\neq j}^{k+1}$, $\{\psi_{iq}, z_i^u, z_i^d\}_{i\in V,q\in\Theta(i)}^{k+1}$, and $\{s_i, v_i, c_i\}_{i\in V}^{k+1}$.

Step 5: $k = k + 1$ and go to Step 2.

8.4 Computational Experiments

In this section, numerical experiments will be conducted to fulfill two purposes: firstly, to test which formulation of the proposed quayside operation problem (i.e., $[\mathcal{F}]$ or $[\mathcal{G}]$) is more preferable and secondly, to evaluate the performance of the proposed greedy search.

To construct an instance for the experiment, for each incoming vessel, suppose its length is measured as the number of bays in it, which is randomly selected from the range of $[8, 20]$. In other words, the length for any vessel is from 8 to 20. All the jobs in each bay are lumped together as a single workload for that bay and the size of workload in each bay is chosen from the range of $[5, 10]$. Additionally, the expected arrival time for each vessel follows the uniform distribution in the interval of $[0, 100]$. And without loss of generality, it is assumed that the weight for each vessel equals to 1 and the minimum and maximum number of quay cranes that can be assigned to one vessel are 1 and 5, respectively. Let the length of quay space L equal to 40. Therefore, the number of vessels that can be moored simultaneously at the quay is ranging from 2 (i.e., L divides the maximum length for all vessels) to 5 (i.e., L divides the minimum

length for all vessels). Besides, the number of quay cranes equipped along the quay is fixed to 8. As noted previously, the parameter p_{iq} is very crucial for the proposed quayside operation problem. Given the pair of i and q, to initialize the value of p_{iq}, the approximation algorithm developed in the Chapter 4 is adopted. The following shows the input for an instance of the problem when the number of vessels is 4.

- $|V|$: 4; Q: 8; L: 40; l_i: [16,19,10,12]; a_i: [24,16,6,12]; w_i: [1,1,1,1]
- p_{iq}:

i,q	1	2	3	4	5
1	122	61	41	34	26
2	151	76	51	41	32
3	80	40	30	24	19
4	86	44	30	22	20

Note that all the experiments are conducted in a PC with 2.40 GHz CPU and 3 GB RAM and the MIP solver used is the ILOG CPLEX 12.1.

Table 8.1 lists the computational results for 10 instances (the number of vessels equals to 4) when they are formulated by different MIP models. As expected, formulation $[\mathcal{G}]$ is much easier to solve compared with the formulation $[\mathcal{F}]$. Although the reason for such phenomenon has been explained in the previous section, the following data are useful to grasp how effective the formulation $[\mathcal{G}]$ is in comparison with formulation $[\mathcal{F}]$: when the number of vessels is equal to 4, the size of the formulation $[\mathcal{F}]$ is 662 rows, 164 columns, 3537 nonzeros, and 88 binaries; while for the formulation $[\mathcal{G}]$, there are only 66 rows, 60 columns, 283 nonzeros, 44 binaries, and 8 generals.

To test the performance of the proposed greedy search algorithm, the instance sets with vessel number 8 and 10 are used. There are two types of stopping conditions for the heuristic:

- the maximum number of iterations reaches 10
- the objective value does not improve for 3 consecutive iterations

Table 8.1: The computational results when the number of vessels equals to 4

	$[\mathcal{F}]$		$[\mathcal{G}]$	
	Obj	Time(s)	Obj	Time(s)
4_1	118	5211	118	0.36
4_2	143	34	143	0.73
4_3	136	719	136	1.20
4_4	171	114	171	1.20
4_5	121	49	121	0.14
4_6	155	278	155	1.00
4_7	137	42	137	1.41
4_8	122	2072	122	1.14
4_9	166	37	166	0.80
4_10	157	345	157	0.33

Tables 8.2 and 8.3 summarize the computational results for the 40 instances (20 instances when the number of vessels is 8 and the other 20 when the number of vessels is 10). In these two tables, the column "Gap" stores the value of

$$\frac{\text{Obj of the heuristic} - \text{the optimal value found by CPLEX}}{\text{the optimal value found by CPLEX}} \times 100\%$$

for all the instances. To evaluate the performance of the proposed greedy search, firstly look at the dimension of solution effectiveness. When $|V| = 8$, the maximum gap between the solution found by the heuristic and the optimal value is around 1.36% for Instance 8_12 and the average gap for the 20 instances is 0.17%. For the case when $|V| = 10$, the maximum gap and average gap are 2.65% and 0.26%, respectively. On the other hand, in terms of efficiency, in order to confirm the global optimum for each instance, the average computational time for CPLEX is 2442.7 seconds when $|V| = 8$ and 4580.2 seconds when $|V| = 10$. Among the 40 instances, the worst case performed by CPLEX is Instance 10_8. It takes more than 7 hours for CPLEX to complete the search. For the proposed greedy search, the average executive time for case $|V| = 8$ and $|V| = 10$ is 387.5 seconds and 452.4 seconds and 35 instances out of the 40 instances

can be finished within 15 minuets (i.e., 900 seconds). However, the greedy search approach does not always run shorter time than CPLEX, for example, Instances 8_3, 8_7, and 10_12 in which instances although the proposed heuristic runs quite fast and achieves the optimums, CPLEX performs much better. In a broader view, given the merits of the proposed greedy search approach, it is confident to claim that the proposed greedy search approach is able to find satisfactory solutions for the proposed quayside operation problem in acceptable time.

Table 8.2: The computational results when the number of vessels equals to 8

	Heuristic		CPLEX		Gap(%)
	Obj	Time(s)	Obj	Time(s)	
8_1	416	110	416	455	0.00
8_2	425	392	425	5681	0.00
8_3	318	147	318	128	0.00
8_4	369	202	369	739	0.00
8_5	335	517	335	2252	0.00
8_6	491	273	491	6407	0.00
8_7	334	50	334	30	0.00
8_8	483	71	483	527	0.00
8_9	330	17	330	31	0.00
8_10	504	440	504	2116	0.00
8_11	369	18	369	64	0.00
8_12	448	54	442	5411	1.36
8_13	351	335	349	561	0.57
8_14	432	2350	432	4754	0.00
8_15	397	195	397	3913	0.00
8_16	301	440	301	735	0.00
8_17	453	235	452	1932	0.22
8_18	505	1095	505	6310	0.00
8_19	421	351	416	4512	1.20
8_20	401	458	401	2295	0.00

Table 8.3: The computational results when the number of vessels equals to 10

	Heuristic		CPLEX		Gap(%)
	Obj	Time(s)	Obj	Time(s)	
10_1	536	560	536	19748	0.00
10_2	437	138	437	234	0.00
10_3	398	119	398	134	0.00
10_4	350	129	350	159	0.00
10_5	394	47	393	70	0.25
10_6	419	875	419	2322	0.00
10_7	499	88	499	312	0.00
10_8	445	2400	445	26104	0.00
10_9	496	53	496	284	0.00
10_10	338	1263	338	3554	0.00
10_11	466	257	466	8781	0.00
10_12	310	126	310	124	0.00
10_13	519	553	519	4347	0.00
10_14	397	362	397	847	0.00
10_15	431	950	426	11343	1.17
10_16	494	125	494	851	0.00
10_17	392	517	392	2811	0.00
10_18	492	209	487	7865	1.03
10_19	464	200	464	292	0.00
10_20	465	77	453	1422	2.65

8.5 Summary

In this chapter, an integrated framework to consolidate all the key information from all aspects of a quayside operation problem in a port container terminal is provided. The proposed problem is formulated as an MIP problem and two different formulations are developed. To solve the problem, a greedy search approach is devised attempting to find out satisfactory solutions as early as possible. Computational experiments are conducted to evaluate the effectiveness of

the two formulations and to test the performance of the proposed heuristic for the quayside operation problem.

Chapter 9

Conclusions

9.1 Concluding Remarks

The quayside operations problem always takes the leading role in the management system of a port container terminal. Through constructing more realistic mathematical models and developing new and effective methods for the corresponding problems, the research presented in this thesis has provided an insightful investigation on how to upgrade the performance level of the quayside operations in a typical port container terminal. It not only attempts to fill the gaps in the previous studies, but also contributes to reflect and highlight the emerging technology updates in port container terminals and introduces the novel MIP solving techniques into this field for applications.

In **Chapter 3**, continuous and dynamic BAP has been studied to minimize the total weighted flow time. Compared with previous research on continuous BAP (Guan and Cheung, 2004; Wang and Lim, 2007), the algorithm to identify all the possible locations in the Time-space Diagram for next vessel is discussed in depth. Although in the work of Chazelle (1983), similar issue has been addressed for the Bottom-left Heuristic in the Bin-packing Problem, the unique feature of continuous BAP leads to the failure of Chazelle's approach to cope with the same problem arising in this field. An efficient method is proposed based on the idea of node classification, which outperforms the exhaustive enumeration approach as demonstrated in the numerical experiments. Additionally, two versions of GRASPs are developed to search for

near optimal solutions. The first GRASP—GRASP_1, attempts to follow the first-come-first-pack rule as much as possible while there is no same rule to comply with in the second version, GRASP_2. Both small and large scale problems have been tested to exhibit the effectiveness of the proposed GRASPs by comparing with CPLEX and the SBS by Wang and Lim (2007). The computational results have indicated that for small scale problems, GRASP_2 is an outstanding method for continuous BAP, while for large scale problems, GRASP_1 is recommended since it has the feature to balance the requirement of effectiveness and efficiency simultaneously.

In **Chapter 4**, firstly the common deficiencies in modeling the QCSP with Non-crossing Constraints in the previous studies are examined. Additional constraints are proposed and the revised model has been proven to guarantee the practicability of the optimal solution for the studied problem. Meanwhile in this chapter, two approximation algorithms are studied to produce effective approximate solutions for the QCSP with Non-crossing Constraints. For BP Method, after examining the properties of the original DP algorithm, its time complexity has been reduced from the original $\mathcal{O}\left(mn^2\right)$ to $\mathcal{O}\left(mn\log n\right)$. Moreover, in order to achieve a better approximation scheme, an EBP Method is proposed as well. The numerical experiments have shown that EBP is able to provide approximation schemes for the QCSP with Non-crossing Constraints efficiently and effectively.

The research presented in **Chapter 5** has extended the traditional QCSP to the environment of indented berth, an innovative concept to surmount the challenges introduced by the emergence of mega-containerships. In comparison with the traditional version of the problem, at indented berth, it is more appropriate to treat the quay cranes as unrelated machines rather than identical ones. Moreover, at indented berth, the quay cranes situated at different sides of the quay are free from the Non-crossing Constraints. Taking these two unique features into account, a group-based QCSP at indented berth is formulated as an MIP problem. To facilitate the search of approximate solutions for the proposed problem, the QCSP at indented berth is decomposed into two subproblems, i.e., assigning subproblem and scheduling subproblem, which can be solved by corresponding designed heuristics individually. Therefore, by solving the two subproblems consecutively, a feasible solution could be constructed. Two versions of Tabu search are also devised to further improve the solution quality. In order to evaluate

the performance of the proposed heuristic framework, a comprehensive numerical experiment has been conducted and its results have shown the good performance of the proposed heuristic framework. Finally, it should be highlighted that the proposed problem is more generic than the traditional one, therefore, the proposed heuristic framework can also be applied to traditional QCSP with minimal modification. Additionally, the heuristic proposed to solve the scheduling subproblem, SSH, has been tested to solve the scheduling subproblem quite well that it can be implemented to enhance the performance of Tabu search in Sammarra et al. (2007), one of the state-of-the-art heuristics designed for the traditional QCSP.

In **Chapter 6**, after the literature review, the framework proposed in Liu et al. (2006) is identified to be a logical and pragmatic platform to integrate the quayside operation problem and then its key component, Berth-level model, is solved by a new exact solution method called the CBC algorithm. The CBC algorithm is a newly developed algorithm to solve the MIP problem especially for the problem that involves logical implications modeled through the big-M approach. Since the formulation of the Berth-level model in Liu et al. (2006) fulfills the prototype of the CBC algorithm and additionally, this formulation can be naturally decomposed into two sub-systems, the CBC algorithm is expected to be a suitable method to solve the studied problem. This conjecture has been tested by a comprehensive numerical experiment and it has shown that the proposed CBC algorithm is capable to save a great amount of computational time compared to CPLEX.

The quayside operation problem always takes the leading role in the management system of a port container terminal. The traditional measure is to decompose the entire problem into a series of sub-problems, which reduces the complexity of the problem but meanwhile cuts off the tight link among the elements. In **Chapter 7**, a reasonable integrated model for the quayside operation problem is proposed in the context of discrete berths. Under the framework, by parameterizing p_{vj} (an idea from Liu et al. (2006)), the information flow of the discrete BAP, the QCAP, and the QCSP has been grouped together naturally as a monolithic part. Initially, the integrated problem is formulated as a mixed integer nonlinear programming problem based on the discrete berth allocation model developed by Monaco and Sammarra (2007), which is then transformed to an equivalent linear counterpart. Through in-depth analysis, three kinds of valid

cut are identified and used to strengthen the linear formulation of the studied problem. For problem solving, a heuristic and a local branching method are developed aiming to provide a menu of viable algorithms for the integrated quayside operation problem. The proposed heuristic consists of two steps, solving a restricted original problem and refining the solution by Tabu search. For the LB Method, in contrast with the conventional one, a problem-specific neighborhood boundary cut is explored and furthermore, a bounding procedure is included to accelerate the algorithm. A comprehensive numerical experiment has been carried out to examine the performance of the proposed algorithms. From the experiment, it can be observed that: 1) the heuristic is a fast algorithm with an acceptable sacrifice in the dimension of solution quality; 2) generally speaking, compared with the B&C in CPLEX, the LB Method is a more efficient exact algorithm to solve the proposed problem. Besides, the LB Method can be transformed to a parallel version without tremendous effort, which is a great potential for the algorithm. Meanwhile, it is important to note that these two algorithms can also be adopted to solve the discrete berth allocation problem with just minor modifications.

Similar to Chapter 7, in **Chapter 8**, great effort has been made to seek the integrated model for quayside operation problem in the context of continuous berths. Instead of constructing a truly integrated but insurmountably complex model, in the study presented in this chapter, a framework which integrates all the key information for all subproblems of the quayside operation problem with continuous berths is proposed. To achieve that, as pointed out in Liu et al. (2006), the basic strategy is to parameterize the information of the QCSP and detach it out of the integrated model for the quayside operation problem. To some extent, the work is a reasonable extension of Liu et al. (2006) by incorporating the continuous BAP developed in Guan and Cheung (2004). Two formulations, i.e., formulations $[\mathcal{F}]$ and $[\mathcal{G}]$, have been constructed for the integrated framework of the quayside operation problem with continuous berths. Herein, the formulation $[\mathcal{F}]$ is a simple combination of the models introduced in Liu et al. (2006) and Guan and Cheung (2004). In contrast, the formulation $[\mathcal{G}]$ is developed by considering the property of the proposed problem. The computational experiments have confirmed that as expected, the formulation $[\mathcal{G}]$ is much more effective than the formulation $[\mathcal{F}]$. For the solution of the proposed quayside operation problem, a greedy search approach which consists

of a serial of line search and neighborhood search is devised. Different from traditional greedy search algorithms, both the line search and the neighborhood search in each iteration of the proposed greedy search approach are solved by an MIP solver with the aim of taking advantage of the current advance and sophisticated MIP solvers developed for industry and academia. Through comprehensive numerical experiments, it has been clearly shown that the proposed greedy search approach possesses the feature of finding satisfactory solutions for the proposed quayside operation problem on a low computational time budget. Additionally, it needs to highlight that the proposed greedy search approach is a generic heuristic for MIP problem and can be customized and adopted to solve other MIP problems arising from different fields.

9.2 Remarks for Future Research

It is acknowledged that the fundamental assumption for the research presented in this thesis is that all the information and data are given and they do not vary throughout the decision making process. However, in real applications, some scenarios might violate the fundamental assumption in the manner of information unavailability or data fluctuation. It should be noted that for almost all engineering applications, the effect of uncertainty is inevitable due to the fact that the involved system is ultimately complex and dynamic. Therefore, it is a common and acceptable practice for engineers to simplify the conundrums by projecting the real but stochastic problems into a deterministic domain. Recently, several techniques such as stochastic programming and robust programming have been developed to cope with the uncertainty in the decision making process so as to guarantee that the generated plans are resilient enough to overcome the influences of the uncertainty. For future works of the quayside operation problem in port container terminals, one of the viable directions is to shift all the deterministic models to stochastic counterparts and to develop corresponding methods for problem solving.

Bibliography

Bierwirth, C., Meisel, F., 2009. A fast heuristic for quay crane scheduling with interference constraints. Journal of Scheduling 12 (4), 345–360.

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

Brown, G., Cormican, K., Lawphongpanich, S., Widdis, D., 1997. Optimizing submarine berthing with a persistence incentive. Naval Research Logistics 44 (4), 301–318.

Brown, G., Lawphongpanich, S., Thurman, K., 1994. Optimizing ship berthing. Naval Research Logistics 41 (1), 1–15.

Cao, J. X., Lee, D. H., Chen, J. H., Shi, Q. X., 2010. The integrated yard truck and yard crane scheduling problem: Blenders' decomposition-based methods. Transportation Research Part E-Logistics and Transportation Review 46 (3), 344–353.

Chae, J.-W., Park, W.-S., Jeong, G., 2008. A hybrid quay wall proposed for a very large container ship in the west terminal of busan new port. In: International Conference on Coastal Engineering in Hamburg, Germany.

Chazelle, B., 1983. The Bottomn-Left Bin-Packing Heuristic: An Efficient Implementation. IEEE Transactions on Computers 32 (8), 697–707.

Codato, G., Fischetti, M., 2006. Combinatorial benders' cuts for mixed-integer linear programming. Operations Research 54 (4), 756–766.

Cordeau, J., Laporte, G., Legato, P., Moccia, L., 2005. Models and tabu search heuristics for the berth-allocation problem. Transportation science 39 (4), 526–538.

Daganzo, C. F., 1989. The crane scheduling problem. Transportation Research Part B-Methodological 23 (3), 159–175.

Dai, J., Lin, W., Moorthy, R., Teo, C., 2007. Berth allocation planning optimization in container terminal. Supply Chain Analysis: A Handbook on the Interaction of Information, System and Optimization, 69.

Ghirardi, M., Potts, C., 2005. Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach. European Journal of Operational Research 165 (2), 457–467.

Guan, Y., Cheung, R., 2004. The berth allocation problem: models and solution methods. OR Spectrum 26 (1), 75–92.

Guan, Y., Xiao, W., Cheung, R., Li, C., 2002. A multiprocessor task scheduling model for berth allocation: heuristic and worst-case analysis. Operations Research Letters 30 (5), 343–350.

Hansen, P., Oğuz, C., Mladenović, N., 2008. Variable neighborhood search for minimum cost berth allocation. European Journal of Operational Research 191 (3), 636–649.

Hariri, A., Potts, C., 1991. Heuristics for scheduling unrelated parallel machines. Computers & Operations Research 18 (3), 323–331.

Henesey, L., Davidsson, P., Persson, J., 2004. Using simulation in evaluating berth allocation at a container terminal. In: Proceedings for the 3 rd International Conference on Computer Applications and Information Technology in the Maritime Industries (COMPIT 04). pp. 61–72.

Hooker, J., 2000. Logic-based methods for optimization: combining optimization and constraint satisfaction. Wiley-Interscience.

Imai, A., Chen, H., Nishimura, E., Papadimitriou, S., 2008a. The simultaneous berth and quay crane allocation problem. Transportation Research Part E 44 (5), 900–920.

Imai, A., Nagaiwa, K., Chan, W., 1997. Efficient planning of berth allocation for container terminals in Asia. Journal of Advanced Transportation 31 (1), 75–94.

Imai, A., Nishimura, E., Hattori, M., Papadimitriou, S., 2007. Berth allocation at indented berths for mega-containerships. European Journal of Operational Research 179 (2), 579–593.

Imai, A., Nishimura, E., Papadimitriou, S., 2001. The dynamic berth allocation problem for a container port. Transportation research part B 35 (4), 401–417.

Imai, A., Nishimura, E., Papadimitriou, S., 2003. Berth allocation with service priority. Transportation Research Part B 37 (5), 437–457.

Imai, A., Nishimura, E., Papadimitriou, S., 2008b. Berthing ships at a multi-user container terminal with a limited quay capacity. Transportation Research Part E 44 (1), 136–151.

Imai, A., Sun, X., Nishimura, E., Papadimitriou, S., 2005. Berth allocation in a container port: using a continuous location space approach. Transportation Research Part B 39 (3), 199–221.

Kim, K., Moon, K., 2003. Berth scheduling by simulated annealing. Transportation Research Part B 37 (6), 541–560.

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

Lee, D. H., Wang, H. Q., Miao, L. X., 2008. Quay crane scheduling with non-interference constraints in port container terminals. Transportation Research Part E-Logistics and Transportation Review 44 (1), 124–135.

Lee, Y., Chen, C., 2009. An optimization heuristic for the berth scheduling problem. European Journal of Operational Research 196 (2), 500–508.

Legato, P., Mazza, R., 2001. Berth planning and resources optimisation at a container terminal via discrete event simulation. European Journal of Operational Research 133 (3), 537–547.

Li, C., Cai, X., Lee, C., 1998. Scheduling with multiple–job–on–one–processor pattern. IIE transactions 30 (5), 433–445.

Lim, A., 1998. The berth planning problem. Operations Research Letters 22 (2-3), 105–110.

Lim, A., Rodrigues, B., Xiao, F., Zhu, Y., 2004a. Crane scheduling with spatial constraints. Naval Research Logistics 51 (3), 386–406.

Lim, A., Rodrigues, B., Xu, Z., 2004b. Approximation schemes for the crane scheduling problem. In: Hagerup, T., Katajainen, J. (Eds.), 9th Scandinavian Workshop on Algorithm Theory. Springer, pp. 323–335.

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

Liu, J., Wan, Y., Wang, L., 2006. Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. Naval Research Logistics 53 (1), 60–74.

Meisel, F., Bierwirth, C., 2009. Heuristics for the integration of crane productivity in the berth allocation problem. Transportation Research Part E 45 (1), 196–209.

Moccia, L., Cordeau, J. F., Gaudioso, M., Laporte, G., 2006. A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. Naval Research Logistics 53 (1), 45–59.

Monaco, M., Sammarra, M., 2007. The Berth Allocation Problem: A Strong Formulation Solved by a Lagrangean Approach. Transportation Science 41 (2), 265.

Moorthy, R., Teo, C., 2006. Berth management in container terminal: the template design problem. OR Spectrum 28 (4), 495–518.

Ng, W. C., Mak, K. L., 2006. Quay crane scheduling in container terminals. Engineering Optimization 38 (6), 723–737.

Nishimura, E., Imai, A., Papadimitriou, S., 2001. Berth allocation planning in the public berth system by genetic algorithms. European Journal of Operational Research 131 (2), 282–292.

Park, K., Kim, K., 2002. Berth scheduling for container terminals by using a sub-gradient optimization technique. Journal of the operational research society 53 (9), 1054–1062.

Park, Y., Kim, K., 2003. A scheduling method for berth and quay cranes. OR Spectrum 25 (1), 1–23.

Peterkofsky, R. I., Daganzo, C. F., 1990. A branch and bound solution method for the crane scheduling problem. Transportation Research Part B-Methodological 24 (3), 159–172.

Resende, M., Ribeiro, C., 2003. Greedy randomized adaptive search procedures. International series in operations research and management science, 219–250.

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

Srivastava, B., 1998. An effective heuristic for minimising makespan on unrelated parallel machines. Journal of the Operational Research Society 49 (8), 886–894.

Tavakkoli-Moghaddam, R., Makui, A., Salahi, S., Bazzazi, M., Taheri, F., 2009. An efficient algorithm for solving a new mathematical model for a quay crane scheduling problem in container ports. Computers & Industrial Engineering 56 (1), 241–248.

Tong, C., Lau, H., Lim, A., 1999. Ant Colony Optimization for the ship Berthing problem. Lecture notes in computer science, 359–370.

UNCTAD, 2008. Review of maritime transport. New York & Geneva: United Nations.

Van de Velde, S., 1993. Duality-based algorithms for scheduling unrelated parallel machines. ORSA Journal on Computing 5, 192–192.

Wang, F., Lim, A., 2007. A stochastic beam search for the berth allocation problem. Decision Support Systems 42 (4), 2186–2196.

Woodruff, D., Zemel, E., 1993. Hashing vectors for tabu search. Annals of Operations Research 41 (2), 123–137.

Zhu, Y., Lim, A., 2006. Crane scheduling with non-crossing constraint. Journal of the Operational Research Society 57 (12), 1464–1471.

Appendix

International Journal (SCI Indexed)

[1] D.H. Lee, Z. Cao, J.H. Chen, J.X. Cao, 2009. Simultaneous Load Scheduling of Quay Crane and Yard Crane in Port Container Terminals. *Transportation Research Record* 2097, 62-69.

[2] D.H. Lee, Z. Cao, J.H. Chen, J.X. Cao, 2009. Load Scheduling of Multiple Yard Crane Systems in Container Terminal with Buffer Areas. *Transportation Research Record* 2097, 70-77.

[3] D.H. Lee, J.H. Chen, 2010. An Improved Approach for Quay Crane Scheduling with Non-Crossing Constraints. *Engineering Optimization* 42(1), 1-15.

[4] D.H. Lee, J.X. Cao, Q. Shi, J.H. Chen, 2010. A Heuristic Algorithm for Yard Truck Scheduling and Storage Allocation Problems. *Transportation Research Part E-Logistics and Transportation Review* 45(5), 810-820.

[5] J.X. Cao, D.H. Lee, J.H. Chen, Q. Shi, 2010. The Integrated Yard Truck and Yard Crane Scheduling Problem: Benders' Decomposition-based Methods. *Transportation Research Part E-Logistics and Transportation Review* 46(3), 344-353.

[6] D.H. Lee, J.H. Chen, J.X. Cao, 2010. The Continuous Berth Allocation Problem: A Greedy Randomized Adaptive Search Solution. *Transportation Research Part E-Logistics and Transportation Review* 46(6), 1017-1029.

[7] D.H. Lee, J.H. Chen, J.X. Cao, 2010. Quay Crane Scheduling for An Indented Berth. *Engineering Optimization*, 43 (9), 985-998.

[8] J.H. Chen, D.H. Lee, J.X. Cao, 2011. Heuristics for Quay Crane Scheduling at Indented Berth. *Transportation Research Part E-Logistics and Transportation Review*, 47 (6), 1005-1020.

[9] J.H. Chen, D.H. Lee, J.X. Cao, 2011. A Combinatorial Benders' Cuts Algorithm for the Quayside Operation Problem at Container Terminals. *Transportation Research Part E-Logistics and Transportation Review*, 48 (1), 266-275.

[10] J.H. Chen, D.H. Lee, J.X. Cao, 2011. An Integrated Model for the Quayside Operation Problem with Discrete Berths. *Computers & Operations Research*, **Submitted**.

[11] J.H. Chen, D.H. Lee, J.X. Cao, 2011. An Integrated Model for the Quayside Operation Problem with Continuous Berths. *Computers & Industrial Engineering*, **Submitted**.

International Conference

[1] J.H. Chen, D.H. Lee, J.X. Cao, 2010. Quay Crane Scheduling for Indented Berth. *89th Transportation Research Board Annual Meeting, Washington D.C., USA*, **Presented**.

[2] D.H. Lee, J.H. Chen, 2011. Quay Crane Scheduling for Multiple Vessels in Container Terminals. *90th Transportation Research Board Annual Meeting, Washington D.C., USA*, **Presented**.

[3] D.H. Lee, J.G. Jin, J.H. Chen, 2011. Integrated Bay Allocation and Yard Crane Scheduling Problem for Transshipment Containers. *90th Transportation Research Board Annual Meeting, Washington D.C., USA*, **Presented**.

[4] D.H. Lee, J.G. Jin, J.H. Chen, 2011. A Tabu Search Heuristic for Group Allocation Problem in Transshipment Hubs. *90th Transportation Research Board Annual Meeting, Washington D.C., USA*, **Presented**.

[5] J.H. Chen, S.X. Zhang, D.H. Lee, 2011. Examination of the Effectiveness and Robustness of the Heuristics for Bay-based Quay Crane Scheduling Problem in Port Container Terminals. *The IEEE International Conference on Industrial Engineering and Engineering Management, Singapore*, **Presented**.