Research on Marine Terminal Logistics:
Crane Double Cycling Models

By

Dusan Ku

A thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy in Information Systems,
the University of Auckland, 2015

Abstract

Marine terminals are the gateways in global distribution networks and play a vital role in completing the global supply chains. With the rapidly growing level of service requirement that shipping lines make of terminal operators, the efficiency of terminal operation is becoming a matter of business survival. Thus, a substantial amount of investment has been made in port developments, in both facilities and IT infrastructure especially over the last two decades. Pursuant to this inevitable trend in industry, we also observe that a rich body of research has emerged in the academia for the study of optimisation in marine terminal operations.

This thesis focuses selectively, among other optimisation problems in marine container terminals, on the seaside planning and optimisation problems. As a viable and economic option to improve vessel productivity, which is regarded as the most critical key performance indicator in marine terminal operations, the crane double cycling strategy has received substantial interests from both academia and industry. However, the practical operation strategy for implementing such concept has been limited to a bare minimum partially due to a lack of practical research or practitioners' fear for a change against the status quo. This is where the motivation for this research has arisen.

This research is comprised of three original papers. **Paper I** provides a comprehensive review of the literature on the quay crane (QC) schedule problem and fills the gap in the literature by providing an exact model for preemptive schedules of bay-wise vessel operation. **Paper II** provides a comprehensive review on the research of crane double cycling problem and identifies that the next challenge in this research is in the formulation of the multi-quay crane double cycling problem. Thus, it presents the motivations and discussions on future research directions with a focus on practical constraints. Finally, **Paper III** presents the multi-QC double cycling model that this research is aimed at developing.

As a computational study, the models presented in this research are based on computational and optimisation theory. Methodology section in the introductory chapter addresses the relevant theory and implementation perspective applied to each model: In **Paper I**, the QC scheduling problem is formulated into a *mixed integer programming*. Due to its $\mathcal{NP}$-completeness, a relaxed formulation with Lagrangian relaxation is also presented. In **Paper III**, the multi-QC double cycling problem is formulated into an *integer programming*. Also due to its $\mathcal{NP}$-completeness, a meta-heuristic approach is presented to cope with the computational intractability. Computational results are shown for analysis with discussions on the future research paths.

Acknowledgements

For so long, I have often dreamt about the moment of writing Acknowledgements after finishing writing this thesis. As soon as this dream came true, I came to realise I am in debt to so many people through the making of this thesis. Especially, I thank my supervisor, Dr Tiru Arthanari, for always challenging me and commenting on my work with a great deal of patience. He stepped in to be my supervisor in the beginning, and has since imparted the knowledge of science and many other areas to me. I also extend my gratitude to my co-supervisor, Dr Tava Olsen, for assisting me with lots of enlightening feedbacks on my work. I do feel privileged to have worked with this valuable supervisor group. Special thanks to Dr Chan-Hyouk Cho and Dr Dai-Yon Cho, who had supported me for my PhD application with their reference letters. Without them, I could not have even started it.

I thank colleagues in the department of information systems and operations management: especially, to Dr Valery Pavolv, Dr Arvind Tripathi, Dr Ami Peiris, Ms Gabrielle Peko and Dr David Sundaram for constantly and kindly checking on me whenever running into the department corridor or the kitchen, to Dr Don Sheridan, Dr Lech Janczewski and Dr Cecil Chua for their good senses of humour, let alone their words of wisdom, and to Dr Michael Myers and Dr Lesley Gardner for their timely responses whenever a department approval was needed.

Special thanks should go to Dr Ananth Srinivasan and Dr Anson Li for allowing me an opportunity to assist in the quantitative study courses, to Lecturer David Gardiner for allowing me an opportunity to assist in the supply chain management course, and to Dr Hua Jonathan Ye for allowing me an opportunity to assist in the management information systems course. Other special thanks should go to Claris Chung, Ron Tiong and Josephine Lee. They were like a secondary supervisor group to me although they may humbly deny it. Also to Susanna Yau, Molly Freeman and Elena Tang for their support at the post graduate office, and to Gabrielle Murphy, Kamla Singh, Mary Hoong, Susan Sum and Chrissy Bretherton for their help at the department office, I am deeply grateful.

Starting the PhD journey four years ago was the biggest decision ever in my life. I would not have been able to complete it without the financial support from the University of Auckland and my alma meter, Handong Global University. I must thank the scholarship committee at the University of Auckland for the doctoral scholarship award.

My PhD friends who accompanied me along this journey are the most precious ones in making the PhD life such a pleasant memory: Ali Vahabzadeh, Andrea Herrera, Angel Hsu, Bonnie Wang, Foad Marzoughi, Gloria Liu, Hameed Chughtai, Hamidreza Shahbaznezhad, Jenny Liu, Jing Zhao, Koteswara Ivaturi, Leila Etaati, Loic Li, Maria Kim, Max Rhode, Miya Qian, Mojtaba Mahdavi, Omid Sherkat, Parastoo Farahani, Peter Shi, Samsul Islam, Spring Zhou, Shandong Mou and Violette Wen. Special thanks to Violette for backing me up whenever I was in need of help, and to Gloria for her proof-reading this thesis.

There are other groups of people in Auckland I wish to thank here. Members at Auckland Dream Church have made my life at Auckland so much enjoyable, and especially pastor Soo-Yong Han has been such a good friend both physically and spiritually. Lee's family (Tae-Sung, Jang-Sook, Paul, Sunny and Annie) with whom I have once lived together as a flatmate has been like my second family.

Lastly, I sincerely thank my family, who have patiently supported me throughout the long journey into the PhD study: my wife Soonae, my dearest little princess Yejin, and my dearest little prince Junha. Without Soonae's relentless support, this thesis would not have been born in this world. I also thank my brothers, Duseok and Hosung, for their unfailing support and care for our parents especially while I was thousands miles away overseas. And to my mother, Hwashin, I am so much grateful to you for your love and never-ceasing prayers. I wish to dedicate this thesis to my late father, who was in a hospital bed around the time I started my PhD study. Four years ago, I promised him that I would come back to him with this doctoral thesis, but he can no longer share this day with his son.

My apologies to those whose names I have omitted here. I still owe you a lot!

THE UNIVERSITY OF AUCKLAND
SCHOOL OF GRADUATE STUDIES

Co-Authorship Form

Graduate Centre
ClockTower – East Wing
22 Princes Street, Auckland
Phone: +64 9 373 7599 ext 81321
Fax: +64 9 373 7610
Email: postgraduate@auckland.ac.nz
www.postgrad.auckland.ac.nz

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work**. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 2: Quay Crane Scheduling in a Preemptive Bay Operation terminal. Unpublished	
Nature of contribution by PhD candidate	As first author, Dusan Ku has taken the lead in writing this article with editing done by the co-author.
Extent of contribution by PhD candidate (%)	90%

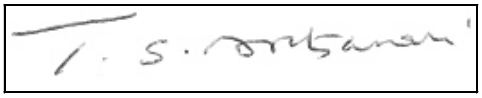
CO-AUTHORS

Name	Nature of Contribution
Tiru Arthanari	Editing

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- in cases where the PhD candidate was the lead author of the work that the candidate wrote the text.

Name	Signature	Date
Tiru Arthanari		1/06/2015
		Click here
		Click here
		Click here
		Click here
		Click here

THE UNIVERSITY OF AUCKLAND
SCHOOL OF GRADUATE STUDIES

Co-Authorship Form

Graduate Centre
ClockTower – East Wing
22 Princes Street, Auckland
Phone: +64 9 373 7599 ext 81321
Fax: +64 9 373 7610
Email: postgraduate@auckland.ac.nz
www.postgrad.auckland.ac.nz

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work**. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 3: On double cycling for container port productivity improvement. Published in Annals of Operations Research (2014)	
Nature of contribution by PhD candidate	As first author, Dusan Ku has taken the lead in writing this article.
Extent of contribution by PhD candidate (%)	80%

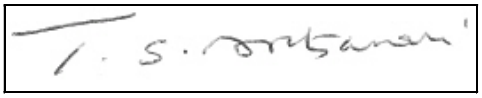
CO-AUTHORS

Name	Nature of Contribution
Tiru Arthanari	Identification of a paper possibility based on presentation in progress review and editing the article.

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- in cases where the PhD candidate was the lead author of the work that the candidate wrote the text.

Name	Signature	Date
Tiru Arthanari		6/10/2014
		Click here
		Click here
		Click here
		Click here
		Click here

THE UNIVERSITY OF AUCKLAND
SCHOOL OF GRADUATE STUDIES

Co-Authorship Form

Graduate Centre
ClockTower – East Wing
22 Princes Street, Auckland
Phone: +64 9 373 7599 ext 81321
Fax: +64 9 373 7610
Email: postgraduate@auckland.ac.nz
www.postgrad.auckland.ac.nz

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work**. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 4: A mathematical formulation for the multi-QC double cycling problem in a marine container terminal. Unpublished	
Nature of contribution by PhD candidate	As first author, Dusan Ku has taken the lead in writing this article.
Extent of contribution by PhD candidate (%)	90%

CO-AUTHORS

Name	Nature of Contribution
Tiru Arthanari	Suggestions for including resource constraints type inequalities to handle violated practical restrictions and editing the article.

Certification by Co-Authors

The undersigned hereby certify that:

- the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- in cases where the PhD candidate was the lead author of the work that the candidate wrote the text.

Name	Signature	Date
Tiru Arthanari		1/06/2015
		Click here
		Click here
		Click here
		Click here
		Click here

Table of Contents

List of Tables

Introduction

Paper I: Quay Crane Scheduling in a Preemptive Bay Operation

Paper II: On Double Cycling for Container Port Productivity Improvement

Paper III: A Mathematical Formulation for the Multi-QC Double Cycling Problem in a Marine Container Terminal

List of Figures

Introduction

Paper I: Quay Crane Scheduling in a Preemptive Bay Operation

Paper II: On Double Cycling for Container Port Productivity Improvement

Paper III: A Mathematical Formulation for the Multi-QC Double Cycling Problem in a Marine Container Terminal

List of Papers

This thesis is comprised of three original articles as well as the introductory and the conclusion chapter. The original articles are listed below with the indication of publication and the author's contribution with a brief summary.

Paper I

> Ku, D., & Arthanari, T. S. (2015). 'Quay Crane Scheduling in a Preemptive Bay Operation' — *Unpublished*

In this paper, the quay crane scheduling problem (QCSP) is formulated into a *mixed integer programming* model, in which the objective function is aimed for the minimisation of the makespan. Due to the problem class being $\mathcal{NP}$-complete, a relaxed formulation with Lagrangian relaxation is also presented. Using the practically modified configuration in the QCSP, i.e., non-preemptive cluster-based tasks → preemptive bay-wise tasks, and bidirectional crane movement → unidirectional crane movement, the computational result presents that the approach of this paper can yield optimal solutions (or at least improved lower bounds) to some benchmark instances that have previously remained open.

As the first author, Dusan Ku has taken the lead in writing this article with editing done by the co-author.

Paper II

> Ku, D., & Arthanari, T. S. (2014). On Double Cycling for Container Port Productivity Improvement. *Annals of Operations Research*, 1-16. doi: 10.1007/s10479-014-1645-z

In this paper, existing models for crane double cycling operations are first reviewed in a comprehensive manner. This paper then identifies that the next challenge in the research of crane double cycling problem is in the formulation of the multi-quay crane double cycling

problem. The discussions on the real world requirements and the suggestions for future research paths presented in this paper are directly linked to the model developed in the follow-up research, Paper III.

As the first author, Dusan Ku has taken the lead in writing this article with the co-author identifying of a paper possibility based on presentation in progress review and editing the article.

Paper III

> Ku, D., & Arthanari, T. S. (2015). 'A Mathematical Formulation for the Multi-QC Double Cycling Problem in a Marine Container Terminal' — *Unpublished*

In this paper, the multi-QC double cycling problem (MQCDCP) is formulated into an *integer programming* (IP) model in which the objective function is aimed for the minimisation of the weighted sum between the makespan and the number of double cycles. Especially the decision variables in this model are all binary, thus the model being also referred to as zero-one integer programming model. Due to the problem class being $\mathcal{NP}$-complete, a meta-heuristic approach is presented to cope with the computational intractability for this problem. A two-phased heuristic decomposes the scheduling problem into two phases: constructive phase and improvement phase. The former, using a greedy type rule, generates initial feasible solutions, which in turn are fed into the latter. The improvement phase gradually improves the solution quality using a local search, for which tabu search technique is employed. Computational results are shown for analysis with the discussion on the future research paths.

As the first author, Dusan Ku has taken the lead in writing this article with the co-author making suggestions for including resource constraints type inequalities to handle violated practical restrictions and editing the article.

Abbreviations

ACO	Ant colony optimisation
AGV	Automated-guided vehicle
B&B	Branch-and-bound
B&C	Branch-and-cut
B&P	Branch-and-price
BAP	Berth allocation problem
BRP	Block relocation problem
BS	Beam search
CFS	Container freight station
CHE	Container handling equipment
CP	Constraint programming
CRP	Container relocation problem
DBAP	Dynamic berth allocation problem
DCQCSP	Double cycling quay crane scheduling problem
DP	Dynamic programming
FCFS	First-come-first-served
GA	Genetic algorithm
GAP	Generalised assignment problem
GP	Genetic programming
GRASP	Randomized adaptive search procedure
IP	Integer programming
ISO	International standards organization
IT	Information technology
LP	Linear programming
LPT	Longest processing time first
LTM	refers to the algorithm used in (Legato, Trunfio, & Meisel, 2012)
MH	Mobile harbour
MIP	Mixed integer programming

MQCDCP	Multi-quay crane double cycling problem
MS	Management science
OOG	Out-of-gauge
OR	Operations research
PSO	Particle swarm optimisation
PTS	Probabilistic tabu search
QAP	Quadratic assignment problem
QC	Quay crane
QCAP	Quay crane assignment problem
QCASP	Quay crane assignment and scheduling problem
QCDCP	Quay crane double cycling problem
QCSHPP	Quay crane scheduling problem with handling priority problem
QCSP	Quay crane scheduling problem
QCSPTW	Quay crane scheduling problem with time windows
RMGC	Rail-mounted gantry crane
RS	Reach stacker
RTGC	Rubber-tyred gantry crane
SA	Simulated annealing
SBAP	Static berth allocation problem
SC	Straddle carrier
SO	Simulation-based optimisation
SWO	Squeaky wheel optimisation
TAS	Truck appointment system
TOS	Terminal operating system
TS	Tabu search
TSP	Travelling salesman problem
TV	Transport vehicle
UDS	Unidirectional Schedule
YC	Yard crane
YT	Yard tractor

Introduction

This thesis focuses on research investigating the operational strategy of seaside operation, especially areas of container terminal operations, aiming to improve the performance of the seaside interface. More specifically, the thesis aims at the practical employment of the crane double cycling strategy, which is regarded as a cost efficient option to improve the vessel operation but often is not employed in a terminal operation. The main contributions of the thesis to knowledge and research are in the directions and frameworks for employing the crane double cycling strategy for the vessel operation.

The remainder of this chapter is organised as follows: The background that has motivated this research is provided in Section 1. Section 2 provides a description of the terminal logistics, which serves as a general understanding of what the terminal logistics are. Section 3 covers the seaside scheduling problems in a broader sense. Section 4 states the aims and objectives for doing this research. Section 5 provides the research methodology adapted in this thesis. Lastly, a graphical overview of papers positioned in this thesis is given.

1 Background

Seaport terminals are the gateways in the global distribution network for cargo and freight. Around 80% of global trade by volume and over 70% of global trade by value are carried by sea and are handled by ports worldwide. Prospects for the world economy, trade and shipping seem to be improving although a number of risks still remain. In line with a 5.2% increase estimated for 2012, the world container port throughput has increased by 5.1% to 651.1 million 20-foot equivalent units (TEUs) in 2013. Despite relatively weak growth in port throughput volumes compared to the trend prior to the economic crisis, the terminal operating

sector is still very active. Several global terminal operators have sold part of their stakes as they seek to streamline and focus on their operations. Terminal operators closely linked to shipping lines have sold terminals, while traditional global terminal operators, such as DP World and Stevedoring Services of America, have attempted to strengthen their position by focusing upon investment (UNCTAD/RMT, 2014).

Over the last several decades, a substantial amount of investments has been made in port developments in both port facilities and its IT infrastructure. Considering the rapidly growing level of service requirement that shipping lines make of the terminal operators, the efficiency of terminal operation is becoming a matter of business survival. Reflecting these inevitable trends, much research has been conducted and the terminal facilities equipped with the state-of-the-art technology and decision support systems using OR/MS techniques are being deployed for the container handling systems. In most cases, however, the cost of implementing such solutions is very significant, thus requiring a strategic decision from the high-level stakeholders bearing responsibilities on their hands.

Consequently, although new facilities or a new decision support systems may increase the terminal productivity, the terminal operators are often reluctant to take this option due to the high investment cost associated with these options. In this thesis, we study a more viable option to boost the productivity of the vessel operation without incurring a significant amount of investment. In comparison to other measures, *crane double cycling* strategy in seaside operation is deemed a low-cost method to boost the vessel productivity; it does not require new technology or infrastructure. Although double cycling will not solve the capacity problem in the long term, it can be more quickly implemented than other solutions, and can be used to complement other strategies (Goodchild & Daganzo, 2006). We will continue our discussion on the *crane double cycling* strategy in the following chapters.

2 Terminal logistics

2.1 Functions of container terminal

There are two major functions that a container terminal performs: *transport* and *storage*. For the transport function, a container terminal processes mainly import, export, or transhipment cycles of container cargoes with respect to the container traffic. For the storage function, a container terminal stores containers in a terminal yard for a certain period. Empty containers which are not laden with goods for transport can be stored at a container terminal for a contracted period. For this category of empty containers, many container terminals have a dedicated area called *empty depot*.

In its broader sense, a container terminal may refer to either a seaport container terminal or an inland container terminal. A seaport container terminal has berths for ships to be attached to the shore, so containers are unloaded from or loaded onto ships via equipment called *quay crane* (QC). An inland container terminal which does not have berths serves a wide range of functions such as a cargo consolidation and deconsolidation centre, where containers are stuffed or stripped, sorted, packed and transported either to seaports or to inland destinations (Hayut, 1980). In this research we consider only a seaport container terminal for study.

Figure 2.1 illustrates the logistics flow of containers through a container terminal. Depending on the terminal environment, some terminals may not have an empty depot, a container freight station (CFS), a rail area or a truck interchange area.

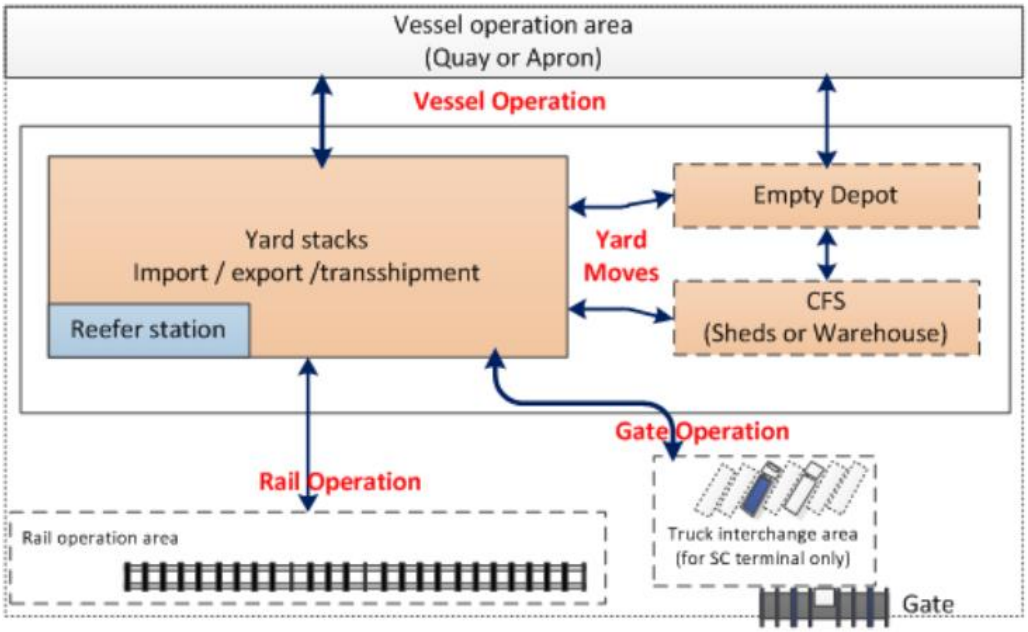


Figure 2.1 Conceptual operation flow of container transport in a container terminal

An export container typically follows the following flow:

1) An export container enters the terminal either via gate, rail, barge or feeder;
2) A container handling equipment (CHE) such as a straddle carrier (SC), a yard crane (YC) or an empty handler like a reach stacker (RS) or a forklift unloads the container from the external transporter and stores in an appropriate place in yard until it is loaded onto the ship;
3) When it is time to be loaded, a SC or a yard tractor (YT) transports the container to the quay;
4) The QC loads the container in the cell planned in the vessel stowage planning, *aka* ship loading plan, which specifies the location of containers to be loaded.

An import container typically follows the following flow:

1) An import container is unloaded from the vessel by a QC;
2) A SC or an YT transports and stores the container in an appropriate place in the yard until delivered to an external transporter;
3) When the container leaves the terminal via the gate, a CHE loads the container onto the external truck. When it leaves via rail, yard moves to transport the container to the

rail buffer area often takes place in advance. Then, a rail crane, typically a rail-mounted gantry crane (RMGC), loads the container onto the train wagon. In cases where the terminal is not equipped with a rail crane, a RS or a SC may do this job.

A transhipment container arrives via a ship and leaves via another ship. The arrival flow resembles the import flow and the departure flow resembles the export flow. If the departure vessel is on berth at the time that the transhipment container is unloaded, the transhipment container may be transported directly to the departure vessel instead of transiting over the yard.

2.2 Containerisation

Since its introduction in the sixties, the container has rapidly taken over the market for intercontinental transport (Meersmans & Dekker, 2001). Containers are typically owned by shipping lines and each shipping company may have its own preferred specification for manufacturing container boxes. Some shippers (usually large shippers) may have their own container boxes, but containers of shipping lines are a major portion of the freight transport via seaport. Nowadays, dimensions of containers for general freight purpose have been standardised by the International Standards Organization (ISO), which enables uniform container handlings across machineries and sites. This standardisation has allowed large savings of time and cost, thus contributing to the wide spread of containers.

The most typical container length is either 20 or 40 feet. However, containers with lengths such as 10, 23, 25, 27, 30, 45 or 48 feet are also in circulation within limited routes or usage. The most common container height is 8′6″ (8 feet 6 inches), but there are also several variations such as half-heights (4′, 4′3″ or 4′6″), normal-heights (8′ or 9′), and high-cubes (9′6″ or 10′6″). Container width varies, too. The most common width of a container is 8′ or 2.438m, but there are containers with extra width, called *pallet-wide* containers.

In addition to the physical dimension, there are several types of containers in which different types or dimensions of cargoes can be allowed. The most common type is the general purpose

container, also known as the dry-van container. Other container types are refrigerated containers for fresh goods, hazardous containers for dangerous goods, tank containers for liquid material, platform containers, usually for heavy load or out-of-gauge (OOG) cargoes, and so on. Several sources about the container specification are available online typically from any shipping line. Refer to (Hapag-Lloyd) as a reference for the details.

2.3 Handling equipment

A variety of container terminals exist mainly depending on which type of handling equipment is combined to form a terminal system (Dirk Steenken, Voß, & Stahlbock, 2004). For QCs, rail-mounted gantry types are common for *almost* all terminals. By '*almost*', we note that some terminals have mobile harbour cranes instead (Gottwald Port Technology, 2012). The transport between quay and yard storage or between yard stacks can be performed by trucks (trailers, multi-trailers, road chassis or automated-guided vehicles (AGVs) or SCs.

Physical handling of containers at container terminal consists of either vertical lift-on and lift-off or horizontal transport. Some equipment can perform both types of handlings fully whereas some others can perform only one type of handling or a limited level of both handlings. Note that equipment listed in Table 2.1 does not cover all types of handling equipment in use at ports due to the variety of equipment and technological advancement. An automatic lifting vehicle (ALV), which can perform both types of handlings, is deemed one of such recent advancements, but is not prevalent in industry. Some research on ALV, for example, can be found from (Meer, 2000; V. D. Nguyen & Kim, 2009; I. F. Vis & Harika, 2004; Yang, Choi, & Ha, 2004).

Table 2.1 Physical handling type by handling equipment

Equipment	Lift-on/off	Transport	Image
Straddle carrier	Yes	Yes	

Yard crane	Yes	No	
Reach stacker	Yes	Limited	
Empty handler (Top handler, forklift, etc.)	Yes	Limited	
Truck (with trailer/chassis, multi-trailers or AGV)	No	Yes	

(*Images sourced from Google image*)

In terms of internal transport of a container, SC and YT are the main vehicles that horizontally transport containers within a terminal. Some terminals use only SCs; other terminals use YTs only; yet another group of terminals use both types of carriers. The difference in the vehicle configuration with respect to transport scheduling is significant such that the whole optimisation strategy for scheduling and dispatching of vehicles is largely affected. Therefore, it would be instructive to provide a short description of container transport by each configuration.

- SC only
  Containers received from the external carrier (vessel, train or truck) are picked up by SC from the interchange area (quay buffer, rail buffer or truck interchange area) and then moved to the appropriate positions in the yard. Departing containers are picked by SC from the stacked yard and moved to the quay buffer (for vessel), rail buffer (for train) or external truck waiting on the truck interchange area (for import pickup).
  When the terminal has an empty depot for storing empty containers, the transport for empty containers may require coupled movement with empty handlers. For example, external trucks will go directly to the empty depot instead of going through the truck interchange area; SCs will move to the interchange area for empty depot instead of

directly accessing the empty depot. Then, it is the empty handlers' job to transport containers to/from the empty depot and the interchange area.

- YT only
  Containers delivered from external carrier (vessel or train) are loaded onto internal trucks at each interchange area and then moved to appropriate positions in the yard. External trucks that deliver the containers will directly go to the appropriate positions. YCs positioned at yard storage area will unload the containers from the trucks and stack them in the designated slots. Departing containers will follow the above transport flow in a reversed order.
- Combination of both
  The combination of both SC and YT as the internal carrier is quite complicated and each terminal may have different transport rules in different configurations. An example of such a terminal can be found in Piraeus Container Terminal (PCT) in Greece, a bird-eye view of which is given in Figure 2.2. This may sometimes occur when a terminal is in a transition from SC operation to YC operation or vice versa. It also may happen when a terminal has a small number of SCs such that a full vessel operation only by SCs is not possible. If the latter is the case, some or most transport may be carried on by YT between quay area and yard storage area. Then, SC will transport containers to/from yard storage location and the interchange area near the storage location. K. Y. Kim and K. H. Kim (1999) have modelled a scenario like this in their study on the routing of SCs for export containers.

  Another possible transport scenario would be the following: Assume that a terminal has both an SC storage area and a YC storage area where the YC storage area is accessible by both SC and YT. Containers bound to the SC storage area will of course be transported by SC. Containers bound to YC storage area will be transported by either SC or YT depending on their availability.

Figure 2.2 Combination between YT and SC operation. *Source*: http://www.pct.com.gr/pct_site

Before closing this subsection, we need to note there is another stream of horizontal transport means called *wheeled* (road chassis) operation. Wheeled operation is commonly employed in North America, where containers are moved directly to and from the terminal on a road chassis. The containers are placed on a road chassis and then parked; Obviously, this operation system takes considerable space due to the fact that the containers are not stacked vertically (Henesey, 2006). An important characteristic of wheeled operation is that it is much less labour-intensive than stacked operations, and therefore incurs lower labour costs (Le-Griffin & Murphy, 2006). Storing containers on parked chassis in terminal yard is common in RoRo [1] terminals (Ku, 2009), but its usage in container terminals is geographically limited to North American ports, which is reflected in the tendency that only researchers from that region deal with wheeled operation in container terminals; (Dowd & Leschine, 1990; Goodchild & Daganzo, 2007; Le-Griffin & Murphy, 2006). One reason that the wheeled container handling system is rarely adopted in Asia or Europe is that in those regions, terminal land is treated as a scare resource and more flexible manning regulations are applied (Le-Griffin & Murphy, 2006).

2.4 Storage yard

Storage capacity in container terminal is often measured in terms of 20-foot equivalent unit (TEU). Lines drawn on yard terrain typically identify the address of each container slot. There may be some temporal storage space often called *heap area* without any line drawn on

[1] RoRo: Abbreviation for Roll-on Roll-off. In contrast, container terminal is regarded as LoLo (Lift-on Lift-off) terminal.

the terrain, but relatively they involve only a bare minimum of containers and are not a major part of the terminal logistics. The storage area with lines drawn is usually separated into different stacks (or blocks), which are differentiated into rows, bays and tiers. Some stack areas are reserved for special containers like reefers which need electrical connection, dangerous goods, or over-height/over-width containers which do not allow for normal stacking. Often stacks are separated into areas for export, import, special and empty containers (Dirk Steenken et al., 2004). One of the classic combinatorial optimisation problems in the stack area is the *container relocation problem* (CRP) or the *blocks relocation problem* (BRP), the objective of which is to minimise unavoidable but unproductive container moves that merely relocate the container positions within the stack area.

As a matter of industrial jargon, the term *stack* sometimes misleads a reader because it may also refer to a pile unit as in Figure 2.3. Therefore, the present paper will refer to it as '*block*' as used by (C. Zhang, Liu, Wan, Murty, & Linn, 2003). Each block consists of a mass of containers, usually placed in six lanes side by side, with each lane including 20 or more container stacks that are of four to six tiers of containers. The number of lanes and the height of the container stacks depend on the height and the span of the container handling equipment that is used to stack containers in a block. The six-lane, four-tier stack size is typical for a rubber-tyred gantry crane (RTGC) block (C. Zhang et al., 2003).

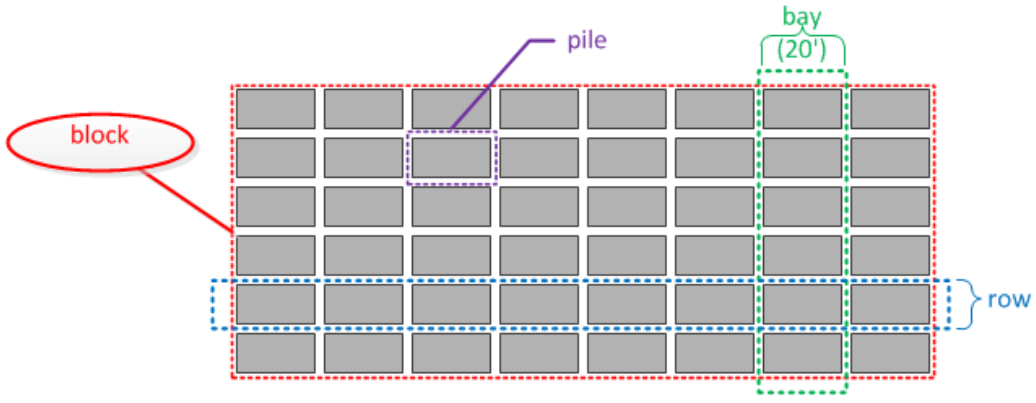
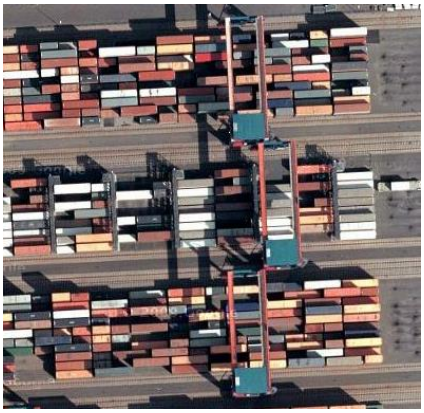


Figure 2.3 Bird-eye view for a block

An RMGC block has more lanes (typically 9-12 lanes) within a block due to its wider span of RMGC. For an SC block, the distinction by a block is somewhat blurry because SC can span only one lane at a time, each block usually contains more than 20 lanes, and each lane within

a block may have different number of bays. RS (or forklift) block is quite dense and the container handling is available only from the side of the block. Table 2.2 presents bird's-eye views for each type of block layout.

Table 2.2 Block layouts by handling equipment

	
RTGC block (Noatum Container Terminal Valencia, Spain)	RMGC block (HHLA CTA, Germany)
	
SC block (Ports of Auckland, New Zealand)	RS (or forklift) block (Noatum Container Terminal Valencia, Spain)

(*Images sourced from Google maps*)

2.5 Vessel layout

A container ship can be modelled as a three-dimensional matrix, each dimension being termed by bay, row and tier, respectively (Figure 2.4). Containers are stacked on top of another and arranged in hatches (or bays) laterally. A hatch is typically composed of two 20-foot bays, thus enabling to stack one 40-foot container or two 20-foot containers. Hatch covers separate a bay (or hatch) vertically into hold and deck, and each bay stretches across

the width of the ship. Large vessels may hold up to around 20 stacks of containers across the width of the ship, and up to around 20 hatches along the length of the ship (Goodchild & Daganzo, 2007). Figure 2.5 illustrates the side view of this structure, and Figure 2.6 provides an example of a bay plan in which containers are labelled in colour.

Figure 2.4 Principle of bay-row-tier coordinates in a ship. *Source*: (GDV, 2012)

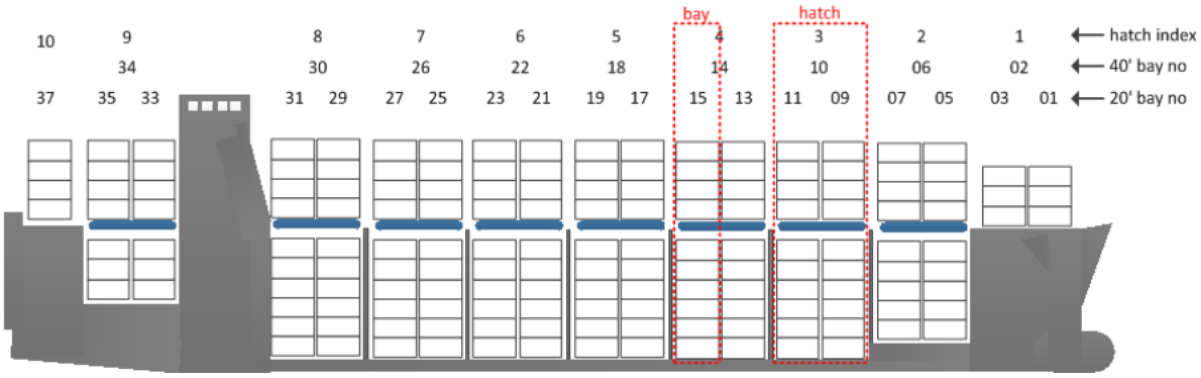


Figure 2.5 Side-view (longitudinal view) of a ship

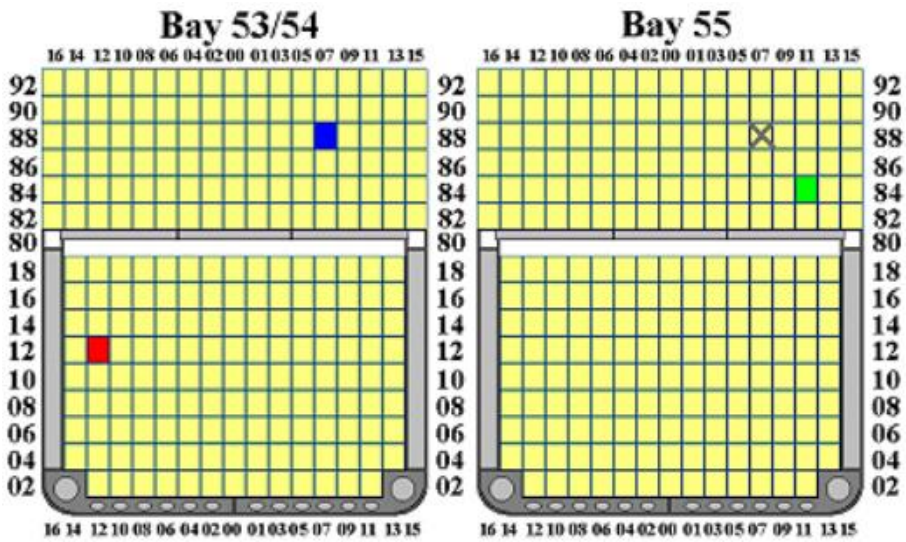


Figure 2.6 Colour-labelled containers in a bay plan. *Source*: (GDV, 2012)

Bay-row-tier coordinates are the most widely used system to express container slot positions in a ship. According to this system, the positions of coloured containers in Figure 2.6 can be expressed as follows: Note that although this system is prevalent, other numbering systems also exist (GDV, 2012).

- 20′ container in red: 531212
- 40′ container in blue: 540788
- 20′ container in green: 551184

There are usually two or three hatch covers (which depend on the size of a hatch) per hatch (H. Zhang & Kim, 2009). These are large steel plates that separate above-deck and below-deck storage. A typical stack includes up to eight containers above deck and a similar number below. Hatches change the nature of the container handling problem of a ship because the stacks cannot be handled without interruption (Goodchild & Daganzo, 2006). The steel covers inevitably lead to some precedence constraints among different QC tasks as follows:

- All the unloading activities below deck (in the hold) cannot begin until all unloading activities above deck area completed.

- All on-deck loading activities cannot start until all below-deck loading activities are completed.

2.6 Logistics planning and control issues

A container terminal represents a complex system with highly dynamic interactions between the various handling, transportation and storage units, and incomplete knowledge of future events (Günther & Kim, 2006). There are many decision problems related to logistics planning and control issues of seaport container terminals. The decision problems at container terminals are comprehensively reviewed in (Stahlbock & Voß, 2008; Dirk Steenken et al., 2004; I. F. A. Vis & De Koster, 2003). Günther and Kim (2006) classify these problems into three different levels as illustrated in Figure 2.7.

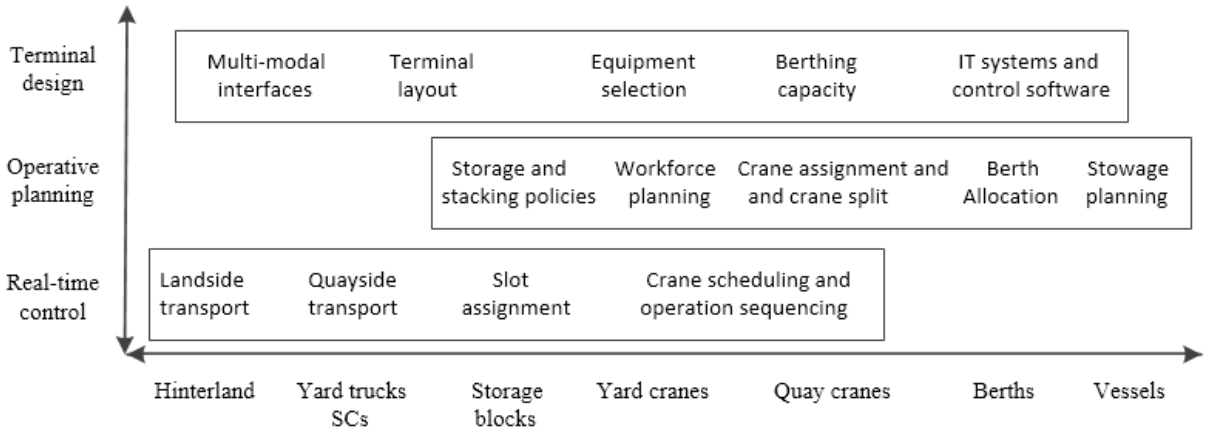


Figure 2.7 Logistics planning and control issues in seaport container terminals. *Source*: (Günther & Kim, 2006)

- Terminal design issues have to be solved by a terminal designer in the initial planning stage of the terminal. Since constructing a new terminal site or expanding the current capacity requires huge investments, this issue involves not only economic analysis and technical feasibility but also interactions with political interest groups.
- Operative planning issues consist of basic planning procedures which prepare the data for operation in such a way as to ensure operational efficiency ahead of a short-term planning horizon, typically several days or weeks but sometimes several hours. In this research, we are particularly interested in the operative planning stage because this is the stage where the planning of daily operation is advanced and the outputs it

generates significantly affect the real-time control stage on a daily basis whereas the terminal design stage typically involves a one-time decision basically in the beginning and occasionally in the middle of terminal history.

- Given the stochastic nature of terminal logistics, real-time control issues are so important that any planned information requires follow-up measures at the time of execution. The decision problems typically involve vehicle scheduling/dispatch and slot/container assignment for the real-time task.

One difficulty in modelling a system like container terminals is that some of the essential information for scheduling and prediction may not be available to the terminal operating system (TOS) at the time of decision making. Containers to and from ship or train, for instance, are generally predictable with respect to the arrival or departure times. Before a ship arrives, it informs the terminal of its expected arrival/departure time and the containers to load/unload so that the terminal can prepare a berthing space for the ship and plan the resources for unloading and loading activities. Containers to and from trains have similar characteristics in terms of the predictability on the receipt and delivery activity.

However, it is often hard to predict containers to and from external trucks. Recent innovations to overcome this inaccuracy of predicting the gate movement, include the *truck appointment system* (TAS) in which external trucks make a reservation to the terminal regarding its arrival times. Hongkong International Terminals Ltd. (HIT), for example, is known to be the first terminal in the world that has implemented the TAS (see Decision Problem D5 in (Murty et al., 2005) for more details about the TAS). In New Zealand, Ports of Auckland (POAL) has introduced an appointment system called *vehicle booking system* (VBS) around 2007. Not only do these systems benefit the port with tempered peaks and troughs of gate traffic, thus enabling better resource utilisation of the terminal; they also benefit the haulier with a much faster vehicle turn-around time by avoiding the normal queues and time spent waiting to be serviced by the terminal. Efficient yard reshuffling strategies under the TAS are studied in (Ku, 2014; Zhao & Goodchild, 2010).

2.7 Terminal operating system

The subsystems illustrated in Figure 2.8 can be addressed in TOS based on the physical area where container flow takes place. The *transfer system* plays an important role in interconnecting between each of the subsystems:

- Ship-to-shore system
- Transfer system
- Storage system
- Delivery and Receipt (typically via gate and rail)

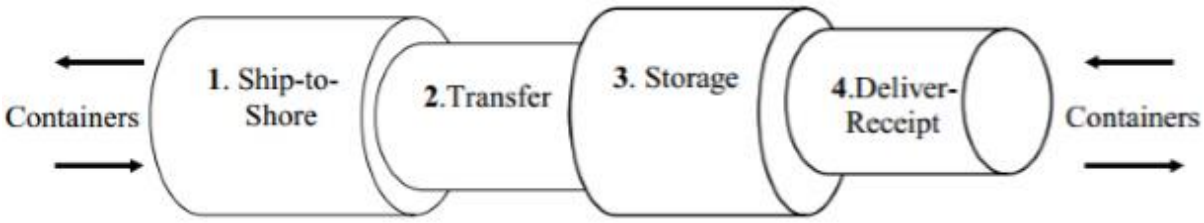


Figure 2.8 container terminal system and the four main subsystems. *Source*: (Henesey, 2006)

It is characteristic to the transfer system that some handling equipment or storage area requires coupled transport between equipment. SC, for example, does not typically require the coupling with other handling equipment to transport a container from one place to another. However, it may need help from empty handlers when it needs to transport a container to/from an empty depot, where empty containers typically grouped by the same shipping line are stacked together without enough space between lanes. For this type of transport, a chain of jobs should be dispatched to each handling equipment sequentially, firstly for an empty handler to move the container from the storage stack to the interchange area and then, as soon as the empty handler drops the container in the interchange area, for a SC to move the container from the interchange area to the destination, either to another location in terminal area or onto a chassis of a truck. YT, on the other hand, is the handling equipment that typically requires a lift-on and lift-off by other handling equipment.

Each subsystem in a container terminal has its unique operational characteristics depending on its configuration, so managing such a system optimally is very complicated. For this

reason, analytical solutions usually tackle each subsystem separately and try to restrict interactions between subsystems even though it is true that pursuing an optimum in one area does not guarantee the global optimum in the entire terminal. For example, Lim (1998) and Y.-M. Park and Kim (2003) studied the berth allocation to ships; Daganzo (1989) and Peterkofsky and Daganzo (1990) analysed the allocation of QCs to ships and the ships' sections; (K. H. Kim & H. B. Kim, 1999; Kim & Park, 2003; Kim, Park, & Ryu, 2000; Taleb-Ibrahimi, De Castilho, & Daganzo, 1993; C. Zhang et al., 2003) focused on storage and stacking models; K. H. Kim and K. Y. Kim (1999) and K. Y. Kim and K. H. Kim (1999) formulated the routing of SCs for loading export containers; Kim, Kang, and Ryu (2004) and Ryu, Kim, Lee, and Park (2001) studied stowage planning and sequencing; Javanshir and Seyedalizadeh Ganji (2010), D.-H. Lee, Cao, and Meng (2007) and Murty (2007) discussed yard crane pooling and scheduling policy for optimisation; Froyland, Koch, Megow, Duane, and Wren (2008) focused on the optimisation for the landside operation.

Motivated by auction theory in economics where system-wide costs are minimised, a holistic approach to TOS can be found in a multi-agent model proposed by (Henesey, 2006). In this work, the author suggested a market-based system where the agents – ship agent, berth agent, yard agent and gate agent – are provided with individual goals and through their interactions with other agents in an auction, a control of the container terminal system can be achieved. The literature of conceptualising the agent model approach to TOS can be traced back to (Degano & Pellegrino, 2002; Gambardella, Rizzoli, & Zaffalon, 1998; Rebollo, Julian, Carrascosa, & Botti, 2000; Thurston & Hu, 2002). Note that these holistic approaches are typically applicable in a simulation study.

Shipping lines, main customers of container terminals, are ultimately interested in decreasing ship turn-around time at a terminal. This particular interest of main customers lets most terminal operators put the highest priority towards the seaside operation among other subsystems in terminal area. The high priority over the seaside operation sometimes results in highly congested gate traffic such as a very long queue of external trucks within or outside a terminal especially when there is not enough yard equipment to handle both seaside and gate operations smoothly. Sometimes, it pushes the terminal operator to invest heavily on the hardware highlighted with advanced technologies such as twin-lift or tandem-lift operation;

simultaneous hoisting, trolley travelling and gantry travel; whichever may reduce cycle times of QCs. It also requires the development of sophisticated decision support systems as larger container ships capable of carrying from 8,000 TEU in 2001 (Meersmans & Dekker, 2001) up to 19,224 TEU in 2015 (Wikipedia, n.d.) have emerged and many terminals are now working at, or close to, capacity with increased trade volume.

2.8 Crane double cycling strategy

As mentioned in (Goodchild & Daganzo, 2006), the crane double cycling operation strategy is regarded as a low-cost strategy to boost the productivity of vessel operation. Double cycling in general refers to a QC operation method to unload a container in the same cycle as a loading operation, thus reducing the empty trolley movement of the QC spreader and hypothetically doubling the number of containers transported in one QC cycle. It also reduces the empty trip of horizontal transport vehicles (SC or internal truck), which means as soon as an internal carrier is released from the laden unit, it is to be engaged to the next lading unit while the QC is running on alternating loading/unloading cycle. A simple comparison between single cycling Figure 2.9a) and 2.9b) and double cycling Figure 2.9c) is depicted as follows.

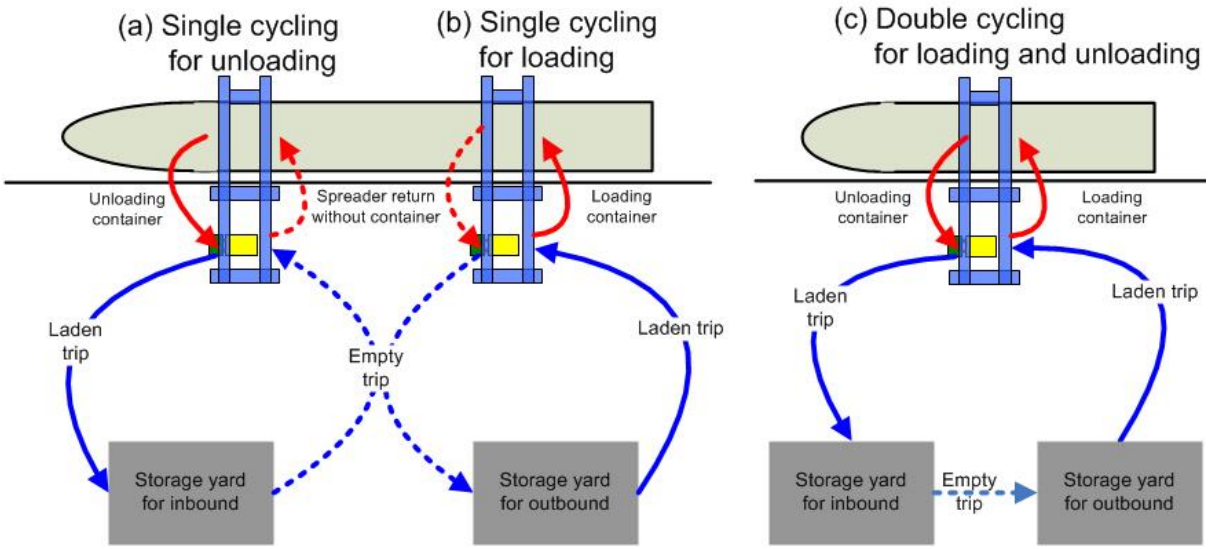


Figure 2.9 Comparison between single cycling and double cycling

Strictly speaking, however, by double cycling two slightly different operation methods can be meant as illustrated in Figure 2.10. In the broadest sense, the double cycling may even refer to any transport cycle within a terminal by which the destination of a laden trip is linked to another container to load near the destination. For example, a container unloaded from a train can be transported by an internal carrier to the storage yard where the carrier finds another container to be transported back to the train for loading. For convenience, we consider that *double cyclin*g in this thesis refers to the double cycling in seaside operation. Also, the term *single-QC* or *multi-QC* can be used for distinction where needed.

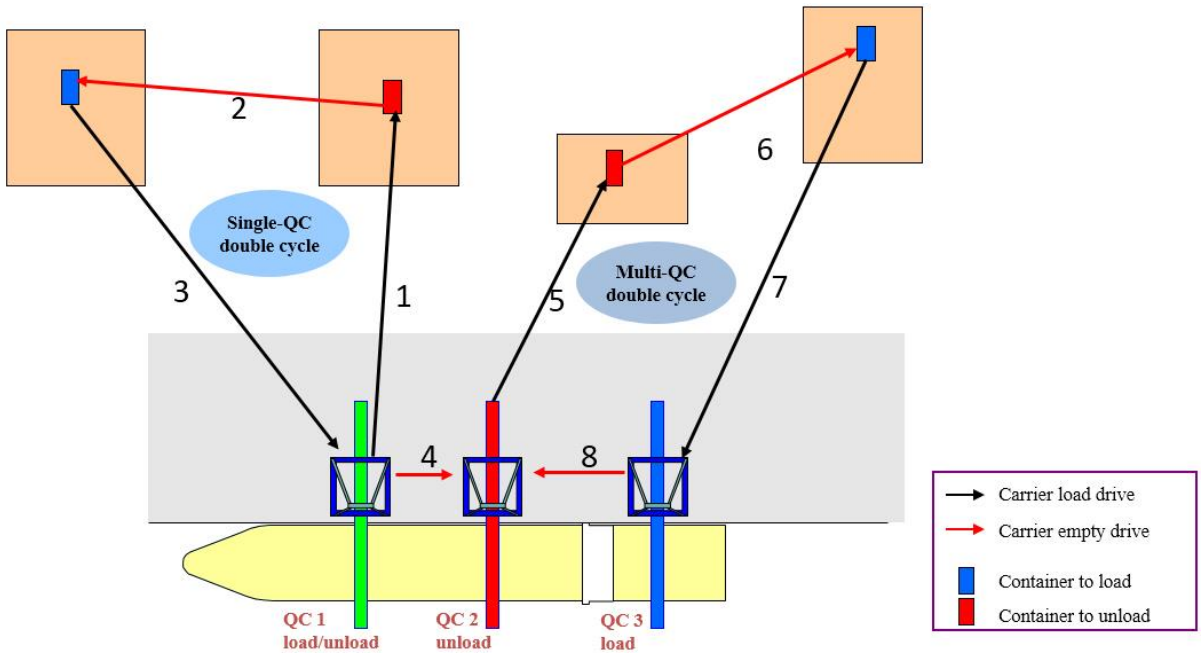


Figure 2.10 Single-QC vs multi-QC double cycling mode

The crane double cycling operation is not the method that is applicable at any time during the vessel operation. There are some preconditions to be satisfied. First, the QC should be equipped with a twin-lift spreader. If it is not, the bays dedicated only to either 20′ or 40′ bays are appropriate bays for double cycling. If a bay, i.e., hatch, is mixed with 20′ and 40′ containers, 20′ containers are typically stowed below 40′ containers for safety reasons. Inside a ship, containers stacked in the vertical direction are linked together by twist-locks (or cones) on four corners. The four corners of a 40′ unit on top of two 20′ units can be adequately

linked by twist-locks, but the opposite can fix only two corners of each 20′ container on either side. This may cause containers to digress while the ship is sailing. Therefore, engaging double cycling under this condition may easily end up with either an accidental hit by a cell guide in the hold or an excessive QC moving time.

3 Seaside scheduling problems

The problem of the seaside planning and operation can be decomposed into sub-problems as is the case with the whole terminal system with respect to its subsystems. The categorisation below focuses on the scheduling aspect given the tasks for seaside operation.

- Berth Allocation Problem (BAP)
- Quay Crane Assignment Problem (QCAP); also known as Crane Split Problem
- Quay Crane Scheduling Problem (QCSP)

Under various terminal configurations, researchers have produced a number of different optimisation models for the above sub-problems. In a procedural approach, the berth schedule can be first constructed by the shipping lines' requests, taking into account the vessel calling schedule, preferred berth location, terminal resource availability and other environmental factors such as tide and weather. Then, the subsequent problems of QCAP and QCSP have to use all available resources to meet the objective and the constraints of the already determined berth schedule. If a model is constructed this way, then it becomes a hierarchical decomposition model among the sub-problems. However, the duration of vessels' berthing at the terminal depends on the number of crane hours allocated to the vessel. If the number of crane hours allocated to a vessel increases, the duration of the vessel's berthing is likely to be reduced. Therefore, a hierarchical decomposition model between berth scheduling and QC split may not be useful in practice. A brief introduction to the literature that addresses each of the sub-problems in seaside planning and operation follows in the remainder of this section.

3.1 Berth allocation problem (BAP)

Berth is the most critical resource for determining the capacity of container terminals because the cost of constructing a berth is very high compared to the investment costs for the other facilities in the terminal (Y.-M. Park & Kim, 2003). Ships have different lengths and they arrive at the terminal at different times to be berthed. A berth planner has to decide if a given set of ships can be berthed in the terminal given certain berthing constraints. If the objective of berthing the set of ships is infeasible, the terminal rejects some ships by their importance or requests the adjustment on the calling schedule to the shipping line.

In early studies, Lai and Shih (1992) suggested a heuristic algorithm for berth allocation based on a first-come-first-served (FCFS) rule. However, this may result in some ships' dissatisfaction regarding the service order. Imai, Nagaiwa, and Tat (1997) developed a heuristic algorithm with two objectives: the minimisation of the service time and the minimisation of the dissatisfaction of ships with the order of service. Lim (1998) considered a berth to be a continuous line rather than a collection of discrete segments, and related the BAP to a restricted form of the two-dimensional packing problem, and have shown that the berthing problem is $\mathcal{NP}$-complete by reducing it to the set partitioning problem. An overview of the theory on the $\mathcal{NP}$-completeness is provided later in the research methodology section.

Imai, Nishimura, and Papadimitriou (2001) introduced two types of BAPs: static berth allocation problem (SBAP) and dynamic berth allocation problem (DBAP). The SBAP, which has already been proposed in (Imai et al., 1997), considers only ships that have already arrived at the port while the DBAP takes into account ships that have already arrived as well as those that have not arrived yet, but will arrive at some later time during the planning horizon. In some scheduling studies, the term *dynamic* is used to refer to problems treating some events with unknown time of their occurrence (Imai, Sun, Nishimura, & Papadimitriou, 2005).

According to (Imai et al., 2001), the SBAP may be formulated as a three-dimensional assignment problem and can be reduced to a two-dimensional (or classical) assignment

problem that is polynomially-solvable with the Hungarian method (Papadimitriou & Steiglitz, 1982). However, the DBAP is represented by a three-dimensional assignment problem with some additional constraints, which is not solvable in polynomially-bounded time. Therefore, after formulating the problem as a *mixed integer programming* (MIP), they developed a heuristic based on the *Lagrangian relaxation* (Geoffrion, 1974). Nishimura, Imai, and Papadimitriou (2001) extended the DBAP to consider some physical constraints such as water depth and berth length. They employed genetic algorithms to solve that problem. In (Imai, Nishimura, & Papadimitriou, 2003), the DBAP was extended to take the service priority into account in the objective function. They found that with a Lagrangian relaxation method the problem was reduced to the quadratic assignment problem (QAP), which is shown to be $\mathcal{NP}$-hard (Sahni & Gonzalez, 1976). Consequently, they applied a genetic algorithm approach for their problem. Imai et al. (2005) and Bierwirth and Meisel (2010) provide a summary on the berth layouts that appear in the literature.

- Discrete layout: The quay is partitioned into a number of sections, called berths. Only one vessel can be served at any single berth at a time. The partitioning can either follow the physical construction of the quay (Figure 3.1a) or be logically separated to make the berth planning more convenient (Figure 3.1b).
- Imai et al. (2005) refer to the model of this layout as the discrete berth allocation problem (BAPD). The literature that applies this layout can be found in (Brown, Lawphongpanich, & Thurman, 1994; Imai et al., 1997; Imai, Nishimura, Hattori, & Papadimitriou, 2007; Imai et al., 2001, 2003; Lai & Shih, 1992).
- Continuous layout: There is no partitioning of the quay, i.e. vessels can berth at arbitrary positions within the boundaries of the quay (Figure 3.1c). The problem class of this layout resembles the two dimensional cutting-stock problem which requires cutting a plane rectangle into smaller rectangular pieces of given sizes and values to maximise the sum of the values of the pieces cut (Christofides & Whitlock, 1977). A ship in service at a berth can be shown by a rectangle in a time-space chart; therefore efficient berth usage is a sort of packing “ship rectangles” into berth-time availability as a box (Imai et al., 2003).
- Imai et al. (2005) refer to the model of this layout as the continuous berth allocation problem. Since the two dimensional cutting-stock problem is known to be $\mathcal{NP}$-hard

in the strong sense (Martello & Vigo, 1998), the literature that formulates a model with this layout typically suggests heuristic solutions to solve the problem. They can be found in (Imai et al., 2005; Kim & Moon, 2003; Lim, 1998; K. T. Park & Kim, 2002).

- Hybrid layout: As in the discrete case, the quay is partitioned into berths, but large vessels may occupy more than one berth (Figure 3.1d) while small vessels may share a berth (Figure 3.1e). The world's first indented berth (Figure 3.1f) is located at the Amsterdam Container Terminal, which was planned to double the productivity due to the berth's capability of serving vessels from both sides (Stahlbock & Voß, 2008). Yet, the respective results do not appear in the literature. Imai et al. (2007) address the modelling problem for the indented berth.

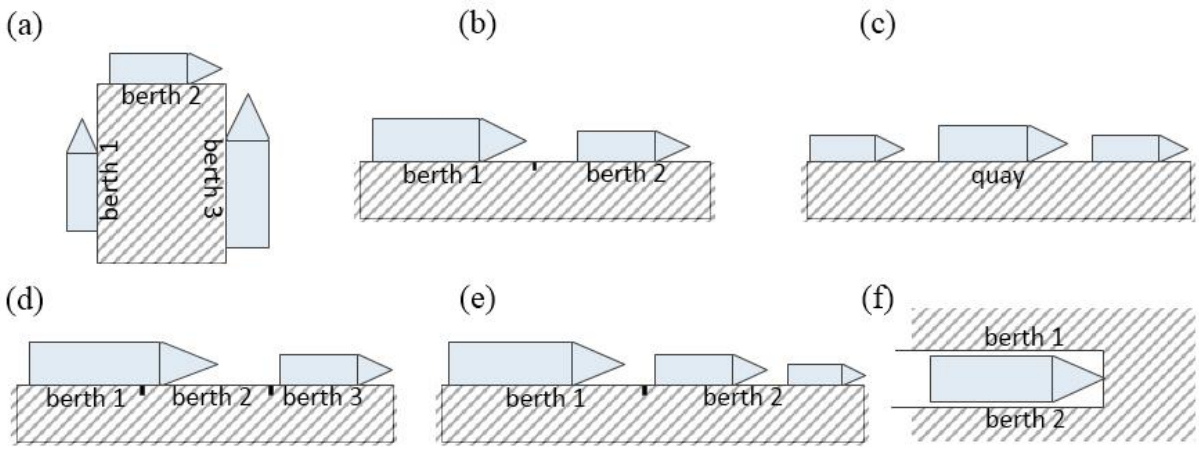


Figure 3.1 Berth layouts

In general, the goal of berth planning is to accommodate as many vessel schedules as possible within a tolerable range of the requested schedules. Therefore, the objective of models in the existing literature is typically to minimise the sum of the waiting and handling times of ships arriving to the terminal. From a practical point of view, the handling time of each vessel is highly linked to the total sum of shore to yard distances for all containers to be unloaded and loaded. From this point of view, the distance for a container to be loaded can be calculated given the possible berthing position of the vessel and the yard position for the container to be loaded. However, it cannot be guaranteed where the yard positions will be for containers to be unloaded from the vessel although storage area for import can be planned in advance at an abstract level. The actual unloading of an individual container is a future event, and the

competition for planned positions among different groups of containers will become very high over time, especially when the terminal is congested. Therefore, a more sophisticated berth planning requires yard planning information for containers to be unloaded as its input.

Due to this complexity, most literature that has been reviewed assumes that the handling time of a vessel is independent of the berthing location. Imai et al. (2005) appear to be one of such literature that attempts to consider the distance factor to and from the yard storage. However, instead of modelling the total sum of containers' travel distance, their approach appears to model the proportional, i.e., linear, increase of the handling time by the distance deviation from the best berth location. It may be an appropriate approach which can eventually simplify the problem domain, but further research needs to address the factor of container travel distances in the BAP model.

3.2 Quay crane assignment problem (QCAP)

The volume of containers to be unloaded and loaded is known in advance for each vessel included in the berth plan. Therefore, the step after the berth allocation is to assign QCs to vessels such that all required container handlings per vessel can be fulfilled within the vessel stay at the terminal. This step is also called *crane split* (D. Steenken, Winter, & Zimmermann, 2001).

Figure 3.2 depicts a simple example of berth window with QCs assigned to each segment of the operation time for each vessel where the number in each cell of the vessel represents the index of each QC assigned to the vessel during the corresponding time. In practice, each segment of the operation time corresponds to either hour or shift (a number of hours). In this example, QCs 1–3 are assigned to vessel A, and QCs 3–5 are assigned to vessel B. Here, the workloads of crane 3 are split between both vessels at the shift 3 in day 1. Likewise, the workloads of crane 3 are split between vessel B and vessel C at the shift 2 in day 2.

To reduce the complexity of the crane split model, some studies assume that the number of cranes assigned to each vessel is unchangeable throughout the entire stay of the vessel, but the variable-in-time crane assignment scheme illustrated here is more realistic and

appropriate for the general QCAP model. Note that the assumption for the time-invariant assignment can have its own merit in specific circumstances. A berth layout like Figure 3.1a) serves a typical example of these circumstances.

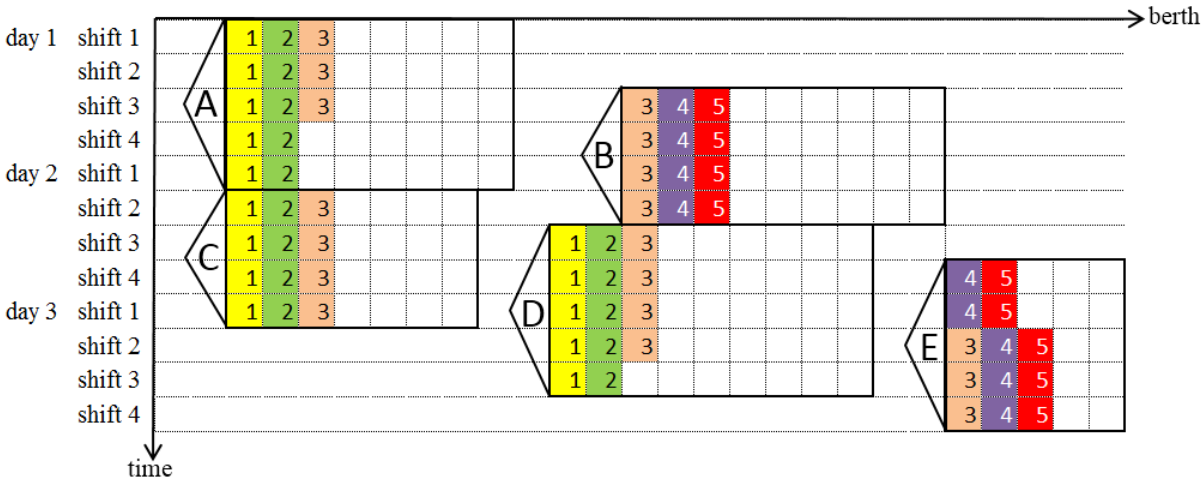


Figure 3.2 An illustration of QC assignments on the berth schedule chart

The assumptions reflecting the real-world constraints have evolved as more research comes to reflect such constraints. The crane scheduling model in (Daganzo, 1989), for example, has well been cited as an early study for this problem. In that study, the berth allocation is assumed to be the static berth allocation, i.e., no additional ships arriving during the planning horizon, and the ships are assumed to be partitioned into bays (referred to as *hold* in that paper). The granularity of a task is defined by each ship bay, therefore no precedence constraints are considered between groups of containers in the same bay. This model, however, does not take into account physical conflicts between QCs. Peterkofsky and Daganzo (1990) assume the similar environment as in (Daganzo, 1989), and a *branch-and-bound* method is employed for the objective of minimising the ship delay cost.

After about a decade's pause in the research interest into this topic, models with more realistic assumptions begin to appear in the literature: cranes' non-crossing (D.-H. Lee & Chen, 2010; D.-H. Lee, Wang, & Miao, 2008b; Lim, Rodrigues, & Xu, 2007; Zhu & Lim, 2006), spatial constraints (Lim, Rodrigues, Xiao, & Zhu, 2004), etc. are some of those improvements. More recent studies tend to integrate the QCAP and the QCSP together, and consequently, the QCAP alone is becoming of less importance. Y.-M. Park and Kim (2003),

for example, decompose the berth and crane scheduling procedure into two phases. In the first phase, the berthing position, arrival time of vessel, and the number of cranes assigned to each vessel are determined. In the second phase, the detailed operating schedule for each crane is constructed with the objective of minimising the number of setups. After formulating an *integer programming* (IP) model, the subgradient optimisation technique with Lagrangian relaxation is applied to the first phase, and the *dynamic programming* (DP) technique to the second phase.

3.3 Quay crane scheduling problem (QCSP)

In the QCSP, the objective in general is to find a working schedule of QCs for the loading and unloading tasks of a vessel such that overall vessel handling time is minimised. Constraints such as non-crossing between cranes, ensuring safety rules, and keeping precedence among tasks may be involved. Tasks in the QCSP are related to the granularity of the workload. According to the summary by (Bierwirth & Meisel, 2010), tasks can be defined on the basis of bay areas or single bays or on the basis of container stacks, container groups or individual containers as in Figure 3.3.

- Bay areas: A task consists of all loading and unloading operations of containers within a connected area of the bays
- Bays: A task consists of all loading and unloading operations within a bay
- Stacks: A task consists of all loading and unloading operations within a stack
- Container groups: A task consists of a group of containers to be processed in adjacent slots of a bay. Grouped containers usually share common characteristics such as the same destination, size and type, weight class or the like.
- Individual containers: A task consists of the loading and unloading of a single container.

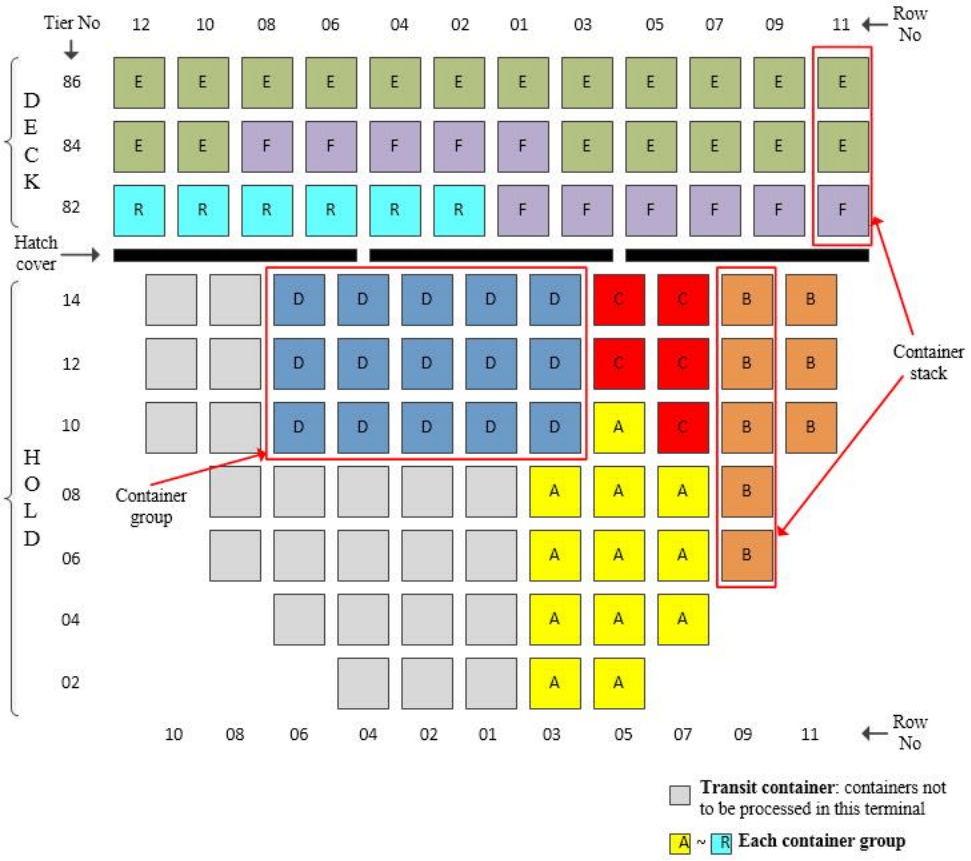


Figure 3.3 Bay-wise (cross-sectional) view of a ship

The concept of dividing the workload of a vessel into tasks is to serve each task exclusively by one crane. The smaller the granularity of the workload, the more balanced workload distribution among the cranes is likely to take place. In general, reducing the granularity will improve crane scheduling at the expense of the increased problem complexity. With hundreds to thousands of container to be handled in a ship, however, solving the QCSP by the range of individual containers as tasks may be intractable (Bierwirth & Meisel, 2010). Therefore, among the above classes, the task granularity based on container groups or bay levels is more practical, as can be seen from the widely studied literature.

Figure 3.4 shows a practical example of showing QC schedules in which two cranes, A and B, are scheduled in a ship. QC A starts the operation from the stern of the ship, progresses towards the middle of the ship and finishes its operation at bay 26. QC B starts the operation from the middle of the ship, progresses towards the stem of the ship and finishes the operation. In this particular example, every bay is dedicated to a specific crane except for the bay 26. The unloading operation on the deck of bay 26 is shared between both QCs. The

decision of dividing the workload of bay 26 may be based on the time estimation given the number of containers to be handled. As the operation progresses towards the end, however, the terminal operator will detect any unbalance of the remaining workloads and may reschedule the working sequences of each QC. The classification of this QC working scheme with respect to the task granularity was a little ambiguous in the literature. In one approach, the task granularity of this scheme was viewed as an individual container because there is a split in the workloads of a bay but the workloads to each QC by the split are left to be decided. In another approach, the task granularity was viewed as a bay area and the preemptive schedule is assumed to accommodate the split of workloads. We view the latter approach to be more practical, thus we apply it to the model studied in **Paper I**.

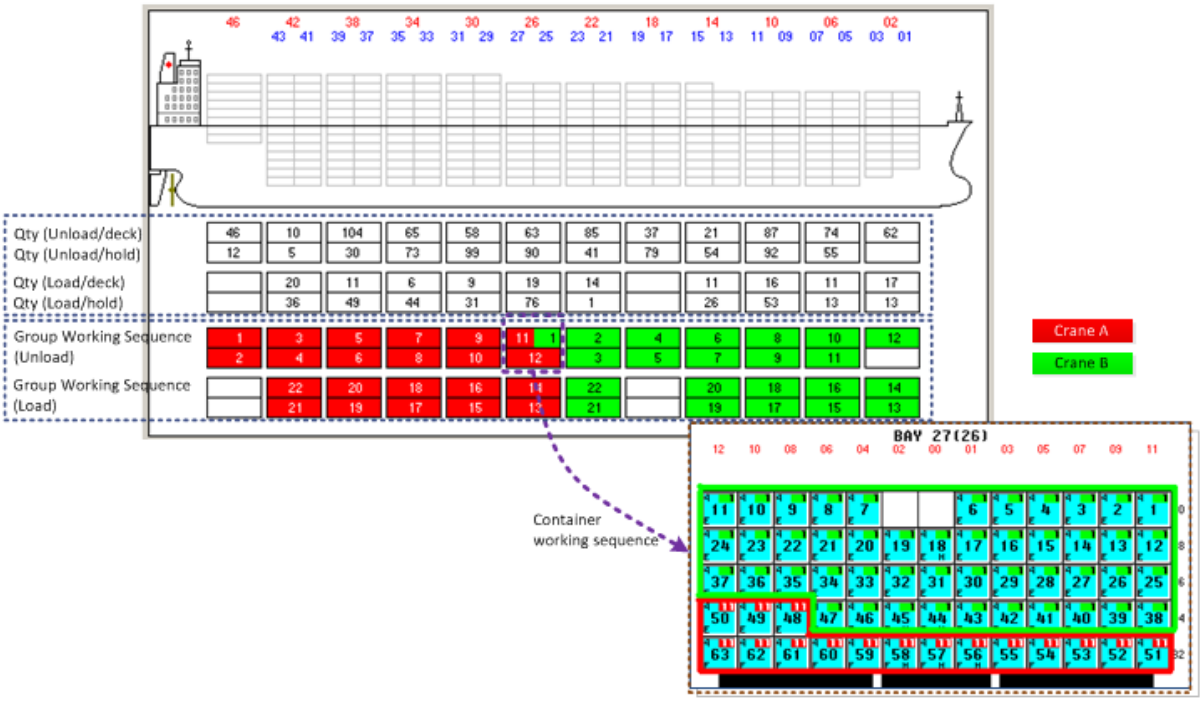


Figure 3.4 A practical example of QC scheduling

In the QCSP, the mainstream research with regard to the task granularity often considers three types of task granularity: container groups, complete bays and bay areas. The QCSP with container groups is first formulated by (Kim & Park, 2004), which consider crane interference constraints, QC travel times and precedence relations among tasks. In their study, two types of crane interference constraints are considered: *safety margin* and *non-crossing*. QCs are positioned on a rail track, thus they can only move laterally. This means that one QC

cannot cross another at all times (*non-crossing constraint*). Also, at all times any two cranes cannot approach each other within the safety margin (*safety constraint*), usually parameterised by a prescribed number of bays.

The research stream pioneered by (Kim & Park, 2004) is further refined by (Bierwirth & Meisel, 2009; Jin & Li, 2011; Moccia, Cordeau, Gaudioso, & Laporte, 2006; Sammarra, Cordeau, Laporte, & Monaco, 2007). The QCSP with complete bays is first formulated by (Daganzo, 1989) and subsequently studied by (Peterkofsky & Daganzo, 1990) and, with over a decade's pause, by (D.-H. Lee, Wang, & Miao, 2008a; D.-H. Lee et al., 2008b; Lim et al., 2007; J. Liu, Wan, & Wang, 2006; Zhu & Lim, 2006). The studies with bay areas are found in (Lim, Rodrigues, Xiao, et al., 2004; Lu, Han, Xi, & Erera, 2012). More thorough review of the literature in this topic is provided in **Paper I**.

Meisel and Bierwirth (2011b) suggest a unified framework for evaluating the algorithms discussed in the various QCSP classes. Two findings about the model evaluation from their study are worth noting: first, vessel handling times obtained for instances under the container group model are no worse than vessel handling times obtained for instances under the other models. Second, vessel handling times obtained for instances under the complete bay model are no worse than vessel handling times observed for instances under the bay area model. The findings will enable us to apply the upper bound for heuristics when solving the most complex model among the three models.

3.4 Quay crane double cycling problem (QCDCP)

In contrast to the potential economic benefits that crane double cycling operation can bring, the double cycling operation has received little attention in both academia and industry. In scholarly journals, the earliest literature modelling double cycling method appears in 2004 (Goodchild & Daganzo, 2004). Before this study, Bendall and Stent (1996) introduced the productivity gains from hatchless ships when double cycling was used, in which the vessel turn-around times fell on average 25% compared with an equivalent TEU hatch-cover vessel. However, the vessel considered in that study was small-sized (of less than 1,000 TEUs), and their work did not address the operational strategy for double cycling to be worked. In

industry, terminal operators have often discussed this strategy for their terminal operation but little has been implemented except for some trial runs (LinkedIn Discussion).

There can be three types of directions for determining the stack working sequence of double cycling: starboard (right-side of a ship) to portside (left-side of a ship), portside to starboard, and the way that can maximise the number of double cycling (which will likely result in a zigzag pattern for stack working sequence). The way that can maximise the number of double cycling may be academically meaningful, but it is difficult for field workers to understand and follow the work flow. Thus, sufficient preparation and training should be done before its application (J.-H. Song, 2007). Goodchild (2005) has also noted this situation and termed the following strategy the *proximal stack strategy* as opposed to the *greedy strategy*, which only pursues the minimal number of cycles by the double cycling method:

1) the crane unloads all containers in the stack closest to the shore, then all containers in sequential stacks until all stacks in the bay have been unloaded;
2) the crane loads stacks using the same ordering. Loading can start in a stack as soon as the stack is empty or contains only containers that should not be unloaded at this port. Once loading has begun in a stack it is continuously loaded until complete.

In (Goodchild, 2005), the mathematic model for finding the optimal cycle by double cycling is solvable in a polynomial time using Johnson's rule (S. M. Johnson, 1954). Even though Goodchild's model for the double-cycling involves only a single QC, the study relates the problem to a two-machine flow shop problem where one job corresponds to one stack and each job has two operations: an unloading operation that must be done first, and a subsequent loading operation. For the formulation of this work, refer to Appendix in this chapter.

J.-H. Song (2007) notes that the double cycling method suggested by (Goodchild, 2005) would incur blocking delay frequently. A ship loading can begin as soon as a stack in a ship becomes empty. The blocking delay is, according to (Goodchild & Daganzo, 2004), the situation where while double cycling, the loading operation must wait if there is no empty stack available. Noting its inconvenience and inefficiency to the seaside operation, he presents a formula to find the optimal starting sequence for double cycling in such a way as not to incur the blocking delay while double cycling. The developed simulator has proved the

validity of his formula. H. Zhang and Kim (2009) extend the work by (Goodchild, 2005) to the case of multiple hatches. In their work, they decompose the whole QC scheduling into two parts: intra-stage optimisation (sequencing all the stacks in one hatch) and inter-stage optimisation (sequencing all the hatches). For the intra-stage optimisation, they apply Johnson's rule as is done by Goodchild. For the inter-stage optimisation, they suggest heuristics based on the gap-based neighbourhood local search. The experimental results show that the proposed method outperforms the real schedules constructed by human planners. The model, however, does not consider practical constraints such as crane interference, for which counter examples are shown in **Paper II**. Research following the configuration similar to that of (H. Zhang & Kim, 2009) can be found in (He, Wang, & Zheng, 2011; D. Wang, Li, & Wang, 2011). C.-Y. Lee, Liu, and Chu (2014) consider hatch covers in the model, which they refer to as the general quay crane double cycling problem (QCDCP). They then relate the problem to a flow shop scheduling problem with series-parallel precedence constraints, which is solvable optimally in polynomial time. **Paper III** addresses the multi-QC double cycling problem (MQCDCP) with the mathematical formulations including constraints such as crane interference, QC travel times, and so on.

4 Aims and objectives

Even though it seems trivial that the crane double cycling operation improves the productivity of vessel operation and yields some economic benefits, its implementation to day-to-day operations for container terminals has been considered difficult. It is said that a container terminal is primarily characterised by four elements: *facility*, *process*, *operating system* and *manpower*. They should be developed together by keeping up with each other, but sometimes the balance among them can easily be broken. This imbalance typically happens when an advanced technology in either facility or operating system is introduced. A typical example would be when a new feature in the equipment (e.g. twin-lift or tandem-lift**)** is introduced to the terminal; when a terminal operating system is being replaced with a process innovation (e.g. equipment pooling or double-cycling). When these changes occur, the process change should be followed and the manpower should be trained to carry out the expected performance.

Therefore, it is quite reasonable to expect that a practical implementation of double cycling, which this research is aiming for, will first affect the operation by requiring the operating system to support double cycling sequence for QC. But in the end it is the manpower that actually must execute the process. Consequently, a modelling problem affecting the operational change will take this factor into account. An editor in an industry journal has once commented about the issue with double cycling in the *LinkedIn* discussion: (Avery, 2011)

> *Some terminals are indeed using double cycling, but what every academic paper I've ever read on the subject fails to recognise is the labour environment in a marine terminal. Port operators are not free to organise the actual work according to what is mathematically optimal. Rules like shift hours, gang sizes, division of labour and work areas between import and export containers have to be considered as these will ultimately determine what actual financial benefit a terminal operator can derive from double cycling.*

As a scientific research, the sequencing problem in double cycling operation cycles was initiated by (Goodchild, 2005). The limitation of this model is that it deals with only a single hatch and does not consider blocking delays. Still in the single hatch scenario, J.-H. Song (2007) provided a formula to determine the starting sequence for double cycling such that blocking delays do not incur. H. Zhang and Kim (2009) proposed a double cycling scenario in multiple hatches, but interference constraints were not considered at all. C.-Y. Lee et al. (2014) presented a general model for double cycling in the sense that the model considers the hatch covers. Still, their work is limited to a single hatch.

Inspired by the studies of the above authors, and also motivated to fill the gap in the current research, our research aims to develop a practical double cycling model considering the multi-QC operation. In **Paper III**, we therefore propose the objective function, where terminal users can weigh the importance between the makespan and the degree of double cycling operation.

5 Research methodology

Research methodology provides tools and techniques for identifying and solving research questions. It does so by enabling researchers to obtain a body of knowledge for a specific subject and investigate important topics not easily answered previously. According to (Yin, 2014), researchers can choose to apply more than one methodology while doing their research. The overall idea is that different research methods can serve complementary functions in answering a research question. As a computational study, the research methodology adopted in this thesis combines two methods: literature review and optimisation technique.

5.1 Literature review

A review of the existing literature is performed on journal articles, conference proceedings, technical reports and online forums related to the subject area. The purpose is to obtain a firm understanding of what has been done in the subject area and identify gaps to be filled in. Let us recall the title of the thesis, *Marine Terminal Logistics: Crane Double Cycling Models*. Three different levels can be considered with respect to the scope of this research.

In the broadest level of the research, we may think of research that addresses overall terminal logistics. We do not consider this level of broadness for the scope of this thesis. For such a broad view of the subject, it is recommended to refer to the survey literature such as (Stahlbock & Voß, 2008; Dirk Steenken et al., 2004). Instead, the thesis addresses the seaside planning and operational aspect in the mid-level broadness, the literature review for which is already provided in the previous section (Section 3). In the finest level, the research addresses the crane scheduling problem, the literature review for which is provided in **Paper I**. More specifically, the seaside operation strategy termed double cycling operation is investigated in **Paper II** and is modelled in **Paper III**, thus relevant reviews of the literature in that topic are provided in **Paper II** and in **Paper III**.

5.2 Optimisation techniques

Mathematical optimisation can alternatively be called as mathematical programming. This technique often consists of various fields of disciplines such as applied mathematics, economics, computer science and operations research. The objective of this technique is to establish whether an optimal solution exists and then to find one, or perhaps all, optimal solutions. The simplest form of an optimisation problem consists of maximising or minimising a real function by systematically choosing input values from an allowed set of domains and computing the value of the objective function. In an applied context, it is convenient to think of an optimisation problem as a model of decision making in which the allowed set of domains represents the set of all permissible decisions and the objective function yields a utility or a penalty computed from the choice of feasible decisions. Applications of this model about the real world are relevant to various branches of engineering, business, and sciences. Thus, in this thesis it is natural to assume that domain-specific knowledge in logistics and transportation can be involved to model the entities behaving in the practice of terminal logistics.

Decision values in an optimisation problem can be either *continuous* (real number) or *discrete* (integer) depending on the context of a model. An IP problem is a mathematical programming problem in which all of the input values are required to be *integers*, whereas an MIP problem is a mathematical programming problem in which at least one, but not all, of the input values are required to be *integers*. An MIP model for the QCSP is formulated in **Paper I**, and an IP model for the QCDCP is formulated in **Paper III**.

In the following, we select several facets of optimisation technique for the description of the research methods relevant to this thesis. We start by giving an explanation of the theory of computational complexity as the complexity of IP or MIP problem is $\mathcal{NP}$-complete. We then come back to the optimisation techniques to solve the $\mathcal{NP}$-complete problems posed in the papers of this thesis. Finally, we briefly describe the benchmark suite available for this domain before closing this section.

5.2.1 Computational complexity and the theory of $\mathcal{NP}$-completeness

When evaluating algorithms, how long the algorithm takes to resolve a problem is of interest. Among several ways to evaluate an algorithm, one obvious way is to code the algorithm in a specific computer language, to execute it on a data set, and to record its execution time on a test PC. This approach only tells us how long a particular implementation of the algorithm takes to resolve a particular instance on a particular machine. Generalising this approach to other implementations, other instances and other machines would be difficult. This is where complexity theory comes in as an attempt to theorise the measurement of time and space required to solve a problem. Apparently, we need some computation models to underlie complexity theory on them.

In 1936, Alan Turing developed his theoretical computation model, which is now known as the "*Turing machine*". He based his model on how he perceived mathematicians think. With the birth of digital computers in the 1940s and 1950s, the Turing machine model proved itself as the right theoretical model for computation. Quickly though, it turns out that the basic Turing machine model fails to account for the amount of time or memory needed by a computer, a critical issue even nowadays. The key idea to measure time and space as a function of the input size came in the early 1960s, and thus the field for the study of computational complexity was born (Fortnow & Homer, 2003). In understanding complexity theory, it is essential to note the complexity class of a problem: class $\mathcal{P}$ and class $\mathcal{NP}$.

Definition 1.1 The class $\mathcal{P}$ of computational problems is the set of decision problems for which an algorithm exists which can be carried out by some deterministic Turing machine in polynomial time.

For clarity, precise definitions for the terms used in **Definition 1.1** are given below:

- A decision problem is a question in some system with a yes-or-no answer, depending on the values of some input parameters.
- A deterministic Turing machine is analogous to our personal computer, which at any given step during computation can make only one move at once. An example of problems in the class $\mathcal{P}$ is the problem of determining if a number is prime (Agrawal,

Kayal, & Saxena, 2004). Clearly, it is a decision problem that requires yes-or-no answer.

- Polynomial time refers to a time complexity that quantifies the amount of time taken by an algorithm, where the number of steps required to complete the algorithm for an input size n is given by O(n^k) for some nonnegative integer k.

The class $\mathcal{P}$ belongs to a larger class of problems called the class $\mathcal{NP}$, which is defined in **Definition 1.2**.

Definition 1.2 The class $\mathcal{NP}$ is the set of decision problems for which if a solution to the problem is given, there exists a proof system such that the proofs are verifiable by some deterministic Turing machine in polynomial time.

The term $\mathcal{NP}$ comes from *nondeterministic polynomial time* and is derived from an alternative characterisation in **Theorem 1.1** (for the proof, see chapter 7 in (Sipser, 2006)) by using nondeterministic Turing machine:

Theorem 1.1 A problem is in $\mathcal{NP}$ if and only if it is solvable by some nondeterministic Turing machine in polynomial time.

A nondeterministic Turing machine is like a conventional Turing machine, but at any given step during computation it can make more than one move at once. As computation continues, a nondeterministic Turing machine can split as many times as necessary, calculating a vast number of possibilities simultaneously. It is only a fictitious computer since there is no such computer in the real world that qualifies as a true nondeterministic Turing machine. Two examples of the problems in $\mathcal{NP}$ are given below, corresponding to **Definition 1.2** and **Theorem 1.1**, respectively.

- Example 1: We simply require that any "yes" answer is verifiable in polynomial time. Consider the subset sum problem where the task is to decide whether a subset with sum zero exists among a set of integers. Assume that the input integers are given as {-7, -4, 2, 3, 5}. In this example, the answer is "yes" because there exists a solution {-7, 2, 5} that corresponds to $(-7) + 2 + 5 = 0$. The solution is easily verifiable because the evaluation of a solution with n members takes only n steps in the computation

which in big-O notation is represented by $O(n)$ and hence can be verified in polynomial time.

- Example 2: A nondeterministic Turing machine can make multiple moves at once. Consider the problem of determining "What integers between 1 and q are prime?" A nondeterministic Turing machine can check every integer between 1 and q simultaneously, in the same amount of computation time that a deterministic Turing machine would take to check just one integer value.

There are several reasons underlying the use of complexity theory. According to (Daskin, 2011), complexity theory (1) defines clearly what solving a problem "efficiently" means, (2) categorises problems into those that can be solved efficiently and those that cannot, and (3) estimates the amount of time (or storage) needed to solve these problems. Using complexity theory, we can evaluate an algorithm whether or not the existing algorithm can resolve the given problem completely with the growth in the problem size. We can also compare algorithms with respect to the time and memory they need, to resolve a given problem. Recognition of the complexity class of a problem is another important help of this theory. Often, the recognised complexity class of a problem determines our future approach we choose to resolve the problem. If, for example, the problem turns out to be $\mathcal{NP}$-complete (which will be defined below), we shift our focus from exact solutions to approximate or so called heuristic approaches (Avazbeigi, 2009).

From a historical point of view, the theory of $\mathcal{NP}$-completeness is typically traced back to Stephen Cook from his paper (S. A. Cook, 1971), which introduced an important subclass of the $\mathcal{NP}$ problems, now known as the $\mathcal{NP}$-complete subclass. It should also be noted that Leonid Levin, then a PhD student in Moscow, proved much the same results at roughly the same time although his paper (Levin, 1973) did not appear until 1973. Over the years, the contemporaneous and independent nature of Levin's accomplishment have come to take precedence over publication dates, and what used to be called "*Cook's Theorem*" is now generally referred to as the "*Cook-Levin Theorem*" (D. S. Johnson, 2012). The theorem states that the Boolean satisfiability problem is $\mathcal{NP}$-complete (See (Garey & Johnson, 1979) or (Papadimitriou & Steiglitz, 1982) for proof). In other words, any problem in the class of $\mathcal{NP}$ can be reduced in polynomial time by a deterministic Turing machine to the problem of

determining whether a Boolean formula is satisfiable. It implies that if an $\mathcal{NP}$-complete problem can be solved in polynomial time, all other $\mathcal{NP}$-complete problems can then be solved in polynomial time, too. The question of whether such an algorithm exists is called the $\mathcal{P}$ versus $\mathcal{NP}$ problem, the result of which has neither been proved nor disproved over the last 40 years. It is one of the seven Millennium Prize Problems selected by the Clay Mathematics Institute in 2000.

Following the theory of $\mathcal{NP}$-completeness, in 1972 Richard Karp published an influential paper, "Reducing among combinatorial problems" (Karp, 1972), which proved several other problems were also $\mathcal{NP}$-complete; thus there is a class of $\mathcal{NP}$-complete problems besides the Boolean satisfiability problem. His paper presented several key methods to prove $\mathcal{NP}$-completeness by reductions from other problems previously shown to be $\mathcal{NP}$-complete. Many of these problems are collected in the seminal text by (Garey & Johnson, 1979), which is one of the most influential publications on the complexity theory. The term $\mathcal{NP}$-hard is used to describe the optimisation versions of the decision problems that are $\mathcal{NP}$-complete (Daskin, 2011).

Next we describe exact methods and heuristic methods as the solution techniques used for the problems in this thesis.

5.2.2 Exact optimisation techniques

When modelling a real-world problem into an optimisation problem, as the first milestone we aim to obtain an exact model that can account for the corresponding optimisation model. There are a few types of exact methods that can be addressed, but broadly, we can group them into *search-based algorithms* and *mathematical programming models*. The solution approach of each optimisation problem can use either way or the combination of both ways.

Branch-and-bound method and integer programming

The branch-and-bound method is in essence a search-based algorithm because by means of state space search, it searches the complete space of solutions for a given problem for the best solution. However, the explicit enumeration is normally impossible due to the exponentially

growing number of potential solutions in the $\mathcal{NP}$-complete problems. By *branching*, it decomposes the current solution space into two or more subspaces. By *bounding*, it searches only parts of the solution space if each subspace is worth to be explored for finding a better solution. Branch and bound is a general algorithm framework for the optimisation world. It is applied routinely to several combinatorial optimisation models, in which exhaustive search is infeasible. Although the term *branch-and-bound* first occurs in the work of (Little, Murty, Sweeney, & Karel, 1963), who invented this algorithm remains debatable. It appears to have three origins, spread out over three years between 1957 and 1960: (Eastman, 1958; Land & Doig, 1960; Markowitz & Manne, 1957). See (W. Cook, 2012) for the historical perspective of the branch-and-bound method.

The branch-and-bound method consists of the repeated application of the branching process and adopting a bounding mechanism to indicate if it is worthwhile to explore any or all of the newly created subspaces. According to (Clausen, 1999), the iteration has three main components: *selection of the node to explore*, *bounding function* and *branching rule*. The sequence of node selection varies according to the adopted search strategy: *depth-first search*, *breath-first search*, *best-first search* and so on. For each node selected, it is checked whether the subspace consists of a single solution, in which case it is compared to the current incumbent keeping the best of these. Otherwise the bounding function for the subspace is calculated and compared to the current incumbent. If there is no hope of the subspace containing the optimal solution, the whole subspace is discarded or stored in the pool of bounded solutions. The search terminates when there are no unexplored parts of the solution space left, and the optimal solution is then the one recorded as the "current incumbent".

For an exemplary application of branch-and-bound method to a mathematical programming model, let us consider a minimisation problem. Note that the case of a maximisation problem can be dealt with reversely especially regarding the bounding directions. We borrow some notations used in (Garfinkel & Nemhauser, 1972). In general, consider the problem (P):

$$(\mathrm{P}): \min_{x \in S} z(x) \tag{1.1}$$

where x is the vector of variables $(x_1, \cdots, x_n)$ over a region of feasible solutions S.

General conditions of the variables, e.g., domains (non-negativity, integrality, binary, etc.) and constraints, usually determine the set of feasible solutions. Suppose the problem (P) is an IP problem, and further suppose a variable x_i is assigned a fractional value t in an optimal solution to the *linear programming* (LP) relaxation. We can then branch by considering separately the subspace having $x_i \leq \lfloor t \rfloor$ and the subspace $x_i \geq \lfloor t \rfloor + 1$ where $\lfloor t \rfloor$ is the largest integer less than or equal to t. Suppose that the enumeration of the branching and bounding procedure is at node j in the search tree. The problem (P_j) at node j can then be defined as:

$$(P_j): \min_{x \in S_j} z(x) \tag{1.2}$$

where S_j ($S_j \subseteq S$) is a region of feasible solutions at node j.

Let

$$z_j^* = \begin{cases} z(x^*(j)) & \text{if } x_j^* \text{ solves } (P_j) \\ \infty & \text{if } S_j = \emptyset \\ -\infty & \text{if } (P_j) \text{is unbounded.} \end{cases}$$

The *bounding* mechanism works by determining a value B such that each solution in a subspace has objective value no smaller (respectively, larger) than B for minimisation (respectively, maximisation) problems. The two *sub-problems*, i.e., (P_j) ^($x_i \leq \lfloor t \rfloor$) and ($P_j$) ^($x_i \geq \lfloor t \rfloor + 1$), created by the *branching* need only be considered for further exploration if their corresponding bound B is smaller (respectively, greater) than the value of the current incumbent for the original problem (P). A lower (respectively, upper) bound $B \leq z_j^*$ (respectively, $B \geq z_j^*$) may be calculated by the relaxation of (1.2). One well-known technique to find a valid lower (respectively, upper) bound B is the LP relaxation, which can be solved optimally using a standard algorithm such as *simplex* or *interior point*. In many cases, however, solving the LP relaxation of the original problem (P) is impractical because (P) often involves extremely large number of variables and/or constraints. Therefore, we need

an alternative technique to find a valid *B*. Lagrangian relaxation described later is one such kind.

Before departing from branch-and-bound method, we note that there is another important approach in the context of solving LP relaxations of IPs. Several researchers have found it beneficial to generate and add additional constraints to the model, prior to its solution, in order to tighten its LP relaxation. Cutting plane methods start with the LP relaxation as the branch-and-bound method does. Once that LP is solved, however, cutting plane methods attempt to add one or more constraints (cuts) to the LP instead of branching. These constraints should be the ones that make the current LP solution infeasible and do not eliminate the optimal integer solution. Once one or more constraints have been added, then the LP is resolved and the process repeated. According to (W. Cook, 2012), the combination of branch-and-bound and cutting planes eventually became the dominant solution procedure in the IP and the combinatorial optimisation problem. The first application of this combination is reported in (Grötschel, Jünger, & Reinelt, 1984). Later the method is applied to the travelling salesman problem (TSP) (Padberg & Rinaldi, 1987, 1991), which have also coined the term *branch-and-cut* for the combination of the two competing algorithms. According to a review on the existing solutions to the TSP by (Hornik & Grün, 2007), a state-of-the-art implementation of the TSP is found in *Concorde TSP Solver* (Applegate, Bixby, Chvátal, & Cook, 2003), which uses the cutting-plane method, iteratively solving LP relaxations of the TSP.

Lagrangian relaxation and subgradient optimisation

The idea of *Lagrangian relaxation* comes from the observation that there are relatively only a small set of side constraints that actually make IP problems difficult to solve. References (Fisher, 1981; Geoffrion, 1974) provide general descriptions of Lagrangian relaxation and surveys of successful applications. By dualising "complicating" side constraints, the original problem is transformed into a Lagrangian problem that is easier to solve and whose optimal value is a lower bound (for minimisation problems) on the optimal value of the original problem. Thus the easier-to-solve Lagrangian problem can be used for the bounding function in a branch-and-bound method. According to (Fisher, 1981), the use of *Lagrangian*

relaxation dates back to the works in (Lorie & Savage, 1955) for capital budgeting, Everett's proposal for generalising Lagrange multipliers (Everett III, 1963) and the philosophically-related device of generating columns by solving an easy combinatorial optimization problem when pricing out in the simplex method (Gilmore & Gomory, 1963). As noted by (Sherali & Myers, 1988), it has been shown to be an effective approach for solving several types of continuous and discrete optimisation problem. Examples of such instances are the TSP by (Held & Karp, 1970, 1971), scheduling algorithms by (Fisher, 1981) and linear assignment problem by (Bazaraa & Sherali, 1981).

In the following, we define the Lagrangian relaxation problem (LR_λ) of the original problem (P) with respect to the constraint set $Ax \geq b$ by introducing a *Lagrange multiplier vector* (or *dual variable*) $\lambda(\geq 0)$ which is weighted to this constraint set and brought into the objective function to give:

$$\begin{aligned} (\mathrm{P}): \quad & \mathrm{Z} = \min\ cx \\ \text{s.t.} \quad & Ax \geq b \\ & Bx \geq d \\ & x \geq 0 \text{ and integral.} \end{aligned} \tag{1.3}$$

$$\begin{aligned} (\mathrm{LR}_\lambda): \quad & \mathrm{Z}_D(\lambda) = \min_{\lambda \geq 0} cx + \lambda(b - Ax) \\ \text{s.t.} \quad & Bx \geq d \\ & x \geq 0 \text{ and integral.} \end{aligned} \tag{1.4}$$

For convenience we assume that the problem (P) is feasible and that the set $X = \{x | Bx \geq d, x \geq 0 \text{ and integeral}\}$ of feasible solutions to the problem (LR_λ) is finite. Then $\mathrm{Z}_D(\lambda)$ is finite for all λ. For any $\lambda \geq 0$, it can be easily shown that $\mathrm{Z}_D(\lambda) \leq \mathrm{Z}$ as $b - Ax \leq 0$. Let Z^* denote the optimal objective value in (P) with an optimal solution x^*, i.e., $cx^* = \mathrm{Z}^*$. Then, we can easily obtain $\mathrm{Z}_D(\lambda) \leq cx^* + \lambda(b - Ax^*) \leq cx^* = \mathrm{Z}^*$. The fact that $\mathrm{Z}_D(\lambda) \leq \mathrm{Z}^*$ allows the problem (LR_λ) to provide a lower bound on the optimal solution to the problem (P). This

enables the problem (LR_λ) to be used as an alternative bounding function in the aforementioned branch-and-bound method.

There are several issues to be highlighted in the use of the Lagrangian relaxation:

- Why did we choose to dualise the set of constraints $Ax \geq b$ when we could equally well have chosen to dualise $Bx \geq d$?

  Theoretically "complicating" constraints are to be chosen to be dualised. However, it sometimes is hard to know in advance which constraints are more complicating than others, in which case some empirical approach may be used to find more complicating constraints.

- How can we find appropriate value for λ?

  Choosing values for λ is of key importance in terms of the quality of the lower bound generated. We are interested in finding the values for λ that give the maximum lower bound, i.e., the lower bound that is as close as possible to the value of the optimal integer solution. This involves finding λ in the Lagrangian dual problem (LD):

$$\text{(LD):} \quad Z_D = \max_{\lambda \in M} Z_D(\lambda) \tag{1.5}$$

  where the constraint set M is given by

$$M = \{\lambda | \lambda \geq 0, Z_D(\lambda) > -\infty\}$$

  The problem (LD) has several important structural properties that make it feasible to solve. As we have assumed that the problem (P) is feasible and that the set $X = \{x \mid Bx \geq d, x \geq 0 \text{ and integeral}\}$ of feasible solutions to the problem (LR_λ) is finite, this allows us to derive the problem (LD) into an LP model with many linear constraints. Since the strong duality theorem holds for a convex optimisation problem subject to linear constraints (Bertsekas, 1999) (see pp. 504-505), there is no duality gap and there exists at least one optimal λ^* satisfying:

$$\max_{\lambda \geq 0} Z_D(\lambda^*) = Z^* \tag{1.6}$$

This implies that solving the problem (LD) optimally gives us the optimal solution to the original problem (P). Two general techniques for deciding λ's are subgradient optimisation and multiplier adjustment. As the former is used in **Paper I**, we describe it later in this subsection. For the latter, we refer to (Fisher, Jaikumar, & Van Wassenhove, 1986).

In connection with the second issue above, Fisher (1981) analyses a closely related question:

- Can we find a value for λ such that $Z_D(\lambda)$ is equal to or nearly equal to Z?
  Let $(LR2_\lambda)$ denote the Lagrangian relaxation of the problem (P) being dualised with respect to the set of constraints $Bx \geq d$. Comparing (LR_λ) and $(LR2_\lambda)$ would give empirical results to find which is harder to solve but might provide better bounds. Also it is related to the question of how either of these relaxations compares with the LP relaxation, which leads us to ask:
- How can we choose between competing relaxations, i.e., different Lagrangian relaxations and the LP relaxation?
  Using the generalised assignment problem (GAP), Geoffrion (1974) provides an analytical result to this question. Let $Z_D = \max_{\lambda} Z_D(\lambda)$, and let Z_{LP} denote the optimal value of the LP relaxation. The result states that in the GAP, $Z_D \geq Z_{LP}$. $Z_D = Z_{LP}$ holds whenever $Z_D(\lambda)$ is not increased by removing the integrality restriction on x from the constraints of the Lagrangian problem. He calls this the *integrality* property. Many successful applications of the Lagrangian relaxations have had the integrality property (Etcheberry, 1977; Fisher & Hochbaum, 1980; Held & Karp, 1970, 1971). It is noted that for these applications, Lagrangian relaxation was successful mainly because the method used to optimise the problem (LD) was more powerful than methods available for solving the (generally large) LP relaxation of the problem (P) (Fisher, 1981).

We are now ready to describe the subgradient optimisation technique. It can be shown that $Z_D(\lambda)$ is a piecewise linear function. Therefore, solving $Z_D(\lambda)$ is a non-differentiable convex optimisation problem since the function is non-differentiable at a finite set of λ values. A successful technique for this problem is the subgradient method (N. Shor, 1962, 1964; N. Z.

Shor, 1985), which is an iterative algorithm to solve a non-differentiable convex function. Unaware of the work of Shor, Held and Karp (1971) developed a method for the TSP that uses subgradient optimisation to compute a bound in a Lagrangian relaxation method (Goffin, 2012). Let ξ denote the subgradient of the dual problem (LD). Then, the subgradient update of λ is

$$\lambda^{(k+1)} = \left[\lambda^{(k)} - t_k \xi^{(k)}\right]^+ \tag{1.7}$$

where $[\cdot]^+$ denotes projection on the closed convex set M, $\lambda^{(k)}$ is the k^{th} iterate, and $t_k (> 0)$ is the k^{th} step size.

The subgradient method for non-differentiable functions is similar to its gradient counterpart for differentiable functions, but with several notable exceptions (Boyd, Xiao, & Mutapcic, 2003). For example, the subgradient method uses step lengths that are fixed a priori, instead of an exact or approximate line search as in the gradient method. In contrast with its gradient counterpart, the new iterate in the subgradient method may not improve the dual cost for all values of the step size; that is, for some k, we may have

$$\mathrm{Z}_D\left(\left[\lambda^{(k)} - t_k \xi^{(k)}\right]^+\right) < \mathrm{Z}_D\left(\lambda^{(k)}\right), \qquad \forall t_k > 0. \tag{1.8}$$

Since the last iterating solution in the subgradient method may not be the best one found thus far, it is needed to keep track of the best incumbent while iterating. A succinct version of the subgradient procedure is described below:

(1) Choose initial $\lambda^{(0)} \in \Re^n$;

(2) In k^{th} iteration, compute the subgradient $\xi^{(k)}$ at $\lambda^{(k)}$. If $\xi^{(k)} = 0$, an optimal point has been found;

(3) The next iterate $\lambda^{(k+1)}$ of the sequence will be obtained by moving from $\lambda^{(k)}$ in the direction of $\xi^{(k)}$ by a certain step size. Go back to (2) with $k + 1$ replacing k.

The choice of step size is a very crucial issue in the performance of subgradient optimisation. As such, many different types of step size rules have been proposed and evaluated: see (Bazaraa & Sherali, 1981; Bertsekas, 1999; Goffin, 1977; Held, Wolfe, & Crowder, 1974). From a practical perspective, the choice of proper step size such that a near-optimal incumbent solution is obtained in a moderate number of iterations can be a formidable and sometimes elusive task (Sherali & Myers, 1988).

Exact models applied

The mathematical formulation derived for the optimisation in the terminal system is mainly a form of IP model or MIP model, the complexity of which typically belongs to the class of $\mathcal{NP}$-hard. Thus, although small scale instances can be solved through various solvers such as Cplex or Gurobi, instances of a more practical size cannot be solved within a reasonable computation time. It is more so when we add more realistic constraints to the model. It is evident not only from the experimental results reported in the literature but also from our computational results reported in **Paper I** (for the MIP model) and **Paper III** (for the IP model).

In the beginning of this subsection, we have briefly reviewed how the branch-and-bound method can be applied as a general algorithm framework for the combinatorial optimisation problem such as the IP. Subsequently, we have also reviewed Lagrangian relaxation, which is used as a part of solution technique in **Paper I**. As much as there are theories and improvements over at least five decades of the computing history, we could grasp only the tip of the iceberg for the subject of optimisation. Luckily though, from a practical perspective one does not have to know every detail of optimisation and computational theory when building an optimisation model and testing it. Good computer packages, i.e., solvers, exist for finding optimal solutions to IPs and MIPs with the most up-to-date computational advances in IP optimal solution methods. What is more important to a researcher who models optimisation problems in any particular area is to put effort into a good model formulation in terms of the variables and constraints. Deciding which model to formulate for a corresponding situation is often a combination of experience and trial and error.

In the next subsection, we shift our attention to the heuristic approach, which is another important branch for the optimisation technique.

5.2.3 Heuristics development

To cope with the possibility of computational intractability, the *second milestone* of this research is to develop the heuristic solutions for the corresponding models by exploiting the problem structure. This way we can elaborate the model, prove the validity of the model mathematically, and expand the boundary of feasible solutions in terms of problem scale. As is often the case with a complex optimisation model, it is essential to use heuristic solutions to find upper and lower bound from the relaxed formulation and push the gap between both bounds to be as narrow as possible. The relaxed formulation is employed in **Paper I**.

Overview of applied heuristics

Aside from the relaxed formulation, heuristic approaches for scheduling problems are generally divided into two types: *constructive type* and *improvement type*. The constructive (or *greedy*) type starts without a schedule, and gradually constructs a schedule by adding one job to the partial schedule at a time. In general, this greedy type heuristic is efficient because it makes a sequence of (local) decisions and there is no backtracking of the already taken decisions. However, the solution is not always optimal.

The improvement type starts with a complete schedule, which may be selected arbitrarily, and then tries to obtain a better schedule by manipulating the current schedule. An important class of improvement type algorithms is *local search procedure*. The local search procedure usually consists of the following four steps:

1) Initialisation: Choose an initial schedule S to be the current solution and compute the value of the objective function $F(S)$.
2) Neighbour Generation: Select a neighbour S' of the current solution S and compute $F(S')$.

3) Acceptance Test: Test whether to accept the move from S to S'. If the move is accepted, then S' replaces S as the current solution; otherwise S is retained as the current solution.
4) Termination Test: Test whether the algorithm should terminate. If it terminates, output the best solution generated; otherwise, return to the neighbour generation step.

Although the local search does not guarantee an optimal solution, it usually attempts to find a better schedule than the current one in the *neighbourhood* of the current one. Two schedules are *neighbours* if one can be obtained through a well-defined modification of the other. At each iteration, a local search procedure performs a search within the neighbourhood and evaluates the various neighbouring solutions. The procedure either accepts or rejects a candidate schedule as the next move, based on a given acceptance-rejection criterion (Michael, 1995). **Paper III** employs this two-staged heuristic approach, i.e., construction phase and improvement phase. A simple greedy heuristic based on the longest processing time (LPT) rule is used for the construction phase, and tabu search is used for the improvement phase.

Overview of neighbourhood search

Several options exist in the neighbourhood search, including efficient methods of finding initial seeds (i.e., initial schedule from the construction heuristic), selecting a generating mechanism and proceeding to a new seed (neighbour schedule). In sequencing and scheduling problem, a simple way is to select schedules in the neighbourhood at random, evaluate these schedules, and decide which one to accept. One may consider swapping those jobs that affect (improve) the objective the most. This approach is often called a *descent* technique because each new schedule represents a lower value of the objective function (in minimisation problems). However, this approach may easily end up with a local optimum, where the local search is stuck, i.e., no improving neighbours are available. To overcome this potential risk, the acceptance-rejection criterion can allow a next move towards less promising neighbours in an anticipation of finding better solutions than local optima. Simulated annealing and tabu search are examples of such heuristic approaches. In simulated

annealing, the acceptance-rejection criterion is based on a probabilistic process, whereas in tabu search, it is based largely or wholly on a deterministic process.

The origin of simulated annealing dates back to as early as 1953 in a paper by (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953), which described a Monte Carlo method to generate sample states of a thermodynamic system. The modern implementation of simulated annealing was done in 1980s by (Kirkpatrick, Gelatt, & Vecchi, 1983) and (Černý, 1985) independently. *Annealing* is a term borrowed from the physical system, specifically with the way that metals, and some liquids, cool and crystallise. The term refers to a process of cooling material slowly, until the material reaches a stable (frozen or crystallised) state. In the beginning of this process, particles in the material at high temperatures will sometimes change to higher-energy states, but at low temperatures such behaviour is much less likely. At very low temperatures, particles virtually always move to lower-energy states whenever the opportunity arises. Eventually, the movement toward low-energy states leads to freezing (Baker & Trietsch, 2009). With the objective function playing the role of energy, the simulated annealing process in combinatorial optimisation problems allows a worse solution with high probability (= *melting the material at a high temperature*) at the start of the search, but later in the search with less probability (=*lowering the temperature by slow stages*) until the system freezes and no further changes occur.

In contrast to the probabilistic nature of simulated annealing, tabu search bases the acceptance-rejection criterion largely or wholly on a deterministic process. In order to diversify the neighbourhood search, tabu search allows a move to a worse neighbourhood in the following systematic way. Instead of stopping when a tabu search procedure arrives at a local optimum, it moves to a new seed even if its solution value is worse than that of the local optimum. In order to avoid cycling back to previously visited seeds (i.e., of a better solution value than the new seed), the move back to the previous seed is designated as *tabu* or *forbidden* move. In the same spirit, we may also forbid a move back to the second or third previous seeds, which all together form a tabu list. At each stage, the procedure selects the best solution from those in the neighbourhood that are not on the tabu list (Baker & Trietsch, 2009).

Both simulated annealing and tabu search are called *meta-heuristics* in the sense that they are problem-independent techniques (as opposed to problem-dependent heuristics), and as such, they do not take advantage of any specificity of the problem. The term *meta-heuristic* was coined in the same paper that introduced the term *tabu* search (Glover, 1986), and has received a wide acceptance in the literature ever since. Two important characteristics of meta-heuristics are worth to be noted: intensification and diversification (Blum & Roli, 2003). Intensification intends to search locally and more intensively while diversification intends to explore the search space more globally (and hopefully more efficiently). A good balance between these two components is important to the overall efficiency and performance of a heuristic.

Simulated annealing and tabu search can also be grouped into a population-based search, the general framework of which is developed in the form of genetic algorithm, which in turn belongs to the larger class of evolutionary algorithms. Inspired by evolutionary biology, genetic algorithms generate solutions to optimisation problems using search methods based on the genetic notions of inheritance, mutation, selection and crossover. This idea was born in the 1950s in a series of papers by (Fraser, 1957) and (Bremermann, 1958). It gained popularity through the works of John Holland in the 1970s, and particularly through his book (Holland, 1975). Up to now, several variants of genetic algorithms have been applied to a number of optimisation problems (Goldberg, 1989; Nikjoofar & Zarghami, 2013; Rani, Jain, Srivastava, & Perumal, 2012). In general, genetic algorithms select subsets of solutions from a population, called *parents*, and combine them to produce new solutions, called *children*. Children that pass a survivability test are then available to be chosen as parents of the next generation. The final result of this process is the best (*fittest*) solution (*survivor*). In what follows, we continue the description of tabu search, which is applied for the improvement heuristic in **Paper III**.

Tabu search

Developed and refined in a series of works by Fred Glover (Glover, 1986, 1989, 1990), tabu search is a global optimisation method that allows hill climbing to overcome local optima. Instead of terminating when the search arrives at a local optimum, it moves beyond the local

optimum by choosing the best possible neighbour. Cycling back to previously visited solutions is prevented by the use of memory with the so called tabu list that keeps history of recent visits during the search. The tabu list is kept in memory, with a prescribed limit in the length of the list. Once the list is full, the algorithm replaces one tabu move from the list with the new tabu, in which the oldest tabu move is usually replaced first. The size of the tabu list should be large enough to avoid cycling but small enough not to forbid too many moves, and not to consume too much memory and effort computationally.

Although central to the art of tabu search, the tabu list is so powerful such that even when there is no risk of cycling back, the tabu list may forbid attractive moves. Thus, it is necessary to devise an algorithmic routine that allows tabus to be revoked. An *aspiration criterion* is one such way. The simplest form of this is to allow a tabu move when it results in a solution better than the current best-known solution, which can be found in almost all tabu search implementations including ours. More sophisticated aspiration criteria have been proposed in the literature, e.g., (De Werra & Hertz, 1989; Hertz & de Werra, 1990), but they are seldom used. In fact, many elements of this tabu search proposal, and some elements of later elaborations, were introduced in (Glover, 1977), including short-term memory to prevent the reversal of recent moves, and longer-term frequency memory to reinforce attractive components (Gendreau & Potvin, 2005). See also (Salhi, 2002) for the study of tabu list size and aspiration criterion within tabu search.

In tabu search, randomisation is de-emphasized on the assumption that intelligent search should be based on more systematic guidance. Accordingly, many implementations in tabu search are largely or wholly deterministic, and randomisation is applied only in a highly constrained fashion. However, as a potential hedge against the risks of strategic myopia, an exception occurs for the variant called *probabilistic tabu search* (Glover & Laguna, 1997). It selects moves according to probabilities based on the status and evaluations assigned to these moves by the basic tabu search principles. For clarification, since more than one interpretation is possible, what is generally regarded as probabilistic tabu search is usually applied to the acceptance criterion. One may view this randomisation in tabu search as a means for obtaining diversity without much reliance on memory. In a computational experiment conducted in (Btażewicz, Salvador, & Walkowiak, 2002), however, there is no

clear evidence that any variant of tabu search, either deterministic or probabilistic, performs much better than other. Hence, our heuristic approach just focuses on deterministic tabu search, which provides a more systemic guidance on the behaviour of the search. For the general description on probabilistic tabu search, see (Glover & Laguna, 1997; Løkketangen & Glover, 1996). Also, some discussions about similarities and differences between probabilistic tabu search and simulated annealing are notable in that both use probabilistic measures for move acceptance. For a description of such differences, see (Dowsland, 1993).

For a glimpse of the search process applied in **Paper III**, we are now prepared to describe a template tabu search as follows. From the construction heuristic, which runs with a controlled randomisation in the greedy fashion, a prescribed number of multiple initial seeds are obtained ordered by their attractiveness in terms of the objective value. A randomisation in the construction heuristic is considered "controlled" in the sense that only the first task-to-machine assignment is randomised in the runs of multiple constructions. This implies that while maintaining the best known-solution, tabu search (improvement heuristic) repeats the search process with each initial seed as many times as there are multiple initial seeds. Clearly, there can be a trade-off between the number of initial seeds and the potential improvement in the final solution. First, notations follow below. We then present the template version of tabu search in Algorithm 1.1.

Notations:

- s the current solution;
- s^* the best incumbent solution;
- v a move available from s;
- f the objective function;
- $\sigma(s, v)$ the function that returns the solution obtained by move v to s;
- $V(s)$ a set of all moves available from s
- $H(s)$ the neighbourhood of s, hence $H(s) = \{\sigma(s, v) | v \in V(s)\}$;
- $\widetilde{H}(s)$ the "admissible" subset of $H(s)$ such that a move v to s' is not in the tabu list and the aspiration criteria hold
- T tabu list.

Algorithm 1.1 Template tabu search

Step 1	**Initialisation**
	Construct multiple initial solutions, and choose the best solution s_0;
	Set $s \coloneqq s_0, s^* \coloneqq s_0, T \coloneqq \emptyset, iter \coloneqq 0$.
Step 2	**Find the admissible subset of neighbours $\widetilde{H}(s)$.**
	Set $iter \coloneqq iter + 1$;
	Generate $V(s)$ and $H(s)$ from s, and find $\widetilde{H}(s)$.
Step 3	**Update the best incumbent s^***
	Set $s \coloneqq \underset{s' \in \widetilde{H}(s)}{\operatorname{argmin}} f(s')$. Then, s is the best solution among $\widetilde{H}(s)$;
	If the evaluation of s is better than s^*, e.g., $f(s) < f(s^*)$, set $s^* \coloneqq$ s.
Step 4	**Update tabu list T**
	Record tabu for the current move in T (after removing the oldest entry if necessary).
Step 5	**Check stopping conditions**
	If a stopping condition is met, then stop;
	Otherwise, go to Step 2.

In the following, we close this methodology section after giving a brief explanation of benchmark instances available to the problems posed in the thesis.

5.2.4 Benchmark suite

The evaluation of solution procedures for particular type of problems requires benchmark instances. Depending on the problem domain, there are some well-known benchmark instances, e.g., for TSP (Reinelt, 1991), for machine scheduling problems (Demirkol, Mehta, & Uzsoy, 1998; E. Taillard, 1993), and for resource-constrained project scheduling problems (Kolisch & Sprecher, 1997; Kolisch, Sprecher, & Drexl, 1995).

As aforementioned, a set of standardised problem instances in the QCSP are proposed by (Kim & Park, 2004) and their instance sets have subsequently been verified and compared as the benchmark instances by several successive researchers. More recently, Meisel and Bierwirth (2011b) propose a unifying benchmark generator for the QCSP, called *QCSPGen* (Meisel & Bierwirth, 2011a), which is gaining attention among researchers in this area. The suitability of this benchmark generator to the double cycling model is briefly discussed in

Paper III. In general, complying with benchmark platforms is a good approach to test the effectiveness of the developed model.

6 Overview of the papers

As an illustrative overview for the papers included in the thesis, Figure 6.1 locates each paper in a two-dimensional chart with x-axis being the physical area in a marine terminal and y-axis being the classification of decision levels to obtain an efficient terminal. I. F. A. Vis and De Koster (2003) identify the latter as the following three levels: the strategic level, the tactical level and the operational level. Examples of the strategic level decisions include the decision for the type of material handling or the minimum amount of storage space needed. Examples of the tactical level decisions include the number of yard equipment to ensure an efficient storage and retrieval process. At the operational level, the problem of minimising, predicting or scheduling the amount of work may be discussed.

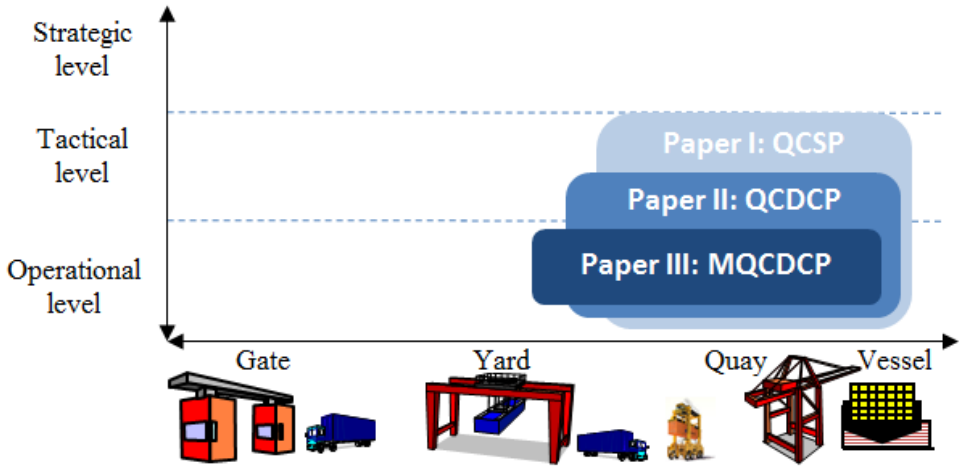


Figure 6.1 Overview of paper positions

The above diagram helps a reader grasp the scope of each paper in the context of marine terminal environment. **Paper I** is an example where the QCSP under consideration is involved with multiple decision levels for an efficient terminal. The trend of more recent literature confirms it with the integration among the BAP, the QCAP and the QCSP. **Paper II** illustrates the QCDCP can be a subclass of the QCSP but has more focus in the operational level with more interaction with yard. **Paper III** considers the MQCDCP as a part of the

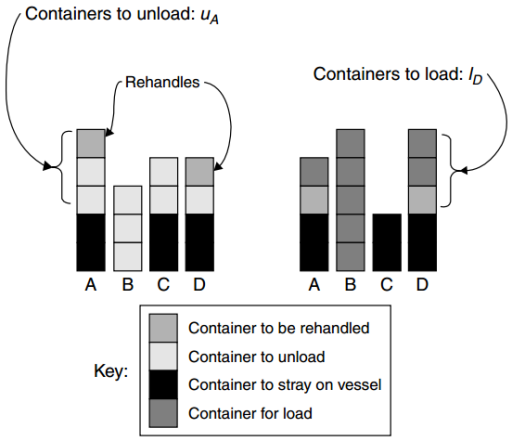
QCDCP but its involvement with yard operation is more emphasised, making the problem look dedicated to the operational level. The following chapters will discuss these problems one after another, filling the gaps in the existing body of literature.

Appendix: Double Cycling Problem as Two-Machine Flow Shop Problem

The formulation below is sourced from (Goodchild & Daganzo, 2005b, 2006).

If we consider that the time it takes to unload and load a ship is a measure of crane efficiency, then the goal of double cycling is to reduce the maximum completion time of all jobs. A proxy for this is the number of cycles required to handle all containers to unload from and load onto the ship. The number of cycles necessary to complete the loading and unloading will be represented by the variable (ω). We will consider the double cycling within one bay of the ship. We use the following notation:

- S – the set of stack labels in a ship bay;
- $|\mathrm{S}| = \mathrm{N}$ – the numer of stacks in the set;
- Π – a permutation of S indicating an ordering of the stacks;
  In the below figure, for example, the set of stack labels is $\mathrm{S} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$. A permutation of these is $\{\mathrm{B}, \mathrm{A}, \mathrm{C}, \mathrm{D}\}$ .given by the function Π_e where $\Pi_e(1) = B, \Pi_e(2) = A, \Pi_e(3) = C, \Pi_e(4) = D$. In the following, we consider two types of permutations, Π for a loading permutation and Π' for an unloading permutation.



- u_c – number of containers to unload in stack $\mathrm{c} \in \mathrm{S}$

- $l_c -$ number of containers to load in stack c ∈ S
- $FU_c -$ completion time of unloading c ∈ S
- $FL_c -$ completion time of loading c ∈ S
- $\omega \ -$ maximum completion time
- $X_{kj} -$ binary variable to for ordering of unloading jobs (1 if $j \in$ S is unloaded after $k \in$ S and 0 otherwise)
- $Y_{kj} -$ binary variable to for ordering of loading jobs (1 if $j \in$ S is loaded after $k \in$ S and 0 otherwise)
- M – a large number

Therefore, the objective of the scheduling problem (SP) is to minimise the maximum completion time (ω) of all jobs subject to constraints. The result is to uniquely identify the permutations Π and Π′, and a feasible set of job start and end times. It is assumed that the process starts at time zero. The formulation is:

(a) $minimise \ \omega$

(b) subject to $\omega \ \geq \ FL_c \ \ \forall c \in S.$

(c) $FL_c - FU_c \ \geq \ l_c \ \ \forall c \in S$

(d) $FU_k - FU_j + MX_{kj} \ \geq \ u_k \ \ \forall j, k \in S$

(e) $FU_j - FU_k + M\left(1 - X_{kj}\right) \geq \ u_j \ \ \forall j, k \in S$

(f) $FL_k - FL_j + MY_{kj} \ \geq \ l_k \ \ \forall j, k \in S$

(g) $FL_j - FL_k + M\left(1 - Y_{kj}\right) \geq \ l_j \ \ \forall j, k \in S$

(h) $FU_c \ \geq \ u_c \ \ \forall c \in S$

(i) $X_{kj}, Y_{kj} = 1, 0 \ \ \forall j, k \in S$

These constraints completely define the double cycling problem.

Constraints (b) ensure that the makespan is greater than or equal to the completion of loading of all stacks. Constraints (c) ensure that stacks are only loaded after all necessary stacks have been un-loaded. Constraints (d), (e) and (i) ensure that every stack is unloaded after the previous one in Π′ has been unloaded. This is achieved by specifying for every pair of stacks (j, k) that either stack k is unloaded before stack j (if $\mathrm{X}_{kj} = 1$) or the reverse (if $\mathrm{X}_{kj} = 0$),

and that the time difference between the two events is large enough to unload the second of the two stacks. Constraints (f), (g) and (i) are equivalent to (d), (e) and (i) but for loading jobs. Constraints (h) ensure that each unloading completion time allows for enough time to at least unload that stack.

Paper I

Quay Crane Scheduling in a Preemptive Bay Operation

Dusan Ku and Tiru Arthanari

— Unpublished

Abstract

A rich body of literature on the quay crane scheduling problem (QCSP) in container terminal operations has emerged over the past two decades. The paper attempts to bring out most of the literature on this subject matter for a review by focusing on various assumptions and constraints discussed in the existing models. The research papers reviewed are summarised and partially re-categorised according to the classification framework introduced in (Bierwirth & Meisel, 2010), which is dedicated to scheduling problems of container terminals. The paper then identifies that the exact model for preemptive schedules in bay-wise vessel operation has not yet been formulated in the literature. Hence, a mixed integer programming (MIP) model is given for a scenario where the granularity of tasks is a ship bay and each task is preemptible. A computational comparison is given for the exact solution and the relaxed formulation under this scenario.

Keywords: *Shipping/transportation, Container terminal, Quay crane scheduling, Preemptive schedules, Mixed integer programming*

1 Introduction

Since its introduction around 1950-60s, the container has rapidly taken over the market for intercontinental transport (Meersmans & Dekker, 2001). Containers are typically owned by shipping lines and each shipping line may have its own preferred specification for manufacturing container boxes. Some shippers (usually large shippers) may have their own container boxes, but containers of shipping lines are a major portion for the freight transport via seaport. Nowadays, dimensions of containers for general freight purpose have been standardised by the International Standards Organization (ISO), which enables uniform container handlings across machineries and sites. This has allowed large savings of time and cost, thus contributing to the wide spread of containers.

Marine container terminals are the gateways for distributing containers globally and play a vital role in completing the global supply chains. Over the last several decades, substantial investments have been made for port developments both in facilities and in IT infrastructure. Considering the rapidly growing level of service requirement that shipping lines demand to the terminal operators, the efficiency of terminal operation is becoming a matter of business survival. Reflecting these inevitable trends, much research has been conducted and the terminal facility equipped with the state-of-the-art technology and decision support systems using OR/MS techniques are being deployed to the container handling system. Comprehensive surveys on the decision problems at marine container terminals can be found in (Stahlbock & Voß, 2008; Dirk Steenken et al., 2004; I. F. A. Vis & De Koster, 2003).

In a typical marine container terminal, ships are berthed alongside the quay, and each ship is served by a number of quay cranes which unload and load containers. Horizontal transport of containers between the quay and the storage yard can usually be performed by a variety of transport vehicles, which can be classified into two different types: The first type, to which yard trailers and automated guided vehicles (AGVs) belong, is not able to lift or stack containers by themselves; The second type, to which straddle carriers (SCs) belong, is able to lift or stack containers by themselves. Thus, in a terminal where the first type of transport vehicles are used, a number of yard cranes are expected to operate in the storage yard to handle (lift or stack) containers to/from the transport vehicles.

In Figure 1.1, Henesey (2006) gives an illustration of four main subsystems in a container terminal system, based on the physical area where the container flow takes place. Each subsystem has its unique characteristics depending on the terminal configuration, so managing such systems optimally is quite complex. For this reason, analytical optimisation solutions usually tackle each subsystem separately and try to restrict interactions between subsystems even though it is true that pursuing an optimum in one area does not guarantee the global optimum in the entire terminal system. Since a substantial amount of machinery is usually involved at each interface for the flow of a fairly large number of containers, a sophisticated planning and operation is desired in the terminal system. The concern of this paper deals with, among other things, the first interface: *ship-to-shore*. Bierwirth and Meisel (2010) provide a comprehensive survey on this interface by decomposing it into three sub-problems: the *berth allocation problem* (BAP), the *quay crane assignment problem* (QCAP) and the *quay crane scheduling problem* (QCSP). The follow-up surveys in (Bierwirth & Meisel, 2014; Carlo, Vis, & Roodbergen, 2013) provide the most recent literature review in this field, including the integrated planning problems among the BAP, the QCAP and the QCSP.

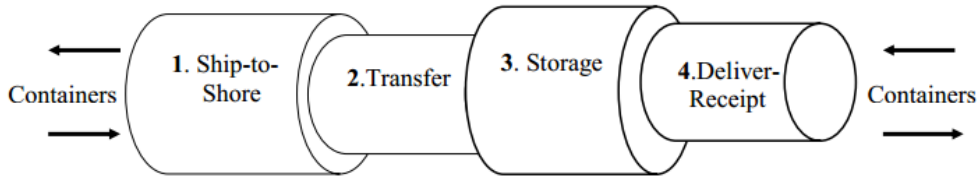


Figure 1.1 Four main subsystems in a terminal system. *Source*: (Henesey, 2006)

The goal of the BAP is, given the berth layout of a container terminal, to accommodate as many vessel schedules as possible within the planning horizon. The goal of the QCAP is, given a feasible berth plan, to assign quay cranes (QCs) to vessels such that the handling of all required containers can be fulfilled. The goal of the QCSP is to determine the sequence of container handling tasks of the QCs serving a vessel, and its objective is usually the minimisation of the makespan, the minimisation of the total completion time of QCs, or the combination of both.

The BAP and the QCAP are basically interrelated because solving the QCAP can have a strong impact on the berthing times. Thus, the integrated problem of the two problems can be termed the *berth allocation and crane assignment problem* (BACAP), of which the goal is to decide on berthing positions, berthing times and the assignment of cranes to vessels (Meisel, 2009). Similar to the concept of the BACAP, the integration of the QC scheduling with the assignment of QCs to vessels simultaneously can be termed the *quay crane scheduling and assignment problem* (QCSAP, or alternatively, QCASP hereinafter) (Tavakkoli-Moghaddam, Makui, Salahi, Bazzazi, & Taheri, 2009). A little research addresses the integration of the BAP and the QCSP simultaneously (Han, Lu, & Xi, 2010; L. Song, Cherrett, & Guan, 2012), for which the integrated problem may be termed the *berth allocation and crane scheduling problem* (BACSP). An integrated framework for the three phases of seaside planning can be found in (Meisel & Bierwirth, 2013).

In a sequential approach, a berth schedule at the high level can be first constructed by the shipping lines' requests, taking into account the vessel calling schedule, its preferred berth location, the availability of terminal resources and other environmental factors such as tide and weather. Then, the lower-level problems, i.e., the QCAP and the QCSP, have to utilise available resources to optimise the objective while satisfying the constraints of the already determined berth schedule. If a model is constructed this way, then it becomes a hierarchical decomposition model among the sub-problems. However, for example, the duration of vessels' berthing at the terminal depends on the number of crane hours allocated to the vessel. If the number of crane hours allocated to a vessel increases, the duration of the vessel's berthing is likely to be reduced. Hence, the hierarchical decomposition model between the berth scheduling and the QC assignment is not necessarily sequential. That is, the model will necessitate a feedback process among the sub-problems, taking simultaneous interactions into account. This behaviour is illustrated in Figure 1.2 to represent the feedback relationship among the seaside sub-problems.

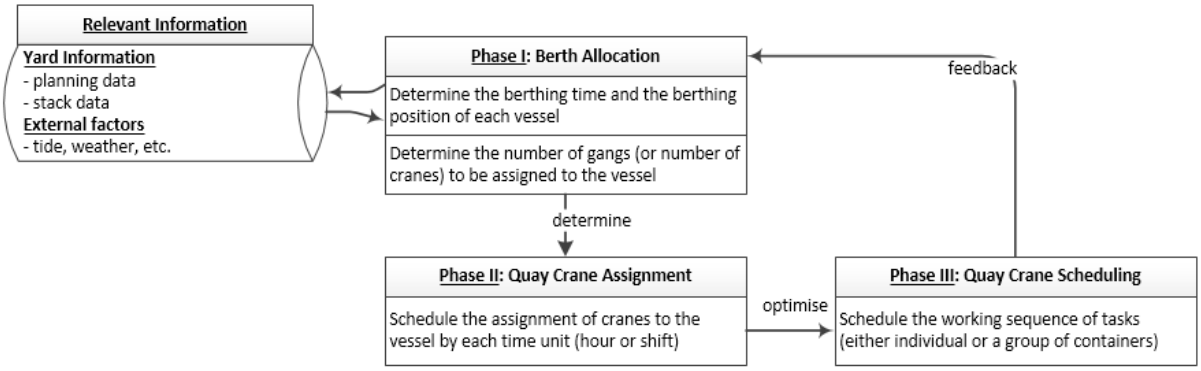


Figure 1.2 Integrated relationship of seaside sub-problems

This paper is organised as follows: Section 2 reviews relevant literature on the QCSP. Section 3 and 4 describe the mathematical model addressed in the paper and the relaxed formulation. Section 5 gives the experimental results using this model. Finally, we conclude the paper in Section 6.

2 Related literature

The scheduling problem of cranes sharing a common track in some industrial processes is introduced as early as in the works of (Lieberman & Turksen, 1981, 1982). They provide the basic model of crane systems with the single track constraint, the terminology, and some complexity results. Except for the simplest crane systems, the computational complexity of crane scheduling is $\mathcal{NP}$-hard. A heuristic algorithm is employed in the two-crane systems (Lieberman & Turksen, 1982). The heuristic yields schedules whose makespan is at worst 4/3 the lower bound. However, the method is insufficiently general to be applied to the QCSP, and no computational results are reported.

In the QCSP, the objective in general is to find a working schedule of QCs for the loading and unloading tasks of containers in a vessel such that overall vessel handling time is minimised. Constraints such as non-crossing between cranes, ensuring safety rules, and keeping precedence among tasks may be involved. Studies on the QCSP date back to as early as the late 80s in (Daganzo, 1989; Peterkofsky & Daganzo, 1990). Then there is over a decade's pause in the scientific research on the QCSP until spatial constraints begin to be

introduced by (Lim, Rodrigues, Xiao, & Zhu, 2002) and precedence relations among tasks are formulated in the model by (Kim & Park, 2004).

In survey research by (Bierwirth & Meisel, 2010, 2014), tasks in the QCSP are defined on the basis of bay areas, single bays, container stacks, container groups or individual containers. The present paper reassigns a few studies classified as *individual containers* in (Bierwirth & Meisel, 2014) to *the preemptive scheduling of bays (or groups in general)* if the model considers the discretised planning horizon where a container is processed during each time unit. Lim et al. (2002) have first proven the QCSP with complete bays to be $\mathcal{NP}$-complete, and Sammarra et al. (2007) have first proven the QCSP with container groups to be $\mathcal{NP}$-hard.

2.1 Preemptive scheduling of bays

In the research stream of bays as tasks, both preemptive and non-preemptive scheduling are studied. In the preemptive scheduling, which allows the sharing of bays among cranes, Daganzo (1989) partitions ships into bays as single task jobs. Thus, no precedence constraints are considered among containers in the same bay. They consider both preemptive and non-preemptive scheduling with parameters. The preemptive version of their model assumes that the workloads in a ship bay may be served by at most two cranes. Cranes can move freely, hence no interference among QCs is modelled. In the static problem, all ships are ready for processing at time zero for a finite horizon. In the dynamic problem, ships arrive during the planning horizon and cannot be handled before their arrival times. The model divides time into discrete slots, and the problem is then formulated as a mixed integer programming (MIP) model with the aim of minimising the ship delay cost.

Under similar assumptions to (Daganzo, 1989), Peterkofsky and Daganzo (1990) develop a branch-and-bound (B&B) algorithm with the objective of minimising the weighted sum of a ship's turnaround times. The crane interference is not considered in their work, either. The performance of the algorithm deteriorates quickly when the problem size grows beyond six ships with each ship having 2–5 bays randomly.

Preemptive scheduling is also studied in a version of (J. Liu et al., 2006), whose ultimate objective is to minimise the maximum relative tardiness of all ships in a planning horizon.

Unlike the above mentioned studies, they consider the non-crossing constraints, the safety distances and the effect of travelling times of QCs. To solve problems of practical sizes, they decompose the problem into two-levels: a vessel-level and a berth-level model. The higher berth-level problem determines vessel berthing times and QC-to-vessel assignments, based on the solutions to QCSPs at the vessel-level. They provide two types of MIP models for unidirectional schedules: one with preemptive schedules and the other with non-preemptive schedules. Note that seeking the optimal schedule among the unidirectional ones is the current practice in marine container terminals. The models, however, fail to determine a suitable temporal distance, explained in (Bierwirth & Meisel, 2009), between any two tasks permitting a safe movement of the in-between cranes. Heuristic algorithms are developed by forcing all QCs working on a vessel to stay there until the processing of all bays in the vessel is completed. Based on experimental results with up to 12 QCs and 14 vessels, the paper concludes that the proposed heuristics yield effective solutions in a reasonable amount of time.

Under the discretised planning horizon for the single and multi-ship cases, Choo, Klabjan, and Simchi-Levi (2010) provide the generalised set covering formulation and outline the underlying branch-and-price (B&P) algorithm. For the multi-ship sequencing, the yard congestion constraints are relaxed by the Lagrangian relaxation. Relaxing the yard congestion constraints allows the separation of the original problem into independent sub-problems having a structure similar to the single ship model. The primal heuristic is proposed to correct some infeasibilities in the Lagrangian solutions. Computational experiments show that large scale instances can be solved in a reasonable computational time.

Lu et al. (2012) point out the concept of contiguous bay range and develop a QCSP model allowing ship bays to be served by more than one QC (i.e., shared bay). The number of containers in each bay to be served by each QC is left as a decision variable. They provide a polynomial-time heuristic to generate conflict-free QC schedules.

The integrated model for the QCAP and the QCSP, namely the QCASP, is studied in (Diabat & Theodorou, 2014; Fu, Diabat, & Tsai, 2014; Theodorou & Diabat, 2014). We classify (Fu et al., 2014) into a preemptive scheduling model because it is based on the crane-to-bay

assignment during the discretised planning horizon. The model in (Fu et al., 2014) is not necessarily restricted to unidirectional schedules, and the QC traveling time is ignored from the model. The objective of minimising the makespan of container terminals is translated into the maximisation of the number of work completion flags during a given planning horizon. An IP model is formulated in discretised time segments, and a GA is proposed to determine near optimal solutions for the given problem.

2.2 Non-preemptive scheduling of bays

Other studies in the stream of bays as tasks are related to non-preemptive schedules. In the non-preemptive scheduling, a smaller granularity of the workload leads to more accuracy of the QC scheduling model, but it can only be achieved at the expense of increased complexity and smaller problem scale.

Lim et al. (2002) and Lim, Rodrigues, Xiao, et al. (2004) represent the QCSP by a bipartite graph matching problem where cranes and jobs (or ship bays) are two separate sets of vertices, and crane-to-job throughputs are the weights of connecting edges. The model considers three types of spatial constraints: namely, non-crossing constraint, neighbourhood constraint and job-separation constraint. In the existing literature, neighbourhood constraint refers to the safety margin, and job-separation constraint refers to the non-simultaneous jobs. There is a profit value when a job is assigned to a QC. The objective is to find a crane-to-job matching which maximises the total profit under those constraints. With the proof that the problem is $\mathcal{NP}$-complete, they provide a dynamic programming (DP) algorithm as an exact solution. A probabilistic tabu search (PTS) heuristic is designed to find good solutions in large scale instances. Lim, Rodrigues, Xiao, et al. (2004) additionally apply squeaky wheel optimisation (SWO) heuristics with and without local search heuristic. However, it is difficult to define a profit value associated with a crane-to-job assignment in practice.

Zhu and Lim (2004) and Zhu and Lim (2006) provide the integer programming (IP) model, which satisfies the non-crossing constraint. The formulation ensures it by specifying the constraint such that if job i is assigned to crane k and job j is assigned to crane l simultaneously, then $i < j$ if and only if $k < l$. The proof of $\mathcal{NP}$-completeness is given for

this problem class by reducing it to the set partition problem. A B&B algorithm is designed to obtain optimal solutions. Also, a simulated annealing (SA) meta-heuristic with effective neighbourhood search is designed to obtain near-optimal solutions in large scale instances.

In (Lim, Rodrigues, & Xu, 2004), a constraint programming (CP) model is provided to find an optimum crane-allocation map, which minimises the latest completion time. With an $\mathcal{NP}$-completeness proof to the problem, they provide a DP algorithm to solve the problem optimally in a pseudo-polynomial time when the number of cranes is fixed. They also provide three approximation algorithms that guarantee approximation factors of 2 when the number of cranes is arbitrarily large. Recall that ρ-approximation algorithm produces a schedule whose latest objective value is at most ρ times worse the optimal solution in polynomial time, where ρ is called the approximation ratio (Vazirani, 2001).

Lim et al. (2007) study an *m*-parallel machine scheduling problem which incorporates a non-crossing spatial constraint. They show that for the QCSP in the case of complete bays, i.e., one task per bay, there is always an optimal schedule among the unidirectional ones. They provide both CP and MIP formulations for the QCSP, and the CP is solved using B&B algorithm. The exact solution grows exponentially in time when the number of cranes increases. Thus, a SA heuristic is presented to solve the problem efficiently when the numbers of machines and jobs are of realistic sizes.

D.-H. Lee, Wang, and Miao (2007) focus on the QCSP for a single vessel. The model considers the non-crossing constraint but ignores the travel time of QCs. An MIP model is provided for an exact solution, and with the proof of $\mathcal{NP}$-completeness, a 2-approximation heuristic algorithm is proposed. Under the same problem setting as (D.-H. Lee, Wang, et al., 2007), D.-H. Lee et al. (2008b) employ a genetic algorithm (GA) to achieve near optimal solutions. Computational experiments show that the average gap above the lower bound, which is the objective value of the optimal solution to the relaxed problem without crane interference constraint, is 0.41% for forty instances with the maximum gap being 2.66%. M. Liu, Zheng, and Li (2014) provide 4/3- and 5/3-approximation algorithms for the case with two QCs and the case with three QCs, respectively, which are improvements over the 2-approximation algorithm proposed by (D.-H. Lee, Wang, et al., 2007).

D.-H. Lee et al. (2008a) consider a potential terminal practice in which a certain ship bay may have handling priority over another. The problem is termed the quay crane scheduling with handling priority problem (QCSHPP). Other constraints considered are the same as in (D.-H. Lee, Wang, et al., 2007). An MIP model is given to the QCSHPP. Since the problem is computationally intractable for large scale instances, a GA is proposed to obtain near optimal solutions. To the same problem setting, D.-H. Lee and Wang (2010) provide a proof of $\mathcal{NP}$-completeness as well as an approximation algorithm with the worst-case analysis. D.-H. Lee and Qiu Wang (2010) provide integrated formulations for the BAP and the QCSP with a proof of $\mathcal{NP}$-completeness. The proposed GA finds near optimal solutions. D.-H. Lee and Chen (2010) strengthen the previous MIP models after identifying potential crane interference. Two approximation schedules are studied to obtain near optimal solutions: the best partition (BP) method and the enhanced best partition (EBP) method.

In (Y. Wang & Kim, 2011), the assignment of yard blocks to vessels is considered in the QCSP. An attempt is made to construct QC schedules whereby the total workload of yard cranes (YCs) in each block does not exceed its capacity. The objective terms include the QC makespan, the total traveling distance of QCs, the expected delay time from interference, and the uniformity of workload among blocks. The proposed heuristic algorithm, based on a Greedy Randomized Adaptive Search Procedure (GRASP) heuristic, consists of the construction and the improvement phases. It shows a reasonable performance for the application to practices.

D.-H. Lee, Chen, and Cao (2011) investigate the QCSP at an indented berth where a ship's operation can be served by QCs that are located at two opposite berths. While cranes located at one side are subject to the non-crossing constraint, cranes located at the opposite side of berths can pass each other when the spreader arm is lifted. As such, the model differentiates between interfering and noninterfering subsets of cranes. The model also considers the safety distance requirements, but the travel time of QCs is ignored. The precedence relations among tasks are not considered as tasks are based on the complete bays. An MIP model is developed to capture the characteristics of the proposed problem. Due to the problem class being $\mathcal{NP}$-hard, a tabu search heuristic is proposed to find near optimal solutions.

S. H. Chung and Choy (2012) base their MIP formulation on the model by (Kim & Park, 2004) and propose a modified GA for solving the model. A set of well-known bench marking problem is solved, and the computational comparison demonstrates that the proposed GA can achieve results very close to the best known solutions by other approaches that were implemented for these instances, such as tabu search in (Sammarra et al., 2007) and branch-and-cut (B&C) in (Moccia et al., 2006). Especially for smaller problem sizes, the gap between the proposed GA and the best known solution is 0%. Although the performance of GA deteriorates in large scale instances, the computational time of GA is more attractive and practical.

Boysen, Emde, and Fliedner (2012) give a proof of $\mathcal{NP}$-hardness for the QCSP at an indented berth. An exact and heuristic procedures based on the DP are designed, and the beam search (BS) finds near optimal solutions in a short time.

In (S. Wang, Zheng, Zheng, Guo, & Liu, 2012), a job is defined as a loading or unloading operation for a ship bay. With the aim of minimising the weighted operation time of jobs and travel time of QCs, a particle swarm optimisation (PSO) algorithm, originally introduced in (Kenndy & Eberhart, 1995), is developed to solve the problem. The simulation experiments are applied for minimising the total turnaround time of discharging process and show that the improved discrete PSO can get the best scheduling solution. Moreover, it is observed that the PSO is more effective compared with the ant colony optimisation (ACO) algorithm in (S. Wang & Hu, 2009).

Guan, Yang, and Zhou (2013) develop a time-space network flow model to formulate the QCSP under non-crossing and non-interference constraints. The model is extended to the Lagrangian relaxation approach to solve medium scale instances. To solve large scale instances, a threshold policy heuristic and a DP heuristic are developed. The worst case analysis shows that the objective value of any feasible solution obtained by the heuristics is bounded by 2 times the optimal solution.

Diabat and Theodorou (2014) consider the QCASP, of which the purpose is to assign cranes to ships that are berthed within a given planning horizon and to decide which crane is allocated to which bays. The objective of the integrated problem is to minimise the time

required for the completion of the handling of the latest ship. An MIP formulation is given for the QCASP, and the employed GA is benchmarked against an exact solution which allows for clear evaluation of its performance.

Theodorou and Diabat (2014) study the integrated problem of the QCASP, in which an MIP model is developed with a Lagrangian relaxation algorithm. In order to ensure feasibility, a restoration technique is developed. The Lagrangian relaxation is solved using the constraint generation method, and computational results show the performance of the Lagrangian heuristic, in terms of computational time, Lagrangian bounds and optimality gaps.

2.3 Scheduling of container groups

The QCSP on the basis of container groups considers only the non-preemptive scheduling. A rich body of research in this stream starts with the model by (Kim & Park, 2004), which consider multiple tasks in each bay, precedence relations among tasks and crane interference. Sammarra et al. (2007) note that with the relaxation of spatial constraints, the QCSP of this kind reduces to the problem of scheduling a set of tasks on a set of parallel identical machines, subject to precedence constraints, the objective function being the makespan ($Pm|prec|C_{max}$). This problem belongs to the class of $\mathcal{NP}$-hard problems (Michael, 1995).

In (Kim & Park, 2004), tasks are characterised by similar container attributes, e.g., the same size, the same destination, etc. Their B&B solution approach works well on instances with two cranes and up to 10-15 tasks, whereas instances over three cranes and 20-25 tasks fail to be solved within a reasonable timespan. To overcome the computational difficulty for larger instances, a GRASP heuristic is suggested. The formulation of the QCSP with container groups has been rendered more robust by the works including (Moccia et al., 2006; Sammarra et al., 2007; Tavakkoli-Moghaddam et al., 2009).

Moccia et al. (2006) note the possibility of crane interference in Kim and Park's model and revise the model of Kim and Park by incorporating the travel times for QCs that subsequently process tasks in the same bay. They improve the solutions for the benchmark suite provided in (Kim & Park, 2004) within a limited computation time of two hours by developing a B&C algorithm.

Sammarra et al. (2007) provide a revised model from (Kim & Park, 2004), but the revision also contains the possibility of not detecting crane interference (Bierwirth & Meisel, 2009). In the heuristic approach, they decompose the QCSP into a vehicle routing problem and a scheduling problem. The routing problem, which determines the sequence of tasks on each machine, is solved by tabu search heuristic (Glover, 1989, 1990). The solution of the scheduling problem, which determines the starting handling time (or equivalently the completion time) of the tasks belonging to that route, is generated by a local search technique. Compared to the B&C algorithm by (Moccia et al., 2006), tabu search provides a good compromise between solution quality and computation time.

Ng and Mak (2006) consider the QCSP with each bay comprised of two tasks for unloading and loading, between which a precedence relation is inserted. The bays on vessel are partitioned into a set of non-overlapping zones such that the sharing of workload by cranes is only possible at the border of two zones. An IP model is proposed, and a DP-based heuristic algorithm is used to find near optimal partitions (on average 4.8% above the lower bounds). The unidirectional operation mode is proved to be optimal for any given partition. However, the model does not consider the safety margin.

Bierwirth and Meisel (2009) address a temporal distance between any two tasks permitting a safe movement of the in-between cranes. The temporal distance is computed as a function of the bay positions of tasks, the safety margin, the QC travel time, and the realised task-to-QC assignment. A revised MIP model incorporating the temporal distance is formulated, and the QCSP is solved with a tree-search-based heuristic solution procedure. The authors attribute the fast performance of the heuristic to the exclusive consideration of unidirectional schedules. As a comparison, an optimal schedule for the QCSP with container groups is found with a non-unidirectional schedule.

Tavakkoli-Moghaddam et al. (2009) extend Kim and Park's model by considering the optimal allocation and schedule of a given number of QCs to vessels planned to arrive in the planning horizon. They assume the dynamic planning horizon because in most real cases, many vessels leave the port in a given time horizon and other vessels are placed with them. They formulate an MIP model for the integrated QCASP. A GA is proposed for the

minimisation of the total weighted finishing time of QCs and the total weighted tardiness of vessels. The GA obtains a near optimal solution for complex and large scale problems.

In (Legato, Mazza, & Trunfio, 2008), each bay consists of both unloading/loading (or deck/hold) operations, and each operation amounts to perform a fixed number of container moves which require a non-deterministic number of time slots to be carried out. The model contains non-simultaneous constraints addressed by a minimum distance between QCs, precedence constraints between some tasks and release constraints in which some QCs are not available immediately. They focus on a simulation-based optimisation (SO) approach which embodies the MIP formulation. Two different SO algorithms are tailored to the problem: the SA and the *adaptive balanced explorative and exploitative search* (A-BEES). Preliminary numerical results are presented on real vessel data.

In (Legato, Mazza, & Trunfio, 2010), tasks are numbered according to an increasing order starting from the first bay of the vessel with tasks in each bay being split by the deck/hold positions and the unloading/loading operations. As a common terminal practice, the model also assumes that cranes operate increasing/decreasing ordered bays moving along a single direction rather than back and forward. They focus on a SO model for the QCSP, and the search process for the optimal schedule is accomplished by a SA algorithm. The fitness of the QC schedule is evaluated using the simulation framework.

Meisel (2011) introduces the concept of time windows in the QCSP where the availability of cranes at a vessel is restricted to certain time windows, thus the problem being called the *QCSP with time windows* (QCSPTW). A QCSPTW solution method is derived from extending the unidirectional schedule heuristic proposed in (Bierwirth & Meisel, 2009). The best unidirectional schedule is not necessarily an optimal solution for a QCSPTW instance, but the solutions are of high average quality. The proposed unidirectional schedule heuristic for the QCSPTW is capable of delivering near optimal solutions even for large scale instances, and the average optimality gap is observed below 4% for the investigated instances.

The QCSPTW is also addressed in (Monaco & Sammarra, 2011). The model considers time windows on crane availability, unidirectional operation mode and spatial constraints on QC movements while satisfying precedence and non-simultaneity relations between task pairs.

An MIP model is provided for an exact solution, and a heuristic based on tabu search is presented to get feasible solutions to the problem. Benchmark instances are adopted from (Bierwirth & Meisel, 2009; Kim & Park, 2004; Moccia et al., 2006; Sammarra et al., 2007) to perform numerical analyses.

Chen, Lee, and Cao (2011) investigate the QCSP with container groups at indented berth. They give an MIP model, which improves Kim and Park's model by incorporating the concept of temporal distance described by (Bierwirth & Meisel, 2009). For the heuristic solution approach, decomposition heuristic framework is developed and enhanced by TS.

Legato, Trunfio, and Meisel (2012) provide an enriched model for the QCSP model that cover several aspects of practical terminal operations such as ready times, due dates and non-uniform processing times of QCs. The model also considers an extension of the unidirectional schedule called the independent unidirectionality, where each crane keeps its moving direction but cranes can move into different directions. They refer to the enriched QCSP with non-uniform QCs, time windows and unidirectional crane movement as P^{LTM}, for which an MIP model is formulated. A relaxed model, denoted by P^{rel}, is proposed for obtaining a lower bound on the makespan for P^{LTM} by removing constraints for crane interference and precedence relations. P^{rel} is still $\mathcal{NP}$-hard as it has the same structure as the well-known scheduling problem $R||C_{max}$. They therefore apply a Lagrangian relaxation to solve the model P^{rel}. They employ a B&B algorithm called LTM, which delivers the optimal unidirectional schedule for the QCSP. Four lower bounds and three branching criteria are presented with the goal of speeding up the search. Numerical experiments are carried out on benchmark instances from the literature as well as real instances from the port of Gioia Tauro. The results confirm high quality performance of LTM algorithm with an average optimality gap of about 4.5% on benchmark instances and the makespan on average being 11.8% shorter than that of the practitioners' schedules.

Kaveshgar, Huynh, and Rahimian (2012) adopt the notion of the independent unidirectional schedules discussed by (Legato et al., 2012). Based on the MIP model developed by (Kim & Park, 2004), the computational experiments confirm that the proposed GA yields near optimal solutions for the QCSP.

S. Nguyen, Zhang, Johnston, and Tan (2013) is based on the MIP model developed by (Bierwirth & Meisel, 2009). They present a priority-based schedule construction procedure to generate QC schedules. From this, they propose two hybrid evolutionary computation methods: GA and genetic programming (GP). The difference between them is their representations which decide how priorities of tasks are determined: GA employs a permutation representation to decide the priorities of tasks, and GP represents its individuals as a priority function which is used to calculate the priorities of tasks. Some new best known solutions for the benchmark instances are discovered by using the proposed methods, which are also extended to SO methods.

S. Chung and Chan (2013) formulate a QCSP model under the same settings as in (Bierwirth & Meisel, 2009; Kim & Park, 2004; Moccia et al., 2006; Sammarra et al., 2007). The proposed GA is constructed with a novel workload balancing heuristics, which is capable of considering the loading conditions of different QCs during the reassignment of task-to-QC. The performance of the proposed GA is tested with benchmark instances introduced in (Kim & Park, 2004) and is compared with that of different algorithms in the literature, including (Bierwirth & Meisel, 2009; Kim & Park, 2004; Moccia et al., 2006; Sammarra et al., 2007). The proposed GA is shown to produce some new best solutions in a much shorter computational time.

Expósito-Izquierdo, González-Velarde, Melián-Batista, and Moreno-Vega (2013) discuss an Estimation of Distribution Algorithm (EDA) with local search. This scheme exploits a priori knowledge about the QCSP with the aim of reaching high quality schedules. A restarting strategy to prevent the premature convergence of the search and to guide it toward insufficiently explored regions is developed, and an adaptive stopping criterion to finish the search is presented. Computational experiments demonstrate the performance of the proposed scheme is better than those of previous related approaches.

Nam and Lee (2013) investigate the QCSP where QCs are mounted on floating platforms called mobile harbour (MH). With an objective of minimising the makespan for a set of vessels, the MH operation scheduling problem is to decide the sequencing of tasks, the assignment of MH units to each task, and their docking positions. Being built upon the

constraints commonly used in (Kim & Park, 2004; Moccia et al., 2006; Sammarra et al., 2007), the problem is formulated into an MIP model, and heuristics based on a rule-based algorithm and a random key-based GA algorithm provide practical solutions to the problem. In computational experiments, the latter produces a better-quality solution than the former.

Unsal and Oguz (2013) develop a CP model for the QCSP, which considers safety margins, travel times and precedence relations. The QCSPTW and the integrated QCASP are discussed in the model. Computational results are compared with those of (Legato et al., 2012; Meisel, 2011), and the performance of the CP method is presented for the QCSP with ready times, the QCSPTW and the integrated QCASP.

Chen, Lee, and Goh (2014) note that aside from the unidirectional QCSP model initiated by (J. Liu et al., 2006), the standard MIP models from (Kim & Park, 2004) to (Legato et al., 2012) have always used the binary decision variable $z_{ij}, i, j \in \Omega$ (equals to 1 if the task i is completed before the task j starts; 0, otherwise). Despite its convenient way of using tasks in the binary z_{ij}, the number of tasks is a relatively large number compared to both the number of QCs and the number of bays. Hence, the problem becomes easily intractable as the number of tasks increases. The unidirectional schedule addresses this issue and simplifies the model by partitioning a vessel into several non-overlapping bay areas and assigning one QC to exactly one bay area. They formulate unidirectional MIP models, which are more general and complete by taking account of crane ready times, initial QC positions and the temporal distance of QCs. A relaxed formulation which allows preemption is devised to obtain a tighter lower bound for the unidirectional cluster-based QCSP. Commonly accepted benchmark suites from (Bierwirth & Meisel, 2009; Meisel & Bierwirth, 2011b) are used to assess the performance of the proposed model.

2.4 Other scheduling problems

The QCSP on the basis of container stacks is mainly related to the crane double cycling operation. The research in this stream starts with a series of studies in (Goodchild & Daganzo, 2004, 2005a, 2005b, 2006) which consider the double cycling QCSP within a single hatch. They formulate the problem as a two-machine flow shop scheduling problem where one job

corresponds to one stack and each job has two operations: an unloading operation first and a subsequent loading operation. The problem can be solved optimally with Johnson's rule (S. M. Johnson, 1954).

H. Zhang and Kim (2009) extend the model to the case of multiple hatches, decomposing the whole QC scheduling into two parts: intra-stage optimisation (hatch sequencing) and inter-stage optimisation (QC tasks sequencing in the same hatch). They formulate it as an MIP model, and then develop a local search-based heuristic. The model, however, does not consider non-crossing constraints of QCs as noted by (Ku & Arthanari, 2014). Studies under similar settings are found in (He et al., 2011; D. Wang et al., 2011).

By considering hatch covers, C.-Y. Lee et al. (2014) distinguish their model from previous studies on the double cycling operation. They refer to the model with hatch covers as the general quay crane double cycling problem (QCDCP). They investigate the computational complexity of the general QCDCP, by showing that it can be formulated as a flow shop scheduling problem with series-parallel precedence constraints, which is solvable optimally in polynomial time. They present an optimal algorithm for the general QCDCP, which is a special and simplified version of Sidney's algorithm (Sidney, 1979).

Research based on individual containers considers precedence relations among containers as tasks. The existing literature presents it as the integrated scheduling problem between QCs and vehicles (Cao, Shi, & Lee, 2010; Kaveshgar & Huynh, 2015; Tang, Zhao, & Liu, 2014) or the individual sequencing of container moves (Meisel & Wichmann, 2010).

In the next section, the model assumptions are first presented and the detailed mathematical model is described.

3 Mathematical formulation

3.1 Assumptions

In this paper, we base our model on the assumptions presented in (Lu et al., 2012), which give only the heuristic approach to the scheme *Bay,prmp|move|cross,save|max*(*compl*)

introduced in (Bierwirth & Meisel, 2010). Lu et al. (2012) explain the concepts of contiguous bay range vs. noncontiguous bay range and, correspondingly, contiguous schedule vs. noncontiguous schedule. In real world practice, terminal operators usually adopt the *divide-and-conquer* strategy of partitioning a vessel into several non-overlapping bay areas and assigning one crane to exactly one bay area (Chen et al., 2014), mainly attributed to its planning convenience and its relatively good performance. We follow this practical lead in the formulation of the QCSP. The assumptions based on the preemptive scheduling for the hatch handling operation are shown as follows:

- Tasks are defined on the basis of a ship hatch such that up to two cranes can share the workload of a hatch. The limit of at most two QCs per hatch is a practical assumption in terminal operations. From the computational perspective, this limitation can also reduce the search space than the assumption of no such limit on the number of cranes per hatch;
- The number of containers in each hatch to be served by each QC is left as a decision variable;
- No precedence relations among tasks are needed as the task is based on the hatch;
- Safety distance between QCs is kept at all times;
- Non-crossing constraints of QCs are considered;
- Initial positions of QCs are ignored. That is, a starting position of a QC is set to be the hatch in which it starts processing tasks in the schedule;
- There is a fixed amount of travel time of a QC between two consecutive hatches;
- The unidirectional structure of schedules is kept in the model, where the direction of crane movement is a decision variable. All QCs have the identical moving direction;

We are given a set of h hatches, $H = \{1, \cdots, h\}$, and a set of q QCs, $Q = \{1, \cdots, q\}$. Hatches are arranged in an increasing order of their indices, i.e., hatch 1 is the left-most hatch and hatch h is the right-most hatch of the vessel at the quay. QCs are also arranged along the quay in an increasing order of their relative locations in the direction of increasing hatch indices. We assume that each QC visits each hatch at most once. Once a QC starts the container handling in a hatch, the QC does not stop the processing until all containers assigned to the QC are to be handled. We call each hatch-to-QC assignment an *operation*.

We denote by p_i^k the time required for QC k to perform jobs in hatch i, and by C_i^k the completion time of QC k in hatch i. Then, the processing schedule of an operation, denoted by σ_i^k $(i \in H, k \in Q)$, is given by $[C_i^k - p_i^k, C_i^k)$. A partial order among operations is represented by $\prec$. If $\sigma_i^u \prec \sigma_j^v$, the operation σ_i^u precedes the operation σ_j^v. Correspondingly, σ_i^u is a predecessor of σ_j^v, and σ_j^v is a successor of σ_i^u.

3.2 Hatch-to-QC assignment and the sharing of the workload

In our model, the workload of each hatch is sharable between QCs, thus making the scheduling problem preemptive in the hatch-handling operation. We denote by n_i the number of container handlings in hatch i. By representing the number of containers to process as the proxy for the processing time, Constraint (2.5) ensures that for each hatch, the sum of processing times by every QC is the same as the number of container handlings in that hatch.

$$\sum_{k \in Q} p_i^k = n_i \qquad \forall i \in H \tag{2.5}$$

3.3 Unidirectional schedule

A schedule is called a unidirectional schedule if the QCs do not change the moving direction after the initial repositioning and have identical directions of movement of either from upper to lower bays or vice versa (Bierwirth & Meisel, 2009). Lim et al. (2007) prove that for the QCSP with *complete bays*, there is at least one optimal solution among the unidirectional ones. However, searching the space of unidirectional schedules does not guarantee finding the optimal schedule for the QCSP with *container groups* (Bierwirth & Meisel, 2009). One such example is illustrated in Figure 3.1, where the bidirectional schedule in the left chart is optimal whereas the unidirectional one in the right chart is suboptimal.

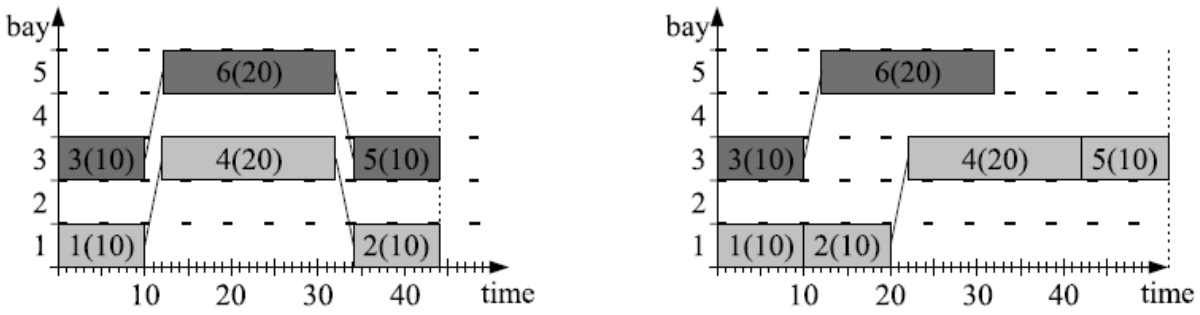


Figure 3.1 Non-unidirectional optimal schedule. *Source*: (Bierwirth & Meisel, 2009)

We introduce a binary variable x_{ij}^k to decide a path which a QC takes along the quay during its service. x_{ij}^k is equal to 1 if and only if QC k moves from hatch i to hatch j; 0, otherwise. Let R denote a binary variable for the direction of QC movements: 1, if QCs travel from lower-indexed hatches to upper-indexed hatches; 0, if QCs travel from upper-indexed hatches to lower-indexed hatches. That a QC has a path through a certain hatch does not mean that any workload in the hatch should be assigned to the QC because the hatch may have no container handlings or the QC may just pass through the hatch without performing any workload. Constraints (2.12)–(2.15) ensure the unidirectionality of QC movements.

$$x_{ij}^k + \sum_{l=1}^{j-1} x_{jl}^k \leq 1 \qquad \forall i,j \in H\ (i<j), \forall k \in Q \tag{2.12}$$

$$x_{ij}^k + \sum_{l=j+1}^{h} x_{jl}^k \leq 1 \qquad \forall i,j \in H\ (i>j), \forall k \in Q \tag{2.13}$$

$$\sum_{k\in Q} \sum_{i=1}^{h-1} x_{i,i+1}^k \leq MR \tag{2.14}$$

$$\sum_{k\in Q} \sum_{i=1}^{h-1} x_{i+1,i}^k \leq M(1-R) \tag{2.15}$$

3.4 Interference constraints

The safety distance δ represents the required number of hatches to lie between adjacent QCs for a safety reason. If $\delta = 1$, for example, QCs can simultaneously operate at the vessel when

they are positioned with at least one hatch away from each other. Moreover, QCs are not allowed to cross each other. Due to the sequential order of QC indices, u should at all times operate below (respectively, above) v to avoid the crane crossing if $u < v$ (respectively, $u > v$). The crossing interference will be incurred when the hatch a QC is scheduled to process next is located across another QC, and the safety interference will be incurred when any two QCs are at any moment located within the safety distance. The QCSP is feasible if and only if for any schedule σ_i^u, σ_j^v ($|i-j| \leq \delta$), σ_i^u and σ_j^v are to be processed separately in time, i.e. $[C_i^u - p_i^u, C_i^u) \cap [C_j^v - p_j^v, C_j^v) = \emptyset$.

Let Ψ denote the set of pairs of hatches that cannot be performed simultaneously. It can be defined as $\Psi = \{(i,j) \in H^2 | \, (|j-i| \leq \delta) \wedge (i \leq j)\}$. Let Θ denote the set of all combinations of hatches and QCs that potentially lead to crane crossings, i.e., hatch i cannot be performed by QC u during the processing time of hatch j by QC v whenever $(i,j,u,v) \in \Theta$. It can be defined as $\Theta = \{(i,j,u,v) \in H^2 \times Q^2 | (i<j) \wedge (u>v)\}$.

We introduce a binary variable s_{ij}^{uv} to represent the separated processing between two operations. s_{ij}^{uv} is equal to 1 if and only if $\sigma_i^u \prec \sigma_j^v$; 0, otherwise. If any two QCs are assigned to a pair of hatches in Ψ or Θ, their separate processing in time is ensured by Constraint (2.18).

$$\sum_{l \in H^0 \setminus \{i\}} x_{li}^u + \sum_{l \in H^0 \setminus \{j\}} x_{lj}^v \leq 1 + s_{ij}^{uv} + s_{ji}^{vu} \qquad (2.18)$$

$$[\forall (i,j) \in \Psi \wedge \, \forall u,v \in Q \; (u<v)] \vee [\forall (i,j,u,v) \in \Theta]$$

3.5 Temporal distance

A weakness with the non-simultaneity condition above is that it does not fully satisfy the requirement for the crane interference because it does not consider the travel time of a QC which at all times should be located at least the safety distance away from any other QC. Meisel (2009) refers to this as the minimum *temporal distance*, which can be computed as a

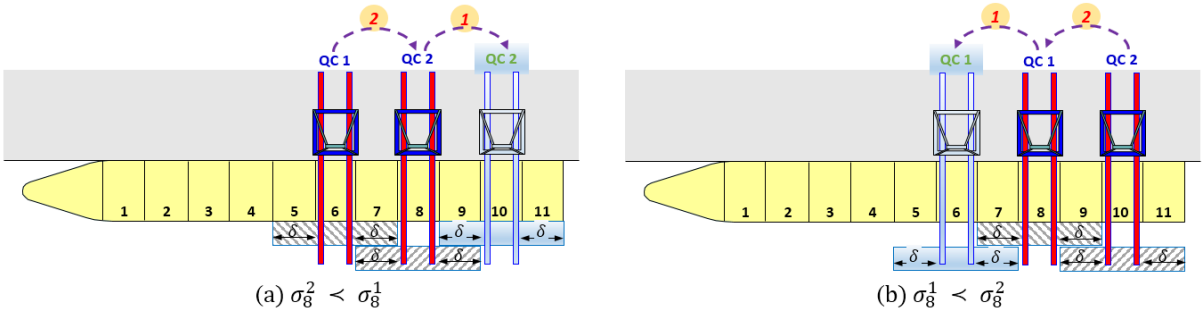
function of the hatch position, the safety distance, the QC travel time and the realised hatch-to-QC assignments. The smallest number of hatches, denoted by δ^{uv}, to lie between any two QCs u and v is calculated as $\delta^{uv} = (\delta + 1)|u - v|$.

The minimum time span to elapse between the processing of two operations σ_i^u and σ_j^v ($i \leq j, u \neq v$) is the minimum travel time of a QC from the safety distance away to the target hatch, complying with interference constraints with another QCs. Let $\hat{t}$ be a fixed constant for a QC's travel time between two consecutive hatches. The minimum temporal distance, Δ_{ij}^{uv}, is then defined as follows, and the feasible QCSP should have the tightened constraint, $[C_i^u - p_i^u, C_i^u + \Delta_{ij}^{uv}) \cap [C_j^v - p_j^v, C_j^v + \Delta_{ij}^{uv}) = \emptyset$.

$$\Delta_{ij}^{uv} = \begin{cases} (i - j + \delta^{uv})\hat{t}, & if\ (u < v)\ \wedge (i - j + \delta^{uv} > 0) \\ (j - i + \delta^{uv})\hat{t}, & if\ (u > v) \wedge (j - i + \delta^{uv} > 0) \\ 0 \end{cases}$$

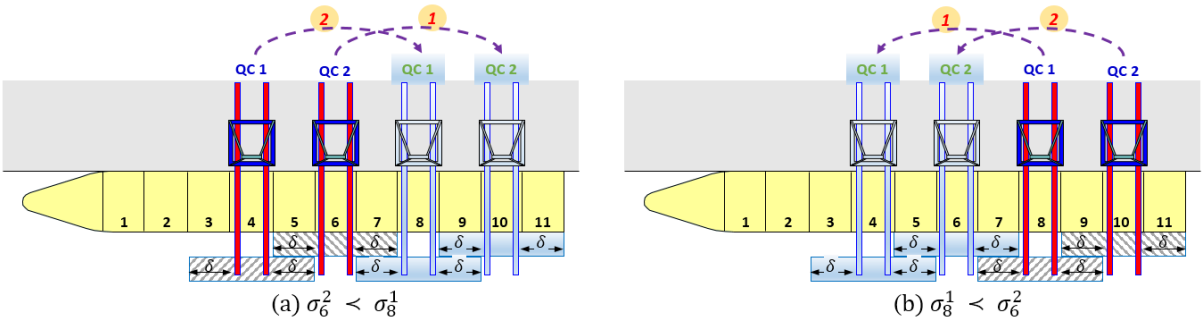
Figure 3.2 illustrates the first case, in which the realised operations are $\sigma_8^1(\sigma_i^u)$ and $\sigma_8^2(\sigma_j^v)$ and they have to be processed separately in time as the hatches of both operations are located within the safety distance. The resulting precedence relationship is either $\sigma_8^2 \prec \sigma_8^1$ or $\sigma_8^1 \prec \sigma_8^2$, represented by Figure 3.2(a) and Figure 3.2(b), respectively. With $\delta = 1$ and $\hat{t}$=1, for example, a temporal distance of two time units should be introduced between the completion of a predecessor and the start of a successor.

Figure 3.3 illustrates the second case, which demonstrates the crane crossings when the above QC (QC 2) is to process the lower hatch (hatch 6) and the lower QC (QC 1) is to process the above hatch (hatch 8). The resulting QC travels are that QC1 needs to be positioned below hatch 5 when σ_6^2 as a predecessor is being processed and QC2 needs to be positioned above hatch 9 while σ_8^1 as a predecessor is being processed. As successors, this incurs the minimum temporal distance of four time units to QC1, as illustrated in Figure 3.3(a), and that of four time units to QC2, as illustrated in Figure 3.3(b), respectively.



(a) $\sigma_8^2 \prec \sigma_8^1$ (b) $\sigma_8^1 \prec \sigma_8^2$

Figure 3.2 Interference (safety distance) between σ_i^u and σ_j^v when $u \prec v$ $(i \leq j)$



(a) $\sigma_6^2 \prec \sigma_8^1$ (b) $\sigma_8^1 \prec \sigma_6^2$

Figure 3.3 Interference (crossing) between σ_i^u and σ_j^v when $u \succ v$ $(i \leq j)$

Let M be a sufficiently large positive constant. Constraints (2.19) and (2.20) then ensure that operations σ_i^u and σ_j^v cannot be performed simultaneously while respecting the temporal distance.

$$C_i^u + \Delta_{ij}^{uv} + p_j^v - C_j^v \leq \left(3 - \sum_{l \in H^0 \setminus \{i\}} x_{li}^u - \sum_{l \in H^0 \setminus \{j\}} x_{lj}^v - s_{ij}^{uv}\right) M$$
$$[\forall (i,j) \in \Psi \wedge \forall u, v \in Q \ (u < v)] \vee [\forall (i,j,u,v) \in \Theta] \tag{2.19}$$

$$C_j^v + \Delta_{ij}^{uv} + p_i^u - C_i^u \leq \left(3 - \sum_{l \in H^0 \setminus \{i\}} x_{li}^u - \sum_{l \in H^0 \setminus \{j\}} x_{lj}^v - s_{ji}^{vu}\right) M$$
$$[\forall (i,j) \in \Psi \wedge \forall u, v \in Q \ (u < v)] \vee [\forall (i,j,u,v) \in \Theta] \tag{2.20}$$

3.6 Mathematical model

With the assumptions and constraints above, we present an MIP model here. First, we give parameters and decision variables for the model. We then give the mathematical formulation for the problem under discussion.

Input data:

h : The number of hatches;

q : The number of QCs;

n_i : The number of container handlings in hatch i;

r_k : The earliest available time of QC k;

$\hat{t}$: The fixed travel time between two consecutive hatches;

δ : The safety distance;

Set of indices:

Q : The set of QCs; $Q = \{1, \cdots, q\}$.

H : The set of hatches; $H = \{1, \cdots, h\}$.

H^0 : $H \cup \{0\}$;

Ψ : The set of pairs of hatches that cannot be performed simultaneously.

$$\Psi = \{(i,j) \in H^2 | \ (|j-i| \leq \delta) \wedge (i \leq j)\}$$

Θ : The set of all combinations of hatches and QCs that potentially lead to crane crossings.

$$\Theta = \{(i,j,u,v) \in H^2 \times Q^2 | (i < j) \wedge (u > v)\}$$

Decision variables:

x_{ij}^k : 1, if QC k moves from hatch i to hatch j; 0, otherwise. Dummy hatch 0 indicates the beginning and the end of the operation in each QC. Thus, when hatch j is the first hatch of QC k, $x_{0,j}^k = 1$. Likewise, when hatch i is the last hatch of QC k, $x_{i,0}^k = 1$.

R : 1, if QCs travel from lower-indexed hatches to upper-indexed hatches; 0, if QCs travel from upper-indexed hatches to lower-indexed hatches.

p_i^k : The time required for QC k to perform jobs in hatch i (in number of containers to process);

s_{ij}^{uv} : 1, if $\sigma_i^u \prec \sigma_j^v$; 0, otherwise;

C_{max} : Makespan;

C^k : The completion time of QC k;

C_i^k : The completion time of QC k in hatch i.

$$\text{minimize } C_{max} \tag{2.1}$$

subject to

$$C_{max} \geq C^k \qquad \forall k \in Q \tag{2.2}$$

$$C^k \geq r_k + \sum_{i \in H} p_i^k + \sum_{i \in H \setminus \{j\}} \sum_{j \in H} x_{ij}^k \, |j - i| \hat{t} \qquad \forall k \in Q \tag{2.3}$$

$$C^k \geq C_i^k \qquad \forall i \in H, \forall k \in Q \tag{2.4}$$

$$\sum_{k \in Q} p_i^k = n_i \qquad \forall i \in H \tag{2.5}$$

$$C_i^k \geq r_k + p_i^k \qquad \forall i \in H, \forall k \in Q \tag{2.6}$$

$$C_i^k + |j-i|\hat{t} + p_j^k - C_j^k \le (1 - x_{ij}^k)M \qquad \forall i,j \in H\ (i \ne j), \forall k \in Q \tag{2.7}$$

$$\sum_{j\in H} x_{0j}^k = 1 \qquad \forall k \in Q \tag{2.8}$$

$$\sum_{i\in H} x_{i0}^k = 1 \qquad \forall k \in Q \tag{2.9}$$

$$\sum_{k\in Q}\sum_{i\in H^0\setminus\{j\}} x_{ij}^k \le 2 \qquad \forall j \in H \tag{2.10}$$

$$\sum_{k\in Q}\sum_{i\in H^0\setminus\{j\}} x_{ij}^k \ge 1 \qquad \forall j \in H \tag{2.11}$$

$$x_{ij}^k + \sum_{l=1}^{j-1} x_{jl}^k \le 1 \qquad \forall i,j \in H\ (i < j), \forall k \in Q \tag{2.12}$$

$$x_{ij}^k + \sum_{l=j+1}^{h} x_{jl}^k \le 1 \qquad \forall i,j \in H\ (i > j), \forall k \in Q \tag{2.13}$$

$$\sum_{k\in Q}\sum_{i=1}^{h-1} x_{i,i+1}^k \le MR \tag{2.14}$$

$$\sum_{k\in Q}\sum_{i=1}^{h-1} x_{i+1,i}^k \le M(1-R) \tag{2.15}$$

$$\sum_{i\in H^0\setminus\{j\}} x_{ij}^k - \sum_{i\in H^0\setminus\{j\}} x_{ji}^k = 0 \qquad \forall j \in H, \forall k \in Q \tag{2.16}$$

$$p_j^k \le M\sum_{i\in H^0\setminus\{j\}} x_{ij}^k \qquad \forall j \in H, \forall k \in Q \tag{2.17}$$

$$\sum_{l\in H^0\setminus\{i\}} x_{li}^u + \sum_{l\in H^0\setminus\{j\}} x_{lj}^v \le 1 + s_{ij}^{uv} + s_{ji}^{vu} \tag{2.18}$$

$$[\forall (i,j) \in \Psi \wedge\ \forall u,v \in Q\ (u < v)] \vee [\forall (i,j,u,v) \in \Theta]$$

$$C_i^u + \Delta_{ij}^{uv} + p_j^v - C_j^v \leq \left(3 - \sum_{l \in H^0 \setminus \{i\}} x_{li}^u - \sum_{l \in H^0 \setminus \{j\}} x_{lj}^v - s_{ij}^{uv}\right) M \quad [\forall (i,j) \in \Psi \wedge \forall u, v \in Q\ (u < v)] \vee [\forall (i,j,u,v) \in \Theta] \tag{2.19}$$

$$C_j^v + \Delta_{ij}^{uv} + p_i^u - C_i^u \leq \left(3 - \sum_{l \in H^0 \setminus \{i\}} x_{li}^u - \sum_{l \in H^0 \setminus \{j\}} x_{lj}^v - s_{ji}^{vu}\right) M \quad [\forall (i,j) \in \Psi \wedge \forall u, v \in Q\ (u < v)] \vee [\forall (i,j,u,v) \in \Theta] \tag{2.20}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall i, j \in H^0\ (i \neq j), \forall k \in Q \tag{2.21}$$

$$s_{ij}^{uv} \in \{0,1\} \qquad \forall (i,j) \in \Psi, \forall u, v \in Q\ (u < v) \text{ or } \forall (i,j,u,v) \in \Theta \tag{2.22}$$

$$R \in \{0,1\} \tag{2.23}$$

$$p_i^k \in Z_{\geq 0} \qquad \forall i \in H, \forall k \in Q \tag{2.24}$$

where M is a sufficiently large positive constant and $Z_{\geq 0}$ is the set of non-negative integers.

The pursued objective given in (2.1) is to minimize the makespan of the vessel operation. The makespan is not necessarily for the operation of a single vessel. The model is flexible enough to capture the makespan of the operation for multiple vessels as long as hatches in those vessels are indexed sequentially along the berth and a proper number of hatches with no containers to handle exist between vessels. Constraint (2.2) defines the makespan as the latest completion time among every QC. The completion time of each QC is defined by Constraints (2.3) and Constraint (2.4). Constraint (2.5) ensures that for each hatch, the sum of processing times by every QC is the same as the number of container handlings in that hatch. The completion time of each QC by each hatch is defined in Constraint (2.6), and Constraint (2.7) gives the definition of x_{ij}^k. Constraints (2.8) and (2.9) ensure that every QC starts at the initial dummy hatch and ends at the final dummy hatch. Constraints (2.10) and (2.11) ensure that

each hatch must be visited by at least one QC and at most two QCs. Constraints (2.12)–(2.15) ensure the unidirectional schedule of each QC. Constraint (2.16) is a flow balance constraint, guaranteeing that each operation is performed in well-defined sequences. Constraint (2.17) ensures that the processing time of each operation is zero if the QC assigned to the operation does not visit the hatch. Constraints (2.18)–(2.20) guarantee that operations σ_i^u and σ_j^v cannot be performed simultaneously when $(i,j) \in \Psi \ \wedge \ u,v \in Q \ (u < v)$ or $(i,j,u,v) \in \Theta$. Finally, (2.21)(2.24) restrict the domains of the decision variables.

4 Relaxation approach

A relaxation of the problem is considered for determining a lower bound on the schedule makespan of a problem instance. Linear programming (LP) is a natural relaxation to apply to integer linear programs because of the availability of very fast routines for solving linear programs. However, the LP relaxation can be a poor approximation of the original problem making enumeration an untenable solution approach (Martin, 2012). In this section, the MIP model is amended to Lagrangian relaxation, which is easier to solve than the original formulation by dualising difficult constraints into the objective function with penalty terms against violations to the dualised constraints.

4.1 Lagrangian relaxation

Lagrangian relaxation has been shown to be an effective approach for solving several types of optimisation problems. The works of (Held & Karp, 1970, 1971) on the traveling salesman problem and Fisher's scheduling algorithms in (Fisher, 1981) provide classical examples of such instances (Sherali & Myers, 1988). The main idea of Lagrangian relaxation is to remove a portion of complicating constraints and put them into the objective function, assigned with weights called Lagrangian multiplier or dual variables. The Lagrangian multiplier or dual variable is a vector of nonnegative weights, and each weight represents a penalty added to a solution that violates the particular inequality constraints. It is well known that dual variables can be interpreted as prices for violating the relaxed constraints. The dual problem is easier to solve and it yields lower bounds (for a minimisation problem) on the optimal value of the original problem. Lagrangian relaxation works best if one can properly identify a small

number of difficult constraints. In the existing literature on the QCSP, interference-related constraints are often relaxed: interference and precedence relations in (Legato et al., 2012), pre-emptive schedules in (Guan et al., 2013), and end bay positioning constraints for middle-numbered QCs and non-crossing constraints in (Theodorou & Diabat, 2014). Our model contains no precedence relations because of the dealing with hatches instead of container groups. Nor does the model assume the pre-emptive schedules. Therefore, we consider the interference constraints (2.18)–(2.20) in our model difficult and thus to be relaxed. The rest of this section develops a Lagrangian dual decomposition approach to solve (2.1).

Let $\lambda^{(1)}$ be a vector variable dualising Constraint (2.18), $\lambda^{(2)}$ be that for Constraint (2.19), and $\lambda^{(3)}$ be that for Constraint (2.20) with a dual variable or a Lagrangian multiplier λ^m_{ijuv} being each element in the vector $\lambda^{(m)}$, i.e., $\lambda^m_{ijuv} \in \{\lambda^m_{ijuv} | \, m \in \{1,2,3\}, \, \forall (i,j) \in \Psi \wedge \forall u,v \in Q \; (u<v) \vee \forall (i,j,u,v) \in \Theta\}$. Then the Lagrangian dual function is

$$L(\lambda) = \text{minimise } C_{max} + \sum\nolimits_{m=1}^{3} \sum\nolimits_{\forall (i,j)\in\Psi \wedge \forall u,v\in Q \; (u<v)\vee\forall (i,j,u,v)\in\Theta} \lambda^m_{ijuv} g^m_{ijuv}$$

where

$g^1_{ijuv} = \sum_{l\in H^0\setminus\{i\}} x^u_{li} + \sum_{l\in H^0\setminus\{j\}} x^v_{lj} - s^{uv}_{ij} - s^{vu}_{ji} - 1$ and

$$g^2_{ijuv} = C^u_i + \Delta^{uv}_{ij} + p^v_j - C^v_j + M\sum_{l\in H^0\setminus\{i\}} x^u_{li} + M\sum_{l\in H^0\setminus\{j\}} x^v_{lj} + Ms^{uv}_{ij} - 3M \qquad (2.25)$$

and

$g^3_{ijuv} = C^v_j + \Delta^{uv}_{ij} + p^u_i - C^u_i + M\sum_{l\in H^0\setminus\{i\}} x^u_{li} + M\sum_{l\in H^0\setminus\{j\}} x^v_{lj} + Ms^{vu}_{ji} - 3M$

and

subject to (2.1)–(2.17) and (2.21)–(2.24).

Off-the-shelf solvers for convex optimisation problems may sometimes hit an error if the objective function contains too many terms. To reduce the large number of terms in the dual objective function attributed to a number of variables in the relaxed constraints, we introduce

in (2.25) the equality variable g_{ijuv}^{m} which replace terms in Constraints (2.18)–(2.20). For any $\lambda_{ijuv}^{m} \geq 0$, $L(\lambda)$ forms a lower bound on the original problem, as $\lambda_{ijuv}^{m} g_{ijuv}^{m} \leq 0$. That is, we have $L(\lambda) \leq C_{max}^{*}$ where C_{max}^{*} is the optimal solution in (2.1). The task is therefore to find

$$L^{*} = \max_{\lambda \geq 0} L(\lambda) \tag{2.26}$$

which can provide lower bounds tighter than or equal to those from a LP relaxation. That is,

$$z_{LP} \leq L^{*} \leq C_{max}^{*} \tag{2.27}$$

where z_{LP} is the optimal solution of the LP relaxation problem.

4.2 Subgradient method

It can be shown that the dual function $L(\lambda)$ is a piecewise linear function. Therefore, solving $L(\lambda)$ is a non-differentiable convex optimisation problem since the function is non-differentiable at a finite set of λ values. A successful technique for this problem is the subgradient method (N. Z. Shor, 1985), which is an iterative algorithm to solve a non-differentiable convex function. Since (2.25) is a convex optimization subject to linear constraints, the strong duality theorem in (Bertsekas, 1999) (see pp. 504-505) can be applied. Assuming that primal function (2.1) is feasible and its optimal value is finite, then there is no duality gap and there exists at least one optimal dual vector. In brief, assuming the existence of an optimal primal solution, there exist $\lambda^{*} \geq 0$ satisfying

$$L(\lambda^{*}) = \max_{\lambda \geq 0} L(\lambda) = C_{max}^{*} . \tag{2.28}$$

The maximisation of dual function can be done via a subgradient method. Let ξ denote the subgradient of the dual function (2.25). The subgradient update of λ then is

$$\lambda^{(k+1)} = \left[\lambda^{(k)} - \alpha_k \xi^{(k)}\right]^+ \tag{2.29}$$

where $[x]^+ = \max(x, 0)$, $\lambda^{(k)}$ is the k^{th} iterate, and $\alpha_k > 0$ is the k^{th} step size.

The subgradient method for non-differentiable functions is similar to its gradient counterpart for differentiable functions, but with several notable exceptions (Boyd et al., 2003). For example, the subgradient method uses step lengths that are fixed ahead of time, instead of an exact or approximate line search as in the gradient method. Unlike the ordinary gradient method, the subgradient method is not necessarily a descent method, i.e., moving in the direction of a subgradient does not necessarily lead to an improvement in the dual function. Since the last iterating solution in the subgradient method may not be the best one found thus far, it is needed to keep track of the best known solution while iterating.

Let θ_j denote a positive number between 0 and 2 for step size adjustor. Often the θ_j are determined by setting $\theta_0 = 2$ and halving θ_j whenever $L(\lambda)$ has failed to improve for some fixed number of iterations. Algorithm 2.1 summarises the procedure for the subgradient algorithm used in our implementation. As noted by (Martin, 2012), getting a subgradient algorithm to work in practice requires a considerable amount of *tweaking* the parameters. The best parameter values are problem-specific.

Algorithm 2.1 Subgradient Optimisation Algorithm

Step 1: Input

An upper bound L^*, which can be provided by any feasible heuristic solution. For convenience, we assign the total number of container handlings to L^*;

A lower bound $LB := -\infty$;

A sufficiently small positive number ε, which is arbitrarily set to 1.0-e09;

A parameter $noimprovement\_limit$, which is arbitrarily set to 1;

A parameter $max\_iteration$, which is arbitrarily set to 300.

Step 2: Initialisation

Set the iteration counter $j := 0$;

Set the counter for no improvement $noimprovement := 0$;

Set the initial Lagrangian multipliers $\lambda^{(0)} := 0.001$;

Set the initial value $\theta_0 := 2$ for the step size adjustment.

Step 3: Solve the Lagrangian function $L(\lambda^{(j)})$

if $L(\lambda^{(j)}) > LB + \varepsilon$, then $LB = L(\lambda^{(j)})$;

Otherwise,

set $noimprovement := noimprovement + 1$

if $noimprovement > noimprovement\_limit$, then

set $\theta_j := \theta_j / 2$ and $noimprovement := 0$.

Step 4: Calculate step size

Let γ_j be the vector for values of g_{ijuv}^m from Step 3;

Set the step size $t_j := \theta_j(L^* - L(\lambda^{(j)})) \Big/ \|\gamma_j\|^2$.

Step 5: Update λ

For each element in the dual vector λ, update $\lambda^{(j)} := \max\{0, \lambda^{(j)} + t_j\gamma_j\}$.

Step 6: Stop condition

if $\|\lambda^{(j+1)} - \lambda^{(j)}\| < \varepsilon$ or $j = max\_iteration$, then stop;

Otherwise, $j := j + 1$ and go to Step 3.

5 Computational experiments

For computational analysis, our approach is executed in a virtual OS of Windows Server 2012 R2 running on Intel Xeon X5680 CPU (3.33GHz) and 16GB of RAM. As a commercial solver for solving the proposed MIP model and its Lagrangian counterpart, Gurobi 6.0 is used with the implementation of Java interface. The memory option for the Java virtual machine (JVM 1.7) is set to a maximum of 8GB, i.e., '–Xmx8G'.

5.1 Benchmark instances

The benchmark suite (*KP*) provided by (Kim & Park, 2004) contain nine instance sets with each set containing 10 instances. The problem size of each instance ranges from 10 to 50 tasks and two to six QCs, as summarised in Table 5.1. As listed in Table 5.2, *KP* instances are most popularly used for the benchmark instances of the QCSP in the existing literature.

Table 5.1 QCSP benchmark instance sets in (Kim & Park, 2004)

Instance set	A	B	C	D	E	F	G	H	I
# tasks	10	15	20	25	30	35	40	45	50
# QCs	2	2	3	3	4	4	5	5	6

Table 5.2 Benchmark instances and algorithms for the QCSP in the literature

Literature	**Benchmark instances**
(Lim, Rodrigues, Xiao, et al., 2004)	Two random instance groups HC, PTS, SWO and SWO+L
(Kim & Park, 2004)	(Kim & Park, 2004) instances B&B, GRASP
(Ng & Mak, 2006)	Random instances C++
(Zhu & Lim, 2006)	Three random instance groups CPLEX, B&B, SA
(J. Liu et al., 2006)	Random instances (50 instances) CPLEX, Decomposition heuristics
(Lim et al., 2007)	(Zhu & Lim, 2006) instances CPLEX, SA
(Moccia et al., 2006)	(Kim & Park, 2004) instances Real data from the MCT at Gioia Tauro CPLEX, B&C
(Sammarra et al., 2007)	(Kim & Park, 2004) instances TS
(D.-H. Lee et al., 2008b)	Random instances CPLEX, GA
(Bierwirth & Meisel, 2009)	(Kim & Park, 2004) instances UDS heuristic
(Meisel, 2011)	Random instances (280 instances) CPLEX, UDSTW heuristic
(Chen et al., 2011)	Random instances Decomposition heuristics, TS
(Meisel & Bierwirth, 2011b)	(Meisel & Bierwirth, 2011b) instances UDS heuristic
(Lu et al., 2012)	(Meisel & Bierwirth, 2011b) instances
(Legato et al., 2012)	(Kim & Park, 2004) instances (Meisel & Bierwirth, 2011b) instances Real data from the MCT at Gioia Tauro

	LTM, Lower bound by Lagrangian in Java
(S. H. Chung & Choy, 2012)	(Kim & Park, 2004) instances Compare with (Kim & Park, 2004; Moccia et al., 2006; Sammarra et al., 2007) GA
(Kaveshgar et al., 2012)	(Meisel & Bierwirth, 2011b) instances GA, MATLAB
(S. Chung & Chan, 2013)	(Kim & Park, 2004) instances Java (WBGA)
(S. Nguyen et al., 2013)	(Kim & Park, 2004) instances (Meisel & Bierwirth, 2011b) instances Compare with (Bierwirth & Meisel, 2009; Kim & Park, 2004; Moccia et al., 2006; Sammarra et al., 2007) Hybrid GA, Hybrid GP (in Java)
(Guan et al., 2013)	Random instances Compare with (Lim, Rodrigues, Xiao, et al., 2004; Zhu & Lim, 2006) CPLEX, Lagrangian
(Unsal & Oguz, 2013)	Random instances designed by (Meisel & Bierwirth, 2011b) IBM CP Optimizer & CPLEX
(Tang et al., 2014)	Random instances CPLEX, GA (D.-H. Lee et al., 2008b), PSO
(Chen et al., 2014)	(Bierwirth & Meisel, 2009) instances from (Kim & Park, 2004) (Meisel & Bierwirth, 2011b) instances Real data from the MCT at Gioia Tauro CPLEX
(Legato & Trunfio, 2014)	(Kim & Park, 2004) instances (Meisel & Bierwirth, 2011b) instances Proposal in (Legato et al., 2012) LTM & USE heuristic in Java
(Fu et al., 2014)	Random instances GAMS
(Theodorou & Diabat, 2014)	Random instances Lagrangian heuristic
(Guo, Cheng, & Wang, 2014)	(Kim & Park, 2004) instances Compare with (Bierwirth & Meisel, 2009; S. H. Chung & Choy, 2012; Kim & Park, 2004; Moccia et al., 2006; Sammarra et al., 2007) MATLAB (MEGO)
(Kaveshgar & Huynh, 2015)	(Meisel & Bierwirth, 2011b)-generated instances CPLEX, GA (MATLAB)

More recently, (Meisel & Bierwirth, 2011b) observe that the average number of tasks per bay in *KP* instances is consistently one in all instances. To test with more diverse settings, they provide a benchmark generator program which can generate test instances under a controllable task-per-bay ratio. Furthermore, their benchmark generator reflects many other practical features such as the loading capacity of bays, variable safety margins of cranes, and heterogeneous container distributions within a vessel. The recent literature shows that these new benchmarks obtain acceptance in the community, see e.g. (Chen et al., 2014; Kaveshgar et al., 2012; Legato et al., 2012; Lu et al., 2012; Unsal & Oguz, 2013). It is worth noting that according to the computational experiments done by (Chen et al., 2014), LTM method proposed in (Legato et al., 2012) is significantly faster than the unidirectional schedule (UDS) heuristic proposed in (Bierwirth & Meisel, 2009) by reducing the computational time down to 14% for certain instance sets. At the time of this writing, therefore, the LTM method appears to be the state-of-the-art method to handle the unidirectional cluster-based QCSP.

For the instances of the QCSP with bays, Lim, Rodrigues, Xiao, et al. (2004) and Zhu and Lim (2006) report that they have randomly generated such instances in their experiment, but their instances were not available for reuse. Since *KP* instances adopt the granularity of only container group, we modify them to be applicable to our experiment. We do this by grouping the processing times of tasks into those of hatches. Table 5.3 shows how *KP* instances can be formed with this way of modification.

Table 5.3 Modified test instances in the number of handlings per hatch (*k*13-*k*22 in *KP* instances)

hatch \ instance	*k*13	*k*14	*k*15	*k*16	*k*17	*k*18	*k*19	*k*20	*k*21	*k*22
1		22	70	1	64	1	50			34
2	87	48	53	49		17	43	1	23	
3	62	41		22		39	89		37	99
4		46		3	31	25			51	77
5	3	79	44	28	19			32	40	60
6	37					44	49	50	36	21
7	58	10	29		21	30	50	13	56	
8			62	6	79	8		59	25	
9		51		27	8	13	2	59	32	51
10	19	50	67	51	61	55	53	37		

5.2 Test results discussion

The model setting similar to ours, characterised by the unidirectional and preemptive schedules with a bay being the granularity of a task, can be found only from (Lu et al., 2012), in which QCs can work in contiguous bay range and can share workloads of a hatch. As with our approach, the number of containers in each hatch is left as a decision variable. They provide only a heuristic solution to the specified problem. The main contribution of our study, in fact, is in providing an exact model under this setting because once we have an exact model, several ways of relaxations, including the Lagrangian relaxation included in this paper, can further be investigated in the follow-up research.

When considering the computational results of testing *KP* instances, three main differences from the previous studies should be noted. First, the assumption for preemptive schedules in our study is likely to result in the makespan less than or equal to that of the non-preemptive counterparts. Second, the assumption for unidirectional schedules is likely to result in the makespan greater than or equal to that of the non-unidirectional counterparts. Lastly, under the unidirectional assumption it is unavoidable to relax the assumption of QC starting positions, thus our model leaves the QC starting positions as decision variables instead of being given by instance parameters. The last relaxation is likely to result in less than or equal to that of the non-preemptive counterparts. Therefore, it is expected that the makespan-reducing effect by the first and the third difference will be offset by the makespan-inflating effect by the second difference. Thus, the overall effect of the above differences to the optimal makespan may be instance-specific, i.e., with our model settings, some instances may result in shorter makespans than in the cluster-based QCSP while other instances may result in the same or longer makespans.

Table 5.4 Comparison between the cluster-based QCSP and our model using KP instances

No	Cluster-based QCSP		This paper		Ratio	LB_{lag}	Gap_{lag}	LB_{lp}	Gap_{lp}
	f_{best} (*LB*) [a]	Cplex (s.)	f_{opt} [b]	Gurobi (s.)	[b]/[a]	[c]	[b]/[c]-1	[d]	[b]/[d]-1
*k*13	453	1	453	3	100.0%	414	9.4%	411	10.2%
*k*14	546	1	534	0	97.8%	534	0.0%	532.5	0.3%
*k*15	513	1	501	0	97.7%	501	0.0%	499.5	0.3%

*k*16	312	1	294	2	94.2%	294	0.0%	292.5	0.5%
*k*17	453	1	438	0	96.7%	438	0.0%	436.5	0.3%
*k*18	375	1	363	0	96.8%	363	0.0%	361.5	0.4%
*k*19	543	1	519	0	95.6%	519	0.0%	516	0.6%
*k*20	399	1	390	0	97.7%	390	0.0%	388.5	0.4%
*k*21	465	1	465	0	100.0%	465	0.0%	462	0.6%
*k*22	540	1	534	2	98.9%	528	1.1%	525	1.7%
*k*23	576	3	567	1	98.4%	567	0.0%	565.5	0.3%
*k*24	666	7	660	2	99.1%	660	0.0%	657	0.5%
*k*25	738	235	723	1	98.0%	723	0.0%	721.5	0.2%
*k*26	639	7	633	2	99.1%	633	0.0%	630	0.5%
*k*27	657	253	648	1	98.6%	648	0.0%	646.5	0.2%
*k*28	531	236	519	1	97.7%	519	0.0%	517.5	0.3%
*k*29	807	897	798	0	98.9%	798	0.0%	796.5	0.2%
*k*30	891	3	888	1	99.7%	888	0.0%	885	0.3%
*k*31	570	69	549	3	96.3%	549	0.0%	546	0.5%
*k*32	591	25	588	1	99.5%	588	0.0%	585	0.5%
*k*33	603	750	597	29	99.0%	597	0.0%	594.99	0.3%
*k*34	717	2867	711	36	99.2%	711	0.0%	708	0.4%
*k*35	684	-	675	175	98.7%	675	0.0%	671.01	0.6%
*k*36	678	531	669	58	98.7%	669	0.0%	666.99	0.3%
*k*37	510	3124	504	23	98.8%	504	0.0%	501.99	0.4%
*k*38	618 (614)	-	603	24	97.6%	603	0.0%	600.99	0.3%
*k*39	513	3560	501	43	97.7%	501	0.0%	498	0.6%
*k*40	564	-	552	36	97.9%	552	0.0%	549.99	0.4%
*k*41	588 (585)	-	582	47	99.0%	582	0.0%	579	0.5%
*k*42	573	1659	573	1463	100.0%	552	3.8%	549.99	4.2%
*k*43	876 (860)	-	861	68	98.3%	861	0.0%	858.99	0.2%
*k*44	822 (821)	-	813	91	98.9%	813	0.0%	809.01	0.5%
*k*45	834 (825)	-	822	132	98.6%	822	0.0%	818.01	0.5%
*k*46	690	-	681	72	98.7%	681	0.0%	678.99	0.3%
*k*47	792	-	786	67	99.2%	786	0.0%	783.99	0.3%
*k*48	639 (629)	-	621	43	97.2%	621	0.0%	620.01	0.2%
*k*49	894 (880)	-	879	77	98.3%	879	0.0%	876.99	0.2%
*k*50	741 (727)	-	732	58	98.8%	732	0.0%	729.99	0.3%
*k*51	798 (778)	-	783	52	98.1%	783	0.0%	780.99	0.3%
*k*52	960 (942)	-	948	99	98.8%	948	0.0%	945	0.3%

In Table 5.4, we summarise the computational results for *KP* instances between *k*13 and *k*52, all of which are tractable by our approach. The results for the cluster-based QCSP are taken from the summary of (Meisel & Bierwirth, 2011b), which suggest that among the 40 instances, optimal solutions are known for only 30 instances. The lower bounds in the parenthesis indicate that the remaining 10 instances are still open. In contrast, our model approach yields optimal solutions in all 40 instances within reasonable computation times. Both solvers (Cplex 11 and Gurobi 6) are applied with a runtime limit of one hour, and it is clear that our approach reaches optimal solutions much faster than the cluster-based QCSP. Two lower bounds by Lagrangian relaxation (LB_{lag}) and LP relaxation (LB_{lp}) are provided for the optimal solutions (f_{opt}) of our approach, and $LB_{lp} \leq LB_{lag} \leq f_{opt}$ is verified for each instance. The column for Gap_{lag}, which represents the difference between the optimal solution of our approach and the lower bound by Lagrangian relaxation, is calculated by $\left(f_{opt}/LB_{lag} - 1\right) \times 100\%$, and the column for Gap_{lp}, which represents the difference between the optimal solution of our approach and the lower bound by LP relaxation, is calculated by $\left(f_{opt}/LB_{lp} - 1\right) \times 100\%$. It is notable that within a runtime limit of one hour, 37 instances out of 40 have reached optimal using the subgradient optimisation technique described in Algorithm 2.1. The table also provides the ratios of the makespan of our approach to the cluster-based makespan, and these ratios are less than or equal to 100% in all listed instances. This indicates that our model approach is very likely to dominate the cluster-based QCSP with respect to the makespan. The findings can be summarised as follows:

- By using lower granularity due to preemption, better workload balance among QCs can be achieved;
- That QC starting position is left as decision variable improves the solution quality;
- The improved solution quality empowered by the above points dominates the counteractive restriction of the unidirectional schedule at least for the 40 tested instances.

For larger instance sets (*E–I* groups of *KP* instances), the lower bounds could not be obtained by Lagrangian relaxation with a runtime limit of one hour, which means that the relaxation of interference-related constraints is not enough for the model to be tractable for large scale

instances. In contrast, Table 5.5 shows that lower bounds by LP relaxation can easily be obtained even for large scale instances. The f_{best} solutions for the cluster-based QCSP are taken from (Legato et al., 2012), which provide the most recent state of knowledge regarding *KP* instances. Again, the ratio represented in the last column of this table is only for a reference because our model setting is more relaxed than that of the cluster-based QCSP, thus the lower bounds by LP relaxation in our approach are presumably lower than the lower bounds in the cluster-based QCSP.

Table 5.5 Results of LP relaxation in *KP* instances (sets E-I)

No	Cluster-based		LP relaxation		Ratio	No	Cluster-based		LP relaxation		Ratio
	LB [a]	f_{best}	*LB* [b]	cpu (s)	[b]/[a]		*LB* [c]	f_{best}	*LB* [d]	cpu (s)	[d]/[c]
E	30 tasks, 4 QCs					F	35 tasks, 4 QCs				
*k*53	220.00	717	659.25	8	91.9%	*k*63	310.00	948	930	13	98.1%
*k*54	252.00	774	759	8	98.1%	*k*64	239.33	741	720	13	97.2%
*k*55	222.33	684	669.75	8	97.9%	*k*65	273.67	837	821.25	14	98.1%
*k*56	223.33	690	670.5	6	97.2%	*k*66	303.33	924	909.75	13	98.5%
*k*57	228.00	705	685.5	8	97.2%	*k*67	287.00	882	862.5	14	97.8%
*k*58	256.33	786	770.25	9	98.0%	*k*68	317.00	963	953.25	16	99.0%
*k*59	223.00	687	672.75	7	97.9%	*k*69	262.33	807	787.5	13	97.6%
*k*60	255.33	783	767.25	10	98.0%	*k*70	313.00	957	940.5	14	98.3%
*k*61	206.67	639	620.25	8	97.1%	*k*71	271.33	834	816	11	97.8%
*k*62	277.00	837	831.75	8	99.4%	*k*72	242.33	744	726.75	12	97.7%
G	40 tasks, 5 QCs					H	45 tasks, 5 QCs				
*k*73	280.00	870	841.8	41	96.8%	*k*83	309.33	948	928.8	77	98.0%
*k*74	275.00	843	827.4	49	98.1%	*k*84	294.00	897	883.8	84	98.5%
*k*75	220.00	675	663.6	50	98.3%	*k*85	317.33	972	952.8	88	98.0%
*k*76	276.67	852	829.8	45	97.4%	*k*86	263.67	816	792.6	67	97.1%
*k*77	226.33	699	678.6	51	97.1%	*k*87	282.00	867	846	66	97.6%
*k*78	207.67	642	624.6	42	97.3%	*k*88	249.00	768	747.6	91	97.3%
*k*79	240.67	744	722.4	41	97.1%	*k*89	275.00	843	827.4	67	98.1%
*k*80	241.33	750	724.2	51	96.6%	*k*90	343.00	1053	1030.8	86	97.9%
*k*81	237.00	738	711.6	40	96.4%	*k*91	271.33	837	817.2	67	97.6%
*k*82	233.67	717	701.4	43	97.8%	*k*92	292.00	897	878.4	65	97.9%
I	50 tasks, 6 QCs										
*k*93	264.00	810	792	270	97.8%						

*k*94	256.00	786	769.5	217	97.9%
*k*95	268.67	834	807.50	238	96.8%
*k*96	261.00	801	785.50	229	98.1%
*k*97	231.00	717	694.5	282	96.9%
*k*98	239.00	735	717.99	184	97.7%
*k*99	274.33	852	824.49	197	96.8%
*k*100	286.00	876	858.99	250	98.1%
*k*101	257.00	813	773.00	193	95.1%
*k*102	291.33	897	875.00	222	97.5%

6 Conclusion

The QCSP is one of the core operations in a marine container terminal, consisting of the sequencing and scheduling of the discharging and loading operations of containers and the assignment of QCs. This paper focuses on the unidirectional and preemptive schedule for the QCSP with bays as tasks. The model is formulated as an MIP model and its relaxation is proposed to enable further computational analysis for the QCSP with the unidirectional and preemptive schedules. The exact method is already tenable for small- and medium- scale instances due to its relaxed but still realistic assumptions, i.e., unidirectional and preemptive schedules. The proposed model is tested by solving a set of well-known benchmarking problems. By comparing the results obtained by the proposed method and the ones obtained by other well-known existing approaches, it is found that the proposed model outperforms other methods in small scale instances and obtains better results in some medium sized instances as well. Future research needs to address more advanced heuristic techniques using the assumptions made in this study.

Acknowledgements

Authors thank Kap-Hwan Kim (Pusan National University, Korea) and Young-Man Park (Korea Naval Academy, Korea) for providing the benchmark suite.

References

Bertsekas, D. P. (1999). *Nonlinear programming* (2nd ed. ed.). Belmont, MA: Athena Scientific.

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.

Bierwirth, C., & Meisel, F. (2014). A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*

Boyd, S., Xiao, L., & Mutapcic, A. (2003). *Subgradient Methods*. Stanford University: Retrieved from https://web.stanford.edu/class/ee392o/subgrad_method.pdf

Boysen, N., Emde, S., & Fliedner, M. (2012). Determining crane areas for balancing workload among interfering and noninterfering cranes. *Naval Research Logistics (NRL), 59*(8), 656-662.

Cao, J., Shi, Q., & Lee, D.-H. (2010). Integrated quay crane and yard truck schedule problem in container terminals. *Tsinghua Science & Technology, 15*(4), 467-474.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2013). Seaside operations in container terminals: Literature overview, trends, and research directions. *Flexible Services and Manufacturing Journal*, 1-39.

Chen, J. H., Lee, D.-H., & Cao, J. X. (2011). Heuristics for quay crane scheduling at indented berth. *Transportation Research Part E: Logistics and Transportation Review, 47*(6), 1005-1020.

Chen, J. H., Lee, D.-H., & Goh, M. (2014). An effective mathematical formulation for the unidirectional cluster-based quay crane scheduling problem. *European Journal of Operational Research, 232*(1), 198-208.

Choo, S., Klabjan, D., & Simchi-Levi, D. (2010). Multiship crane sequencing with yard congestion constraints. *Transportation Science, 44*(1), 98-115.

Chung, S., & Chan, F. T. (2013). A workload balancing genetic algorithm for the quay crane scheduling problem. *International Journal of Production Research, 51*(16), 4820-4834.

Chung, S. H., & Choy, K. L. (2012). A modified genetic algorithm for quay crane scheduling operations. *Expert Systems with Applications, 39*(4), 4213-4221.

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

Diabat, A., & Theodorou, E. (2014). An integrated quay crane assignment and scheduling problem. *Computers & Industrial Engineering, 73*, 115-123.

Expósito-Izquierdo, C., González-Velarde, J. L., Melián-Batista, B., & Moreno-Vega, J. M. (2013). Hybrid estimation of distribution algorithm for the quay crane scheduling problem. *Applied Soft Computing, 13*(10), 4063-4076.

Fisher, M. L. (1981). The Lagrangian relaxation method for solving integer programming problems. *Management science, 27*(1), 581-615.

Fu, Y.-M., Diabat, A., & Tsai, I.-T. (2014). A multi-vessel quay crane assignment and scheduling problem: Formulation and heuristic solution approach. *Expert Systems with Applications, 41*(15), 6959-6965.

Glover, F. (1989). Tabu search-part I. *ORSA Journal on computing, 1*(3), 190-206.

Glover, F. (1990). Tabu search-part II. *ORSA Journal on computing, 2*(1), 4-32.

Goodchild, A. V., & Daganzo, C. F. (2004). *Reducing ship turn-around time using double-cycling*. Institue for Transportation Studies Research Report (UCB).Retrieved from http://escholarship.org/uc/item/86r4p6sc

Goodchild, A. V., & Daganzo, C. F. (2005a). *Crane double cycling in container ports: effect on ship dwell time*. Institue for Transportation Studies Research Report (UCB).Retrieved from http://escholarship.org/uc/item/9qp7p7jq

Goodchild, A. V., & Daganzo, C. F. (2005b). *Performance Comparison of Crane Double Cycling Strategies*: Institute of Transportation Studies, University of California, Berkeley.

Goodchild, A. V., & Daganzo, C. F. (2006). Double-cycling strategies for container ships and their effect on ship loading and unloading operations. *Transportation Science, 40*(4), 473-483.

Guan, Y., Yang, K.-H., & Zhou, Z. (2013). The crane scheduling problem: models and solution approaches. *Annals of Operations Research, 203*(1), 119-139.

Guo, P., Cheng, W., & Wang, Y. (2014). A modified generalized extremal optimization algorithm for the quay crane scheduling problem with interference constraints. *Engineering Optimization, 46*(10), 1411-1429.

Han, X.-l., Lu, Z.-q., & Xi, L.-f. (2010). A proactive approach for simultaneous berth and quay crane scheduling problem with stochastic arrival and handling time. *European Journal of Operational Research, 207*(3), 1327-1340.

He, X., Wang, S., & Zheng, J. (2011). *A hybrid heuristic algorithm for integrated large-capacity quay crane scheduling problem.* Paper presented at the Computer Research and Development (ICCRD), 2011 3rd International Conference on.

Held, M., & Karp, R. M. (1970). The traveling-salesman problem and minimum spanning trees. *Operations research, 18*(6), 1138-1162.

Held, M., & Karp, R. M. (1971). The traveling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming, 1*(1), 6-25.

Henesey, L. E. (2006). *Multi-agent systems for container terminal management.* (PhD thesis), Blekinge Institute of Technology, Karlskrona, Sweden.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval research logistics quarterly, 1*(1), 61-68.

Kaveshgar, N., & Huynh, N. (2015). Integrated quay crane and yard truck scheduling for unloading inbound containers. *International Journal of Production Economics, 159*, 168-177.

Kaveshgar, N., Huynh, N., & Rahimian, S. K. (2012). An efficient genetic algorithm for solving the quay crane scheduling problem. *Expert Systems with Applications, 39*(18), 13108-13117.

Kenndy, J., & Eberhart, R. (1995). *Particle swarm optimization.* Paper presented at the Proceedings of IEEE International Conference on Neural Networks.

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

Ku, D., & Arthanari, T. S. (2014). On double cycling for container port productivity improvement. *Annals of Operations Research*, 1-16. 10.1007/s10479-014-1645-z

Lee, C.-Y., Liu, M., & Chu, C. (2014). Optimal Algorithm for the General Quay Crane Double-Cycling Problem. *Transportation Science*

Lee, D.-H., & Chen, J. H. (2010). An improved approach for quay crane scheduling with non-crossing constraints. *Engineering Optimization, 42*(1), 1-15.

Lee, D.-H., Chen, J. H., & Cao, J. X. (2011). Quay crane scheduling for an indented berth. *Engineering Optimization, 43*(9), 985-998.

Lee, D.-H., & Qiu Wang, H. (2010). Integrated discrete berth allocation and quay crane scheduling in port container terminals. *Engineering Optimization, 42*(8), 747-761.

Lee, D.-H., & Wang, H. Q. (2010). An approximation algorithm for quay crane scheduling with handling priority in port container terminals. *Engineering Optimization, 42*(12), 1151-1161.

Lee, D.-H., Wang, H. Q., & Miao, L. (2007). An approximation algorithm for quay crane scheduling with non-interference constraints in port container terminals. *Tristan VI, Phuket, June*, 10-15.

Lee, D.-H., Wang, H. Q., & Miao, L. (2008a). Quay crane scheduling with handling priority in port container terminals. *Engineering Optimization, 40*(2), 179-189.

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

Legato, P., Mazza, R. M., & Trunfio, R. (2008). *Simulation-based optimization for the quay crane scheduling problem.* Paper presented at the Simulation Conference, 2008. WSC 2008. Winter.

Legato, P., Mazza, R. M., & Trunfio, R. (2010). Simulation-based optimization for discharge/loading operations at a maritime container terminal. *OR spectrum, 32*(3), 543-567.

Legato, P., & Trunfio, R. (2014). A local branching-based algorithm for the quay crane scheduling problem under unidirectional schedules. *4OR, 12*(2), 123-156.

Legato, P., Trunfio, R., & Meisel, F. (2012). Modeling and solving rich quay crane scheduling problems. *Computers & Operations Research, 39*(9), 2063-2078.

Lieberman, R., & Turksen, I. (1981). Crane scheduling problems. *AIIE transactions, 13*(4), 304-311.

Lieberman, R., & Turksen, I. (1982). Two-operation crane scheduling problems. *IIE Transactions, 14*(3), 147-155.

Lim, A., Rodrigues, B., Xiao, F., & Zhu, Y. (2002). *Crane scheduling using tabu search.* Paper presented at the 14th IEEE International Conference on Tools with Artificial Intelligence, 2002 (ICTAI 2002), Washington, DC.

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

Lim, A., Rodrigues, B., & Xu, Z. (2004). Approximation Schemes for the Crane Scheduling Problem. In T. Hagerup & J. Katajainen (Eds.), *Algorithm Theory - SWAT 2004* (Vol. 3111, pp. 323-335): Springer Berlin / Heidelberg.

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

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

Liu, M., Zheng, F., & Li, J. (2014). Scheduling small number of quay cranes with non-interference constraint. *Optimization Letters*, 1-10.

Lu, Z., Han, X., Xi, L., & Erera, A. L. (2012). A Heuristic for the Quay Crane Scheduling Problem Based on Contiguous Bay Crane Operations. *Computers & Operations Research*

Martin, R. K. (2012). *Large scale linear and integer optimization: a unified approach*: Springer Science & Business Media.

Meersmans, P. J. M., & Dekker, R. (2001). *Operations research supports container handling*. Erasmus School of Economics (ESE).

Meisel, F. (2009). *Seaside operations planning in container terminals*: Springer.

Meisel, F. (2011). The quay crane scheduling problem with time windows. *Naval Research Logistics (NRL), 58*(7), 619-636.

Meisel, F., & Bierwirth, C. (2011b). A unified approach for the evaluation of quay crane scheduling models and algorithms. *Computers & Operations Research, 38*(3), 683-693.

Meisel, F., & Bierwirth, C. (2013). A framework for integrated berth allocation and crane operations planning in seaport container terminals. *Transportation Science, 47*(2), 131-147.

Meisel, F., & Wichmann, M. (2010). Container sequencing for quay cranes with internal reshuffles. *OR spectrum, 32*(3), 569-591.

Michael, P. (1995). Scheduling, theory, algorithms, and systems. *Englewood Cli s, New Jersey*

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 (NRL), 53*(1), 45-59.

Monaco, M. F., & Sammarra, M. (2011). Quay crane scheduling with time windows, one-way and spatial constraints. *International Journal of Shipping and Transport Logistics, 3*(4), 454-474.

Nam, H., & Lee, T. (2013). A scheduling problem for a novel container transport system: a case of mobile harbor operation schedule. *Flexible Services and Manufacturing Journal, 25*(4), 576-608.

Ng, W., & Mak, K. (2006). Quay crane scheduling in container terminals. *Engineering Optimization, 38*(6), 723-737.

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2013). Hybrid evolutionary computation methods for quay crane scheduling problems. *Computers & Operations Research, 40*(8), 2083-2093.

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.

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

Sherali, H. D., & Myers, D. C. (1988). Dual formulations and subgradient optimization strategies for linear programming relaxations of mixed-integer programs. *Discrete Applied Mathematics, 20*(1), 51-68.

Shor, N. Z. (1985). *Minimization methods for non-differentiable functions* (Vol. 3): Springer.

Sidney, J. B. (1979). The two-machine maximum flow time problem with series parallel precedence relations. *Operations research, 27*(4), 782-791.

Song, L., Cherrett, T., & Guan, W. (2012). Study on berth planning problem in a container seaport: Using an integrated programming approach. *Computers & Industrial Engineering, 62*(1), 119-128.

Stahlbock, R., & Voß, S. (2008). Operations research at container terminals: a literature update. *OR spectrum, 30*(1), 1-52.

Steenken, D., Voß, S., & Stahlbock, R. (2004). Container terminal operation and operations research-a classification and literature review. *OR spectrum, 26*(1), 3-49.

Tang, L., Zhao, J., & Liu, J. (2014). Modeling and solution of the joint quay crane and truck scheduling problem. *European Journal of Operational Research, 236*(3), 978-990.

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.

Theodorou, E., & Diabat, A. (2014). A joint quay crane assignment and scheduling problem: formulation, solution algorithm and computational results. *Optimization Letters*, 1-19.

Unsal, O., & Oguz, C. (2013). Constraint programming approach to quay crane scheduling problem. *Transportation Research Part E: Logistics and Transportation Review, 59*, 108-122.

Vazirani, V. V. (2001). *Approximation algorithms*: Springer Science & Business Media.

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

Wang, D., Li, X., & Wang, Q. (2011). *A two-stage composite heuristic for dual cycling quay crane scheduling problem.* Paper presented at the Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on.

Wang, S., & Hu, W. (2009). *Multi quay crane scheduling problem based on ACO in container terminals.* Paper presented at the Management and Service Science, 2009. MASS'09. International Conference on.

Wang, S., Zheng, J., Zheng, K., Guo, J., & Liu, X. (2012). Multi Resource Scheduling Problem Based on an Improved Discrete Particle Swarm Optimization. *Physics Procedia, 25*, 576-582.

Wang, Y., & Kim, K. H. (2011). A quay crane scheduling algorithm considering the workload of yard cranes in a container yard. *Journal of Intelligent Manufacturing, 22*(3), 459-470.

Zhang, H., & Kim, K. H. (2009). Maximizing the number of dual-cycle operations of quay cranes in container terminals. *Computers & Industrial Engineering, 56*(3), 979-992.

Zhu, Y., & Lim, A. (2004). *Crane Scheduling with Spatial Constraints: Mathematical Models and Solving Approaches.* Paper presented at the AMAI.

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

Paper II

On Double Cycling for Container Port Productivity Improvement

Dusan Ku and Tiru Arthanari

— Published in *Annals of Operations Research* (2014), doi: 10.1007/s10479-014-1645-z

Abstract

How quay cranes (QC) are scheduled is vital to the productivity of seaside container port operations. Double cycling concept is an operation strategy of loading the containers into ships as they are unloaded, thus improving the efficiency of a QC as well as the container port. Goodchild and Daganzo (2006) first described QC double cycling problem and solved the problem after formulating it into a two machine flow shop problem. J.-H. Song (2007) studied the formula to determine the optimal starting sequence for double cycling while reflecting on the practical issue of QC working direction. The above studies focused on a single QC double cycling and their empirical trials showed the double cycling could improve the productivity of each QC approximately by between 10 and 20 %. In (H. Zhang & Kim, 2009), a multiple QC double cycling model was first suggested by formulating a mixed integer programming model to maximise the number of double cycles between multiple QCs. In the present paper we point out a flaw with the existing multiple QC double cycling model that lets cycles that are not implementable. In addition, the paper discusses the need for imposing constraints arising from real world requirements to the formulations aiming at double cycling.

Keywords: *Container terminal, Quay crane scheduling, Double cycling*

1 Introduction

Ports are the gateways in global distribution networks and play a vital role in completing the global supply chains. Considering the growing level of service requirements that shipping lines demand to the terminal operators, the efficiency of terminal operation is becoming a matter of business survival. Although new facility or a new decision support system may increase the terminal productivity, the terminal operators are often reluctant to take this option due to the high investment cost associated with it. In the present paper, we aim to discuss a more viable option to boost the productivity of the vessel operation without incurring a significant amount of investment. In contrast to other measures, double cycling strategy in seaside operation is regarded as a low-cost method to boost the vessel productivity; it does not require new technology or infrastructure except that the vessel planning tool that terminal planners use should be able to cater for double cycling sequencing. Although double cycling will not solve the capacity problem in the long run, it is more quickly implementable than other solutions, and can be used to complement other strategies.

Double (or dual) cycling is a seaside operation strategy by which idle moves of QCs or horizontal transport vehicles, represented by yard tractors (YT) or straddle carriers (SC), for shore-to-yard transport are converted into productive ones. With single cycling, as opposed to double cycling, QCs carry out only either unloading or loading activity in each cycle and horizontal transport vehicles transport containers for only one direction in each cycle. From the QC perspective, a cycle is a crane spreader's round-trip between ship and shore. As illustrated in the Gantt chart of Figure 1.1, for example, the double cycling operation requires a total of 24 cycles whereas the single cycling operation requires 35 cycles to complete the same amount of tasks.

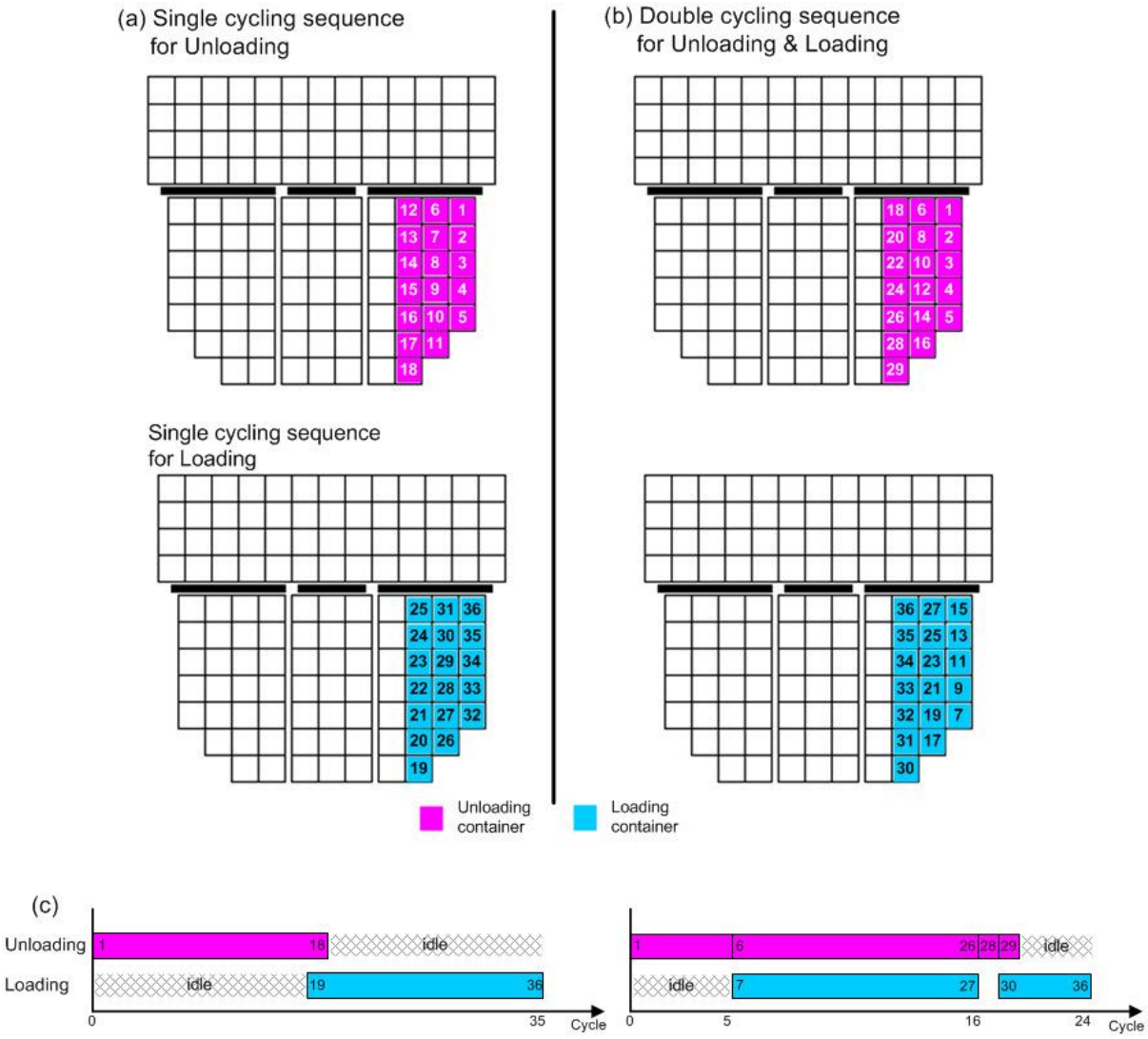


Figure 1.1 Comparison of sequence order between single and double cycling

There are two types of models with respect to the scope of double cycling. One is single QC double cycling as described in the above example and the other is multiple QC double cycling. In the single QC double cycling, the operation is carried out by a single QC within one hatch. In the multiple QC double cycling, it is carried out between two or more QCs located in separate hatches, first loading by one QC and then unloading by another QC, and the transport of the respective container(s) is handled by the same landside vehicle. The flows of equipment between the two models are illustrated in Figure 1.2.

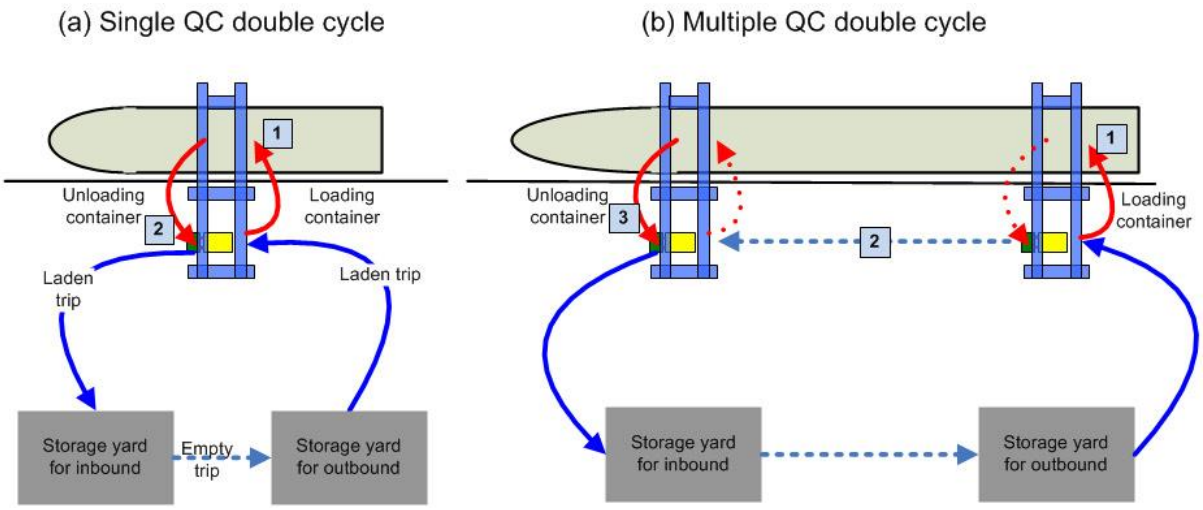


Figure 1.2 Single QC double cycling vs Multiple QC double cycling

The advantage of single QC double cycling is the reduced number of QC cycles. According to (Goodchild & Daganzo, 2006), a trial run at the Port of Tacoma, US, showed that the adjusted average time for a single cycle was 1 min 45 s, and for a double cycle it was 2 min 50 s. Thus, double cycling saved 40 s per pair of double-cycled containers and the operation time with the parameters input from the trial run was estimated to be reduced by around 10 %. The advantage of multiple QC double cycling is twofold: First, it reduces the required number of horizontal transport cycles and its direct benefit is less congestion at yard traffic as well as less consumption of fuel. Second, as in single QC double cycling it also implies the potentially reduced number of QC cycles wherever applicable. However, the empirical data of the multiple QC double cycling is yet to come out. For one reason, there is a lack of research into operational methods. For another, it requires a substantial change in work procedure, supported by the enhancement of the terminal operating system (TOS) and much training towards manpower.

Double cycling is not the method that is always applicable during the operation. There are some preconditions to be satisfied. First, the QC should be equipped with a twin-lift spreader. If it is not, the bays dedicated only to either 20- or 40-foot containers are appropriate bays for double cycling. If a bay (i.e. hatch) is mixed with 20- and 40-foot containers, 20-foot

containers are typically stowed below 40-foot containers for safety reasons.[2] Engaging double cycling under this condition may easily lead to either an accidental hit by a cell guide in the hold or an excessive QC moving time.

Double cycling strategy can be formulated within the context of the quay crane scheduling problem (QCSP). The purpose of the QCSP is to find a complete schedule for the QCs working on a vessel with its objective being typically the minimisation of the vessel turnaround time, the decisions including assigning a sequence of tasks to each QC, and the constraints including physical limitations as well as administrative limitations: physical limitations include precedence relationship among tasks, safety distance between QCs, and non-crossing constraint against another QC; administrative limitations include allowed operation time such as vessel arrival time, the deadline for departure time and break-time between shift changes of stevedore.

In this paper we first overview the literature relating to double cycling concepts in container terminal operation. The main contribution of the paper is to highlight the flaws in a recent article formulating the multiple QC double cycle problem. In addition, this paper discusses the need for imposing constraints arising from real world requirements to the formulations aiming at double cycling.

The rest of this paper is organised as follows: Section 2 reviews relevant literature on the quay crane scheduling problem (QCSP) and the double cycling quay crane scheduling problem (DCQCSP), which is a special case of the QCSP. Section 3 describes the DCQCSP from the literature. Section 4 notes the issues on the DCQCSP from the latest research trends, and discusses considerations for the double cycling to be implementable. Finally, summary of this paper and the future research direction are provided in Section 5.

[2] Inside a ship, containers stacked in the vertical direction are linked together by twist-locks (or cones) on four corners. The four corners of a 40-foot unit on top of two 20-foot units can be adequately linked by twist-locks, but the opposite can fix only two corners of each 20-foot container on either side. This may cause containers to digress while the ship is sailing.

2 Literature review

Each subsystem in container terminal has its unique operational characteristics depending on its configuration, so managing such a system optimally is very complicated. For this reason, much research tackles each subsystem separately and tries to restrict interactions between subsystems although some research tries to integrate the subsystems across terminals (Avriel, Penn, Shpirer, & Witteboon, 1998) or within a terminal. We first introduce the quay crane scheduling problem (QCSP) and the relevant literature on this problem.

2.1 Quay crane scheduling problem (QCSP)

Determining the granularity of a task is the first step in modelling the **QCSP** in general. Tasks can be defined on the basis of bay areas, complete bays, groups of containers in the same bay or individual containers (Bierwirth & Meisel, 2010). Precedence relationships can be given to the tasks where, for example, unloading precedes loading for the same ship spaces, unloading from the upper slots precedes unloading from the lower slots, and loading on the lower slots precedes loading on the upper slots. By its definition, a task is assumed to be processed without preemption by a QC while a QC can process at most one task at a time. A QC schedule solved by this problem contains a sequence of tasks assigned to each QC. Therefore, assigning a whole ship bay or a bay area, i.e. a series of consecutive bays, to a task is of little practicality because the chunk of a bay or a bay area is too big for a task and it is often in real situations in a ship bay, the workload is split between QCs. On the other hand, applying the finest possible granularity, i.e. a container, to a task makes the problem size significantly large to the point where very few algorithms are available to solve the problem within a reasonable time.

The approach of (Kim & Park, 2004) is regarded as a breakthrough work for the problem formulation with respect to assuming multiple tasks in a ship bay. They define a task as an unloading or a loading operation for a cluster, which consists of a group of homogeneous containers with, for example, the same destination port. In their mixed integer programming (MIP) model, the authors propose a B&B method and a greedy randomized adaptive search procedure (GRASP). Coined by (Feo & Resende, 1995), GRASP is a metaheuristic

algorithm commonly applied to a variety of combinatorial problems. It consists of two phases: the construction phase and the improvement phase. In the construction phase, an element of a solution is added iteratively to the solution set by using a random number and a greedy function. In the improvement phase, the solution is iteratively improved through a local search. The B&B method provides an exact solution very well for instances with two cranes and up to 10–15 tasks while it fails on instances with three cranes and 20–25 tasks. Although B&B method outperforms GRASP in terms of the solution quality, it is inadequate to solve large instances and therefore GRASP is suggested to solve the model.

In subsequent studies following the approaches taken by (Kim & Park, 2004), Moccia et al. (2006) note that the model in (Kim & Park, 2004) does not avoid crane interference completely. It only avoids collisions between QCs working on tasks in the same bay by forcing non-simultaneity constraint between two tasks in the same bay, but fails to take the travel time of QCs into account, making the model unable to consider safety margin of QCs when the tasks are in the neighbouring bays. Therefore, after revising the model of (Kim & Park, 2004) by incorporating travel times for QCs, they develop a branch-and-cut algorithm which finds improved solutions both in terms of solution quality and of computation time for the benchmark suite of problems from (Kim & Park, 2004). However, the fast growth of solution time as a function of the size of instances is still the major drawback of the branch-and-cut method although it is faster than the B&B method. Sammarra et al. (2007) present a Tabu Search heuristic, tackling the QCSP as a vehicle routing problem. Developed by (Glover, 1989), Tabu Search is basically a local search heuristic in which it moves iteratively from a candidate solution to a better solution in a subset of the neighbourhood by applying local changes. It may perform moves that deteriorate the current solution in order to avoid being trapped in a local optimum, with the hope of eventually finding a better one. In their model, a neighbourhood is defined by resequencing the tasks of a crane and by swapping tasks between cranes. Compared to the branch-and-cut method of (Moccia et al., 2006), it provides a reasonable compromise between computational burden and solution quality, yielding less computation time but slightly weaker solution quality. Bierwirth and Meisel (2009) note that even the model revised by (Sammarra et al., 2007) may yield solutions where interference between QCs is not detected. They propose a B&B algorithm for searching a reduced solution space of unidirectional schedules. Although the unidirectional schedule

approach can fail in finding the optimal solution, it might be a good strategy for solving the QCSP with container groups heuristically when satisfying all requirements of the QCSP including the issue of crane interference. Still, this algorithm clearly outperforms the heuristics proposed in all aforementioned literature in terms of solution quality and of computation time.

2.2 Double cycling quay crane scheduling problem (DCQCSP)

The DCQCSP is a special case of the QCSP in the sense that QCs are operated by double cycling method where applicable.

As for the literature on the DCQCSP, Goodchild and Daganzo are the first group of authors that formulate the double cycling problem mathematically (Goodchild, 2005; Goodchild & Daganzo, 2004, 2006). The model in their studies considers a single QC double cycling where a ship arrives in port with a set of containers on board to be unloaded and a loading plan for containers to be loaded. Given are u_c and l_c, the number of containers to be unloaded and loaded, respectively, in each stack labelled c. A key feature of double cycling is that the order in which the stacks within each ship bay are handled; this is represented as permutation functions, one for an unloading permutation and the other for a loading permutation. If we consider the makespan as the time that takes to finish the unloading and loading activities in the assumed ship bay, the proxy for the makespan is the number of cycles required to unload and load containers. Then, the problem of determining the optimal permutation to reduce the makespan can be formulated as a two-machine flow shop scheduling problem where one job corresponds to one stack and each job has two operations: an unloading operation that must be completed first, and a subsequent loading operation. The exact solution for this class of problem is solved via Johnson's rule (S. M. Johnson, 1954). First, list the tasks of each stack by unloading and loading with its number of containers to be handled, then according to u_c and l_c, select the task with the shortest cycles. If the task is for the first work centre (unloading operation), schedule the task first; if that task is for the second work centre (loading operation), schedule the last task and eliminate the selected task from further consideration. Then, repeat the above steps until all tasks have been scheduled. The output of this procedure is the permutation of handling the stacks to minimise the makespan via single QC double cycling method.

A double cycling's stevedoring direction can be divided into *from starboard to portside*, *from portside to starboard* or *zigzag type*. J.-H. Song (2007) notes that the third type may maximise the frequency of a double cycling as formulated by Goodchild and Daganzo, but it is difficult for stevedores to understand and follow such work flow. Therefore, it is a practical approach that the stevedoring direction goes in only one way, either *starboard to portside* or *portside to starboard*. Given the constraint of only one direction for double cycling, J.-H. Song (2007) also notes that the double cycling operation may frequently encounter blocking delays[3] were it not optimally considered how to decide the starting sequence for the double cycling. The point here is once a double cycling begins in a hatch, it is desirable to continue the double cycling without having blocking delays until the unloading tasks are completed, so his study focuses on finding a general rule on the optimal starting point of double cycling.

(a) Working direction →

Row No.	06	04	02	00	01	03	05
Sum of discharges(D)	D_1	~~D_2~~	~~D_3~~	~~D_4~~	~~D_5~~	~~D_6~~	~~D_7~~
Sum of loads (L)	~~L_1~~	~~L_2~~	~~L_3~~	~~L_4~~	~~L_5~~	~~L_6~~	L_7
D-L	D_2-L_1	D_3-L_2	D_4-L_3	D_5-L_4	D_6-L_5	D_7-L_6	
Formula	$D_1 + (D_2-L_1) + (D_3-L_2) + (D_4-L_3) + (D_5-L_4) + (D_6-L_5) + (D_7-L_6) = W$						
The optimal starting point of double cycle	From discharging container W						
The frequency of double cycle	Sum of discharging containers – (W-1)						

(b) Working direction →

Row No.	02	00	01	03
Discharge	6	5	4	7
Load	3	5	6	3
D-L	2	-1	1	
Formula	W = 6 + 2 + (-1+1) = 8			

(c) Working direction →

Row No.	02	00	01	03
Discharge	6	5	4	7
Load	6	5	6	3
D-L	-1	-1	1	
Formula	W = 6 + {~~(1)+(1)+1~~} = 6			

Figure 2.1 Double Cycle Formula: from portside to starboard. *Source*: (J.-H. Song, 2007)

Figure 2.1a shows the formula for calculating the optimal starting point of double cycling. In this formula, a careful consideration should be given to the case where the value of the element $(D_i - L_{i-1})$ becomes negative. If it becomes negative, it should be separately treated by being added to the next elements repeatedly until the sum of the added elements becomes nonnegative. If the sum of the added elements is still negative even after the last element has

[3] Loading can begin as soon as at least one stack has been unloaded or is empty. Loading can proceed in any unloaded or empty stack until loading operations are complete. If there is no column available, the loading operations must wait. This is called blocking delay (Goodchild & Daganzo, 2004).

been added, then those elements should be abandoned from the calculation. In Figure 2.1b, for example, $(D_3 - L_2)$ became negative by -1, so it has been added to the next element, $(D_4 - L_3)$, and the sum becomes zero, thus being admitted into the calculation. In Figure 2.1c, however, the sum of $[(D_2 - L_1) + (D_3 - L_2) + (D_4 - L_3)]$ is still negative, and therefore they are cancelled out from the calculation. The end result is the amount of single unloading cycles after which the double cycling should start given the working direction.

In contrast to a single QC double cycling model, H. Zhang and Kim (2009) extend the model to the case of multiple hatches. They formulate the problem as an MIP model and decompose the whole QC scheduling into two parts: intra-stage optimisation (sequencing all the stacks in one hatch) and inter-stage optimisation (sequencing all the hatches). For intra-stage optimisation, they apply Johnson's rule similar to that of Goodchild and Daganzo. As an improvement to this method, D. Wang et al. (2011) suggest a gap-shifting strategy in their two-stage composite heuristic, in which stacks in a hatch are scheduled by the Johnson rule integrating with the gap-shifting strategy, and the Johnson rule is reconstructed for the inter-hatch sequencing. Note that J.-H. Song (2007) has already suggested the formula in finding the optimal starting sequence with a gap-shifting concept, and the result of that formula is conceptually the same as the gap-shifting strategy of (D. Wang et al., 2011) except that the stevedoring direction in (J.-H. Song, 2007)'s model is only one direction due to practical reasoning. In (D. Wang et al., 2011)'s model shown in Figure 2.2, the Gantt charts can be partitioned into three sections.

- SU : the interval with only unloading operations
- SL : the interval with only loading operations
- DUAL : the double cycling interval
- SLOT : the gathered gaps by left shifting

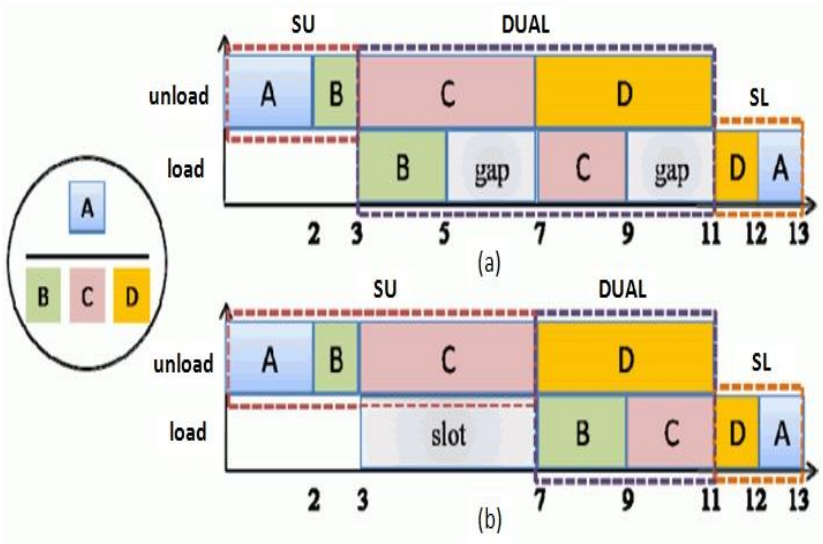


Figure 2.2 Gap-shifting strategy for intra-stage optimisation. *Source*: (D. Wang et al., 2011)

In Figure 2.2b, DUAL can be shortened by shifting all gaps before the first operation on the loading QC without changing the makespan of the hatch. Correspondingly the length of SU is increased by the length of SLOT. Therefore, there is a possibility that a better makespan may be yielded when reconnecting hatch sequences with the prolonged SUs.

For inter-stage optimisation, H. Zhang and Kim (2009) suggest heuristics based on gap-based neighbourhood local search. The experimental results show that the proposed method outperforms the real schedules constructed by human planners. However, their model fails to recognise the cranes' interference constraints and the non-crossing constraints between cranes, thus making it practically contradictory. He et al. (2011) include the cranes' interference constraints in their study on the integrated large-capacity quay crane scheduling but still have the shortcomings of ignoring the non-crossing constraints between the cranes.

Gadeyne and Verhamme (2011) propose two models in the QCSP, one for the QCSP in general and the other for the DCQCSP. In their DCQCSP model, they incorporate some practical constraints such as sequence-dependent setup times and interference constraints. Also, the discretised time steps (e.g. 30 s in one time unit) are added in their model to visualise and verify the correctness of each constraint, at every time unit. The decision on one time unit is made with approximation from the literature (Goodchild & Daganzo, 2006) and their own measurements at a reference port (Port of Zeebrugge), and is summarised in Table 2.1.

Each QC can have only one action status among the five statuses. The application of the time segment concept for measuring the makespan is a more practical approach than just the number of cycles as assumed in the previous literature. Consequently, however, this model is quite detailed, and has too many variables. Therefore, this approach is only applicable to small problems.

Table 2.1 Time units per QC action. *Source*: (Gadeyne & Verhamme, 2011)

Action	Average	Approximation	Time units
Spreader setup	28 s	30 s	1 (per action)
Single cycle	1 min 45 s	1 min 30 s	3 (per action)
Empty movement	–	30 s	1 (per action)
Double cycle	2 min 50 s	3 min	6 (per action)
QC displacement	–	(Fixed part) 4 min 30 s	9 (per action)
	1 min 00 s	(Variable part) 30 s	1 (per 40ft bay)

Except for the studies mentioned above, it is not easy to find notable literature as to double cycling. As Goodchild opened the floodgate for modelling double cycling and Zhang and Kim extended the scope of the model to multiple hatches, more thorough research is expected to come out in future. As it is expected that much of the future research bases their model on the scenario of multiple hatches following Zhang and Kim's model, it is important to note what the critical shortcomings are in this model, thus motivating the future researchers to tackle these issues first when formulating their models.

3 Multiple QC double cycling

3.1 Problem description

The ultimate goal of a vessel operation is to minimise the makespan of all QC activities on a ship. When attempting to apply multiple QC double cycling method, the problem consists of at least two independent QCs working on different hatches, and where at some moments one QC loads a container (or twin containers) onto the vessel, another QC unloads a container (or twin containers) from the vessel, and both QC activities are interacted with the same landside vehicle. Regarding the makespan, the following assumptions are made:

1. the operation time for a QC to complete all the discharging and the loading operations in a ship-bay can be expressed as the number of cycles;
2. the operation of QCs is the bottleneck during the vessel operation, so the throughput rate of QCs determines the throughput rate of the integrated handling system consisting QCs, landside transporters and yard cranes.

Regarding landside transport, we assume a closed queueing system in which landside vehicles are the customers shuttling between two types of server stations, i.e. the QC type and the storage yard type. In reality, the QC productivity is significantly affected by the productivity of the landside processing. However, the appropriate number of landside vehicles assigned to a QC or a pool of QCs varies depending on the environment of each terminal. For the simplification of this model, the following assumptions are made:

1. there are enough number of landside vehicles so that the queues at the QC servers are never idle;
2. there are infinitely many servers at the storage yard so that landside vehicles do not have to wait for services at yard.

Regarding hatch covers by which most modern container ships separate the deck from the hold of a ship bay, the following assumptions are made within the range of a hatch:

1. all the unloading tasks under the deck (in the hold) cannot begin until all the unloading tasks on the deck are completed;
2. all the loading activities on the deck cannot begin until all the loading tasks in the hold are completed. These assumptions lead to an additional implicit assumption that if single QC double cycling is applicable to reduce the makespan it will be conducted only within the tasks in the hold.

We revisit (H. Zhang & Kim, 2009) to describe a mathematical formulation of this problem. We are given a set of hatches, H, a set of stacks for which unloading operation is planned, S^u, and a set of stacks for which loading operation is planned, S^l. A set $\overline{S}_h^u$ is defined as the set of stacks for which unloading operation is planned in the hold of hatch h, $\overline{S}_h^l$ as the set of stacks for which loading operation is planned in the hold of hatch h, $\underline{S}_h^u$ as the set of stacks

for which unloading operation is planned on the deck of hatch h, and $\underline{S}_h^l$ as the set of stacks for which loading operation is planned on the deck of hatch h. Let n^u denote the number of containers to unload from stack i, and n^l the number of containers to load into stack i. The notations for the completion times are as follows:

- C_i^u: completion time of unloading all containers from stack i
- C_i^l: completion time of loading all containers into stack i
- $\overline{C}_h^u$: completion time of unloading all containers from the hold of hatch h
- $\overline{C}_h^l$: completion time of loading all containers into the hold of hatch h
- $\underline{C}_h^u$: completion time of unloading all containers from the deck of hatch h
- $\underline{C}_h^l$: completion time of loading all containers into the deck of hatch h

The objective is to minimise ω, which represents the number of cycles necessary to complete all the QC tasks. It can be constrained by the following inequalities:

$$\omega \geq \overline{C}_h^u \qquad \forall h \in H \tag{3.1}$$

$$\omega \geq \overline{C}_h^l \qquad \forall h \in H \tag{3.2}$$

$$\omega \geq \underline{C}_h^u \qquad \forall h \in H \tag{3.3}$$

$$\omega \geq \underline{C}_h^l \qquad \forall h \in H \tag{3.4}$$

Constraints regarding hatch covers and the precedence relationship between the loading/unloading activities in the same stack are represented as follows: Constraint (3.5) ensures that the unloading activities in a hold cannot begin until all the unloading activities on the deck of the same hatch are completed; Constraint (3.6) ensures that the loading activities in a stack cannot begin until all the unloading activities in the same stack are completed, while

constraint (3.7) ensures that in each hatch the loading activities on deck cannot begin until all the loading activities in hold of the same hatch are completed.

$$C_i^u \geq \underline{C}_h^u + n_i^u \qquad \forall i \in \overline{S}_h^u,\ \forall h \in H \tag{3.5}$$

$$C_i^l \geq C_i^u + n_i^l \qquad \forall i \in S^u \cap S^l \tag{3.6}$$

$$C_i^l \geq \overline{C}_h^l + n_i^l \qquad \forall i \in \underline{S}_h^l,\ \forall h \in H \tag{3.7}$$

X_{ij} and Y_{ij} denote binary decision variables. X_{ij} is equal to 1 iff unloading for stack j is performed immediately after unloading for stack i, and Y_{ij} is equal to 1 iff loading for stack j is performed immediately after loading for stack i. This definition can be represented as the following constraints (3.8) and (3.9) where M represents a sufficiently large number:

$$C_j^u - C_i^u + M(1 - X_{ij}) \geq n_j^u \qquad \forall i, j \in S^u \tag{3.8}$$

$$C_j^l - C_i^l + M(1 - Y_{ij}) \geq n_j^l \qquad \forall i, j \in S^l \tag{3.9}$$

Apparently, there is no mention of crane interference constraints in this mathematical formulation, let alone the indices for cranes. This has led us to test the validity of this model and our findings will be noted in the next section.

3.2 Two-staged hybrid heuristic

In the hybrid heuristic approach taken by (H. Zhang & Kim, 2009) for the DCQCSP in multiple hatches, the algorithm is decomposed into two parts: inter-stage sequencing and intra-stage sequencing. If we assume that the intra-stage sequencing has been optimised by the combined contributions from (Goodchild, 2005; Goodchild & Daganzo, 2004, 2006; J.-H.

Song, 2007; D. Wang et al., 2011), a simple example of the inter-stage sequencing problem can be illustrated as in Figure 3.1.

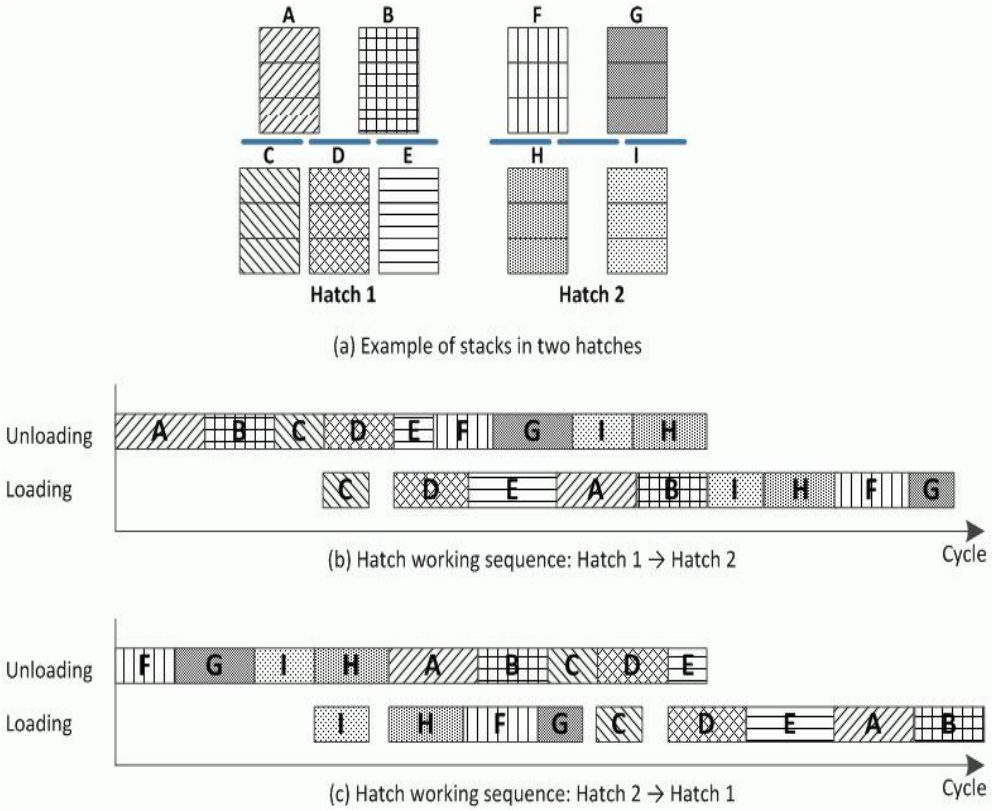


Figure 3.1 Simple comparison between different hatch sequencing orders

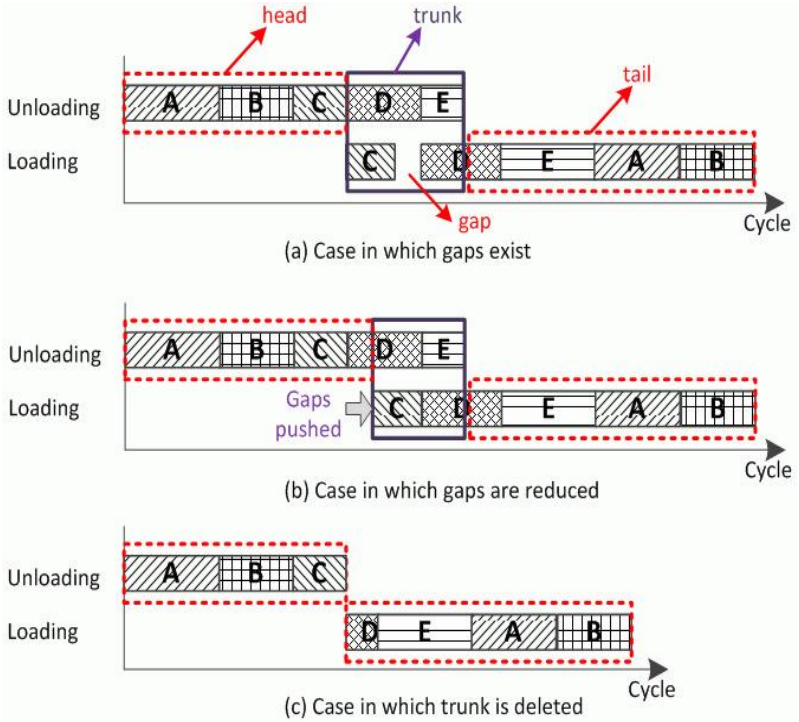


Figure 3.2 Abstraction of tasks in one hatch

For a single QC double cycling to be started in a ship hold, at least one stack, usually the first stack to load, should be free of any unloading task. This is evident the first loading stack C in Figure 3.1b begins to be filled in, at which point double cycling also begins, after the unloading task in that stack is completed, and the same applies to the first loading stack I in Figure 3.1c. This illustration shows that the different hatch working sequence may yield different number of cycles as its makespan, so finding the best hatch working sequence of each QC for minimising the makespan of the vessel operation is the crux of this problem.

Zhang and Kim's approach is to apply a local search heuristics for the inter-stage optimisation. Initially, an arbitrary hatch sequence is given (or chosen), and swapping the hatches has the possibility of yielding the different number of total cycles: see Figure 3.1b, c. Then, using a local search heuristic called Gap-based neighbourhood local search, they propose to find the best hatch sequence among the neighbourhood that minimises the total makespan. In order to simplify the Gantt chart for processing each hatch, they classify the tasks in one hatch into three parts: head, tail and trunk as in Figure 3.2a.

Head part consists of not only all the unloading activities on the deck, but also the first unloading activity in the hold. Tail part consists of not only all the loading activities on the deck, but also some part of loading activity in the hold. Trunk part consists of both loading and unloading activities, which represent dual cycles. It is the ignorable part in the whole schedule during the hatch sequencing, and can be reduced by pushing the gaps as in Figure 3.2b. Since the trunk part involves both loading and unloading activities, it cannot affect the makespan of the original problem when swapping hatch sequences during the inter-stage optimisation. Therefore, it can be reformulated as in Figure 3.2c by deleting the trunk and linking the head and the tail up together. Then, the reformulation falls into a category of a two-machine flow shop scheduling problem, which again enables the use of Johnson's rule to obtain the optimal hatch working sequence.

4 Issues on the DCQCSP

4.1 Crane interference constraints

Since a QC is a rail-mounted vehicle, crane interference constraints are of such an importance when it comes to the QCSP that involves multiple QCs. This is also true for the DCQCSP in multiple hatches, which is a special case of the QCSP. Two types of crane interference constraints are typically considered:

- Non-crossing constraint: QCs cannot cross each other;
- Safety constraint: There is a safety distance between QCs at any time.

In the QCSP, interference constraints were first included in (Kim & Park, 2004) as a mathematical form, and thereafter rigorously studied by (Bierwirth & Meisel, 2009; Moccia et al., 2006; Sammarra et al., 2007), and so on. However, the existing models for multiple QC double cycling do not take these constraints into account. To demonstrate the incorrectness in the model of (H. Zhang & Kim, 2009), we consider a sample problem where there are four hatches and two QCs.

Assume that the optimised hatch sequence among the hatches arranged laterally as A, B, C and D is resulted in the sequence of A→D→C→B through the inter-stage optimisation. One example of the Gantt chart of this hatch sequence would be such as in Figure 4.1a. We can expect two QCs work simultaneously with double cycling between the hatches of 'A and D,' 'D and C,' and 'C and B.' In this sample schedule, the first hatch to work is A, and therefore we locate one QC (QC1) in this hatch. By the time QC1 has to work on the single loading tasks in the hatch A, another QC (QC2) in the hatch D participates in the single unloading tasks in the hatch D, the combination of which makes QC1 and QC2 perform the double cycling operation between the hatch A (loading) and the hatch D (unloading). Once the tasks in the hatch A are completed, QC1 that was working in the hatch A moves to the hatch C, and starts to perform the single unloading tasks when it is appropriate to engage in the double cycling operation with QC1 in the hatch D.

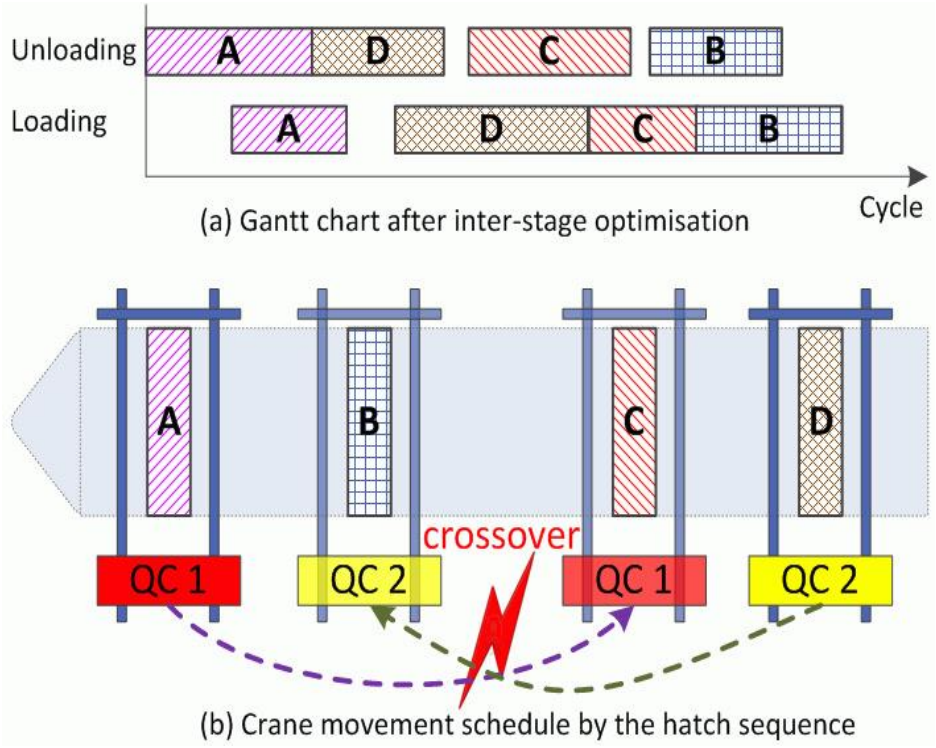


Figure 4.1 Counter example of crane interference

In this manner, the entire movements of QCs along the hatches can be illustrated in Figure 4.1b. Here we can detect a crossover situation between the two QCs when QC2 attempts to work in the hatch B while QC1 is assigned to the hatch C. This means the solution found by this model is infeasible by the non-crossing constraint. Furthermore, it could be that hatches in

the neighbouring hatch sequences are located within the safety margin, in which case the solution derived by this model is also infeasible by violating the safety constraint.

4.2 Maximising double cycles by connecting two hatches

The reasoning of Zhang and Kim's approach is that the more double cycles are carried out between groups of two hatches, the less number of operation cycles as the makespan will be yielded. So, in their model, maximising the number of double cycles is equivalently treated as minimising the number of operation cycles. If it were limited to a single QC double cycling, maximising the double cycles would likely lead to the minimisation of overall makespan. However, it is worthwhile to note that maximising the double cycles by connecting two appropriate hatches continuously does not necessarily lead to the minimisation of overall makespan due to the following reasons:

1. At times one QC, which is connected to another QC for double cycling, travels to a hatch and waits there until the scheduled double cycling sequence arrives. Therefore, travel time should be considered and also unnecessary idle time easily arises until a matching pair of double cycle sequences between the two QCs arrive;
2. To mitigate the issue with the above situation, it can sometimes be more efficient that two or more unloading QCs are connected to one loading QC, or vice versa. However, Zhang and Kim's model has no room for this consideration;
3. A delay of a QC resulting from avoiding the interference with another QC should be considered.

4.3 Granularity of a task

The granularity of a task needs to be elaborated for single QC double cycling. In the existing literature on the DCQCSP, a task is typically defined on the basis of a stack. As noted earlier, a task is a unit of activities to be processed without preemption by a QC. We can easily encounter a contradiction if we assume a stack as a task in the case of double cycling operation. In double cycling, two stacks are alternatingly involved in the operation, one stack for unloading and the other stack for loading. However, this does not mean we can isolate these two stacks as one task because the combination of two stacks for double cycling is

continuously changing one after the other and therefore there is no way to isolate two specific stacks as a task. As Figure 4.2 illustrates, stacks involved with double cycling are continuously changing. Between the fourth and the fifth cycle stacks A and B are involved, for the sixth cycle stacks A and C are involved, and between the seventh and the eighth cycle stacks B and C are involved in the double cycling operation. It is evident that no single stack or two can be isolated to consist of a task in case of double cycling.

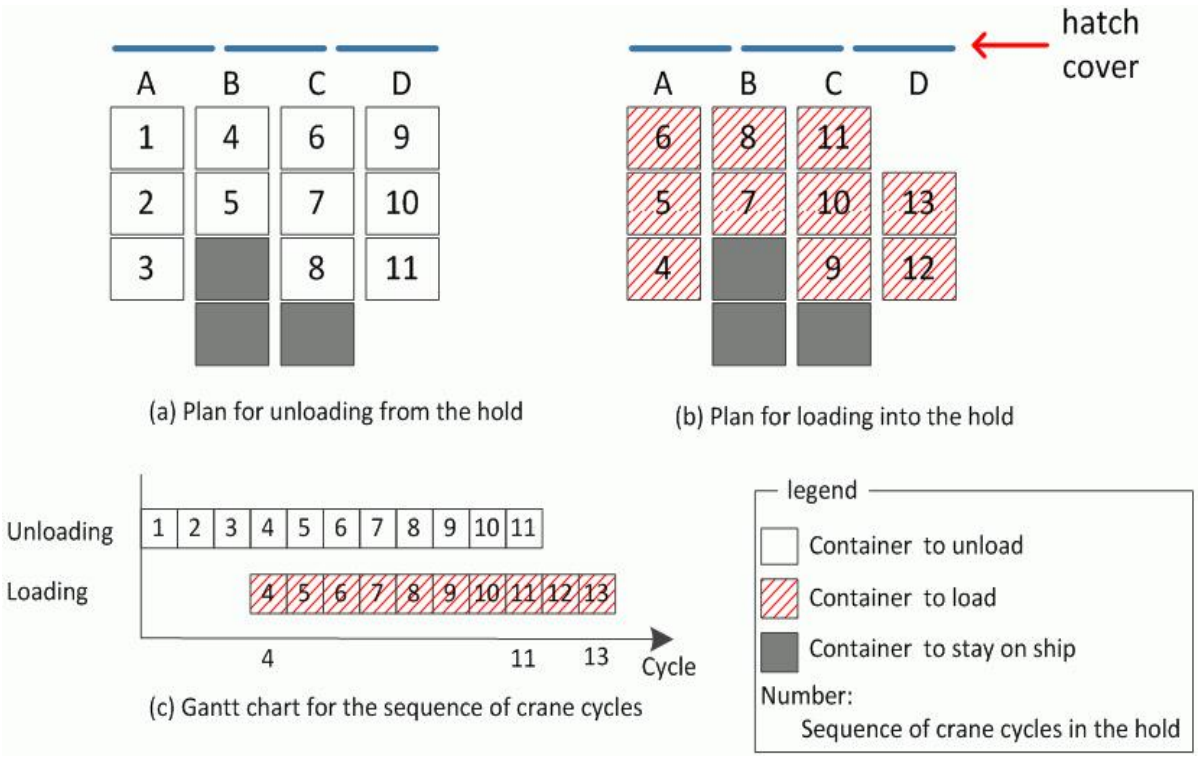


Figure 4.2 Sequence for unloading, dual and loading cycles

Considering the issues with a stack as a task in double cycling model, our question is what could be an alternative unit of a task for future research direction. Obviously, this remains to be explored through further research, so there is neither more efficient nor less efficient answer to this question yet. Our suggestion to consider is as follows:

For single QC double cycling, all activities above the deck have nothing to do with double cycling. So, we can leave the modelling of tasks above the deck to the traditional QCSP model. For example, Kim and Park (2004) suggest that tasks be separated by the hatch covers for unloading containers, and by container groups for loading containers. Following this suggestion, there is only one task for containers to unload from above the deck, and are as

many tasks as there are distinguishable groups (container size and type, destination and so on) for containers to load into above the deck. However, retaining the stack concept as a task for activities below the deck seems inevitable because double cycling only becomes available as soon as any stack in the hold is free of containers to unload.

Considering the continuously changing combination of stacks for double cycling, we propose to ignore tasks on loading stacks where double cycling takes place. In this proposal, *dual* tasks are additionally categorised on top of the traditional *load* and *unload* tasks. The *unload* tasks are reclassified as *dual* tasks where unloading stacks are simultaneously worked with other loading stacks. Therefore, the tasks in the sample shown in Figure 4.2 can be reduced to five tasks—A (unload), B (dual), C (dual), D (dual) and D (load)—instead of having eight tasks (four *unload* tasks and four *load* tasks). The details of reclassified tasks are as follows:

- A(unload) : unload 1–3
- B(dual) : unload 4–5 and load 4–5
- C(dual) : unload 6–8 and load 6–8
- D(dual) : unload 9–11 and load 9–11
- D(load) : load 12–13

5 Summary and future research

The analysis made in this paper discloses some serious weaknesses of the existing model for multiple QC double cycling. First and foremost, they do not take into account the assignment of tasks to each QC properly, and thereby the yielded solution can cause the crane interference, thus making it infeasible. Later work by (He et al., 2011) attempts to include the crane interference constraints in the formulation but it only considers the situation where two QCs operate on the same stack simultaneously, which is far from satisfactory for the consideration of the safety distance between QCs, let alone the non-crossing constraint.

Secondly, the objective in the existing model on the multiple QC double cycling is to maximise the frequency of double cycles by connecting two appropriate hatches continuously.

However, we have rebutted that doing so does not necessarily result in the minimisation of overall makespan.

In addition, there are some real world requirements like additional physical constraints and operational convenience to the formulations aiming at double cycling. To take care of the missing constraints may need a totally different approach than just used by the existing models. Since the existing models do not seem to be scalable enough to include those missing constraints straightaway, our current research effort is on formulating a new model that will include the missing constraints on top of the constraints addressed by the existing models. Some of the approaches we have investigated or plan to investigate are as follows:

- the parameters for QCs need to be included to support the assignment of tasks to QCs;
- the objective function needs to consider not only the efficiency of the seaside, i.e. the number of the required QC cycles, but also the efficiency of the landside, i.e. the number of landside transport cycles;
- a new perspective on the granularity of a task needs to be examined as we have already elaborated in the previous section.

Exploiting double cycling strategy for quayside operation at seaport is a useful option to boost the terminal productivity especially when the infrastructure expansion is limited. A properly implemented double cycling process can improve the QC operation, which is generally regarded as the bottleneck to the entire throughput at seaport, by reducing the required number of QC cycles as well as that of landside transport cycles. Also, this can economically benefit the terminal operators by saving fuel consumption with less travelled yard-to-shore transports.

In contrast to the potential benefits of the double cycling, single cycling method is commonly employed in most container terminals for various reasons. One critical reason, among others, is that the industry practitioners have been used to the current practice for a long time to the point where many rules and systems are already mingled with the current single cycling method. For one, there can be issues with labour environment. A gang is a group of stevedores that work on a QC during the given shift hours. In some cases, the labour agreement even specifies that once a gang is assigned to operate on a ship, the gang will not

work on another ship unless paid by an extra amount of money. For another, the tools the terminal operators use often require modifications such as enabling to plan double cycling sequences easily if the current tools they use do not support it yet. For the other, even though most academic research such as this paper assumes the yard condition is so favourable that any internal carrier as well as any container to be loaded can be available whenever needed by the double cycled QC, the reality is often not the case. There can be more and more reasons that hamper the implementation of double cycling. Therefore, the above items, but not limited to, are the agenda that can be addressed in future research direction.

Acknowledgements

The authors would like to thank the referees for their valuable comments and suggestions.

References

Avriel, M., Penn, M., Shpirer, N., & Witteboon, S. (1998). Stowage planning for container ships to reduce the number of shifts. *Annals of Operations Research, 76*, 55-71.

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.

Feo, T. A., & Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization, 6*(2), 109-133.

Gadeyne, B., & Verhamme, P. (2011). *Optimizing Maritime Container Terminal Operations.* (Master of Business Economics Master Thesis), Ghent University.

Glover, F. (1989). Tabu search-part I. *ORSA Journal on computing, 1*(3), 190-206.

Goodchild, A. V. (2005). *Crane double cycling in container ports: algorithms, evaluation, and planning.* (PhD thesis), University of California at Berkeley, CA.

Goodchild, A. V., & Daganzo, C. F. (2004). *Reducing ship turn-around time using double-cycling*. Institue for Transportation Studies Research Report (UCB).Retrieved from http://escholarship.org/uc/item/86r4p6sc

Goodchild, A. V., & Daganzo, C. F. (2006). Double-cycling strategies for container ships and their effect on ship loading and unloading operations. *Transportation Science, 40*(4), 473-483.

He, X., Wang, S., & Zheng, J. (2011). *A hybrid heuristic algorithm for integrated large-capacity quay crane scheduling problem.* Paper presented at the Computer Research and Development (ICCRD), 2011 3rd International Conference on.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval research logistics quarterly, 1*(1), 61-68.

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

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 (NRL), 53*(1), 45-59.

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

Song, J.-H. (2007). A study for optimization of double cycling in container ports. *Port Technology International, Edition 36*, 50-52. Retrieved from http://www.porttechnology.org/technical_papers/a_study_for_optimisation_of_double_cycling_in_container_ports/

Wang, D., Li, X., & Wang, Q. (2011). *A two-stage composite heuristic for dual cycling quay crane scheduling problem.* Paper presented at the Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on.

Zhang, H., & Kim, K. H. (2009). Maximizing the number of dual-cycle operations of quay cranes in container terminals. *Computers & Industrial Engineering, 56*(3), 979-992.

Paper III

A Mathematical Formulation for the Multi-QC Double Cycling Problem in a Marine Container Terminal

Dusan Ku and Tiru Arthanari

— Unpublished

Abstract

Since its inception about a decade ago, research on the crane double cycling problem in a marine container terminal has focused on the single crane model. The idea to extend the scope of the quay crane double cycling problem (QCDCP) to multiple quay cranes (QCs) has been discussed by several researchers, but the realisation into a plausible model has often failed, either because of the complexity in modelling the seamless integration between yard-side flow and quay-side flow or because of the violation against critical constraints such as crane clearance conditions. Under the discretised planning horizon, the paper develops an exact model respecting the crane's interference constraints. As the objective of the integer programming model, a function of weighted sum is considered with weights assigned to the makespan and the number of multi-QC double cycles. A polynomial time solution with two-phase heuristics (construction and improvement phase) is proposed as an alternative to the exact solution when the instance size is intractable.

Keywords: *Shipping/transportation, Container terminal, Quay crane scheduling, multi-QC double cycling, Integer programming*

1 Introduction

The competitiveness of a marine container terminal depends strongly on the time during which the terminal processes the amount of workloads of ships berthed at the terminal. The faster the terminal can handle its required tasks, the more attractive the terminal becomes to the shipping lines. Often it can be achieved by expediting the process with more resources into the operation. When the ports are congested with ships to be processed or the limited port capacity is a bottleneck in adding such resources, however, just adding resources may not be a viable option. It is especially the case because the expansion of the port capacity usually faces a significant amount of capital investment or resistance from interest groups. As an affordable low-cost method to boost vessel productivity, practitioners have long thought of employing the crane double cycling strategy; it does not require new technology or infrastructure. Although adopting the double cycling process will not solve the capacity problem in the long term, it can be more quickly implemented than other solutions, and can be used to complement other strategies (Goodchild & Daganzo, 2006).

Crane double cycling is a quay crane (QC) operation strategy that minimises an empty trip of the equipment involved in the vessel operation. From the perspective of QC, a QC unloads a container onto a horizontal transport vehicle, such as yard tractor (YT), automated guided vehicle (AGV) or straddle carrier (SC), and instead of returning the crane's empty spreader back to the ship, the QC picks up another container available from the shore to load into the ship. From the perspective of the transport vehicle, a vehicle delivers a loading container to a QC and instead of directly returning to the storage yard for the next loading container, the vehicle accesses another QC which can deliver an unloading container to the vehicle itself. When any of the double cycling operations described above occurs, it can be expected that according to the sequence scheduled for each QC, in the former case the QC is operating on alternating loading/unloading cycles and in the latter case the first QC is progressing on loading sequences while the second QC is progressing on unloading sequences. The flow of container and equipment for each double cycling mode is illustrated in Figure 1.1.

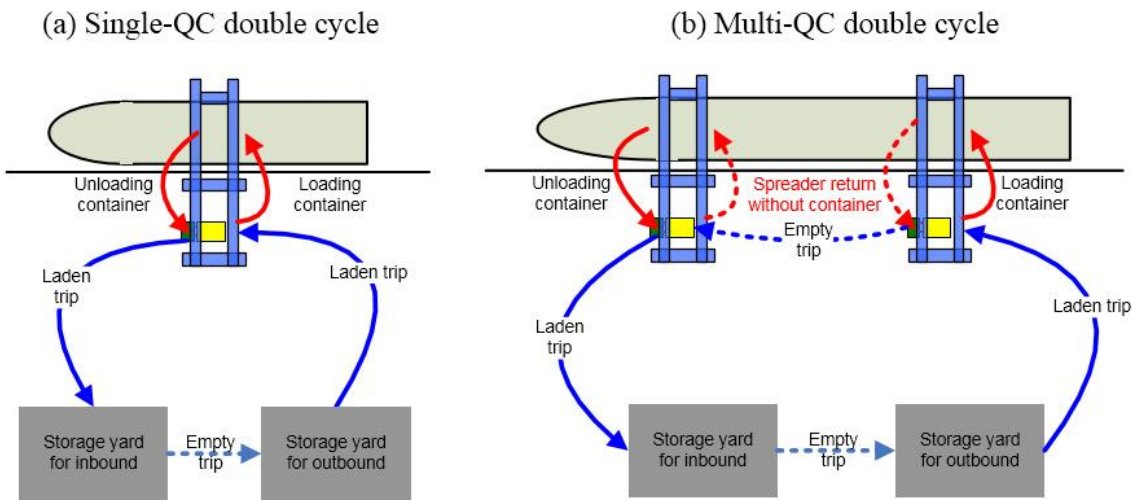


Figure 1.1 Single QC double cycle vs Multi-QC double cycle

The quay crane double cycling problem (QCDCP) is a special case of the well-known quay crane scheduling problem (QCSP), the main objective of which is to find a complete QC working schedule with its objective typically being the minimisation of the makespan for the vessel operation, the minimisation of the total completion time of QCs, or the combination of both. The QCSP and other scheduling problems in the seaside planning and operation have been reviewed intensively in (Bierwirth & Meisel, 2010, 2014). The QCDCP was first studied by Goodchild and Daganzo in a series of works in (Goodchild, 2005; Goodchild & Daganzo, 2004, 2006), with the aim of minimising the number of QC cycles. Their model is formulated as a two-machine flow shop scheduling problem, which is proven to be solvable in a polynomial time (S. M. Johnson, 1954), to determine a permutation of stacks for double cycling operation flow. Recently, a more complex scenario considering hatch covers, which divide a ship bay into deck (above hatch covers) and hold (below hatch covers), is studied by (C.-Y. Lee et al., 2014). They term the double cycling model considering hatch covers the general QCDCP and prove that it can be formulated as a flow shop scheduling problem with series-parallel precedence constraints, which is solvable in polynomial time (Sidney, 1979). By taking into account the features of double cycling, Meisel and Wichmann (2010) study the container sequencing problem for reshuffles in a ship bay. An integer programming (IP) model and a heuristic solution method, namely a greedy randomized adaptive search procedure (GRASP), are provided for planning crane operations under internal reshuffles. Their computational results give an insight into the performance enhancement of employing

internal reshuffles compared to a sole application of crane double cycling. Aside from the aforementioned single QC scenario for double cycling, more recent research trends attempt to consider the multi-QC double cycling problem (MQCDCP) as discussed in (Ku & Arthanari, 2014). However, no breakthrough for modelling the multi-QC double cycling has been made so far. This is the aim of the present study.

This paper is structured in the following order: Section 2 describes the multi-QC double cycling problem with the complete IP formulation in Appendix. Then, a two-phased heuristic approach is proposed in Section 3 in order to make the solution approach tractable. The results of computational experiments are given in Section 4. Section 5 concludes the paper with suggestions for the future research direction.

2 Multi-QC double cycling problem (MQCDCP)

The objective of the multi-QC double cycling problem is to find an appropriate number of sequence pairs between a loading operation from a QC and an unloading operation from another QC. Transport vehicles are then scheduled to travel between the sequence pairs of the two QCs in order to minimise the empty trips of the transport vehicles. Although maximising the number of double cycles is an important performance measure as studied in (H. Zhang & Kim, 2009), the minimisation of the makespan, the ultimate goal of vessel operation, can never be neglected. Hence, we propose in our study a function of minimising the makespan and maximising the number of double cycles.

We base the MQCDCP on the following QCSP model. We are given a set of tasks distributed along ship hatches and a set of QCs along the rail track on the quay. Each task consists of multiple container handlings, and the processing time of each task depends on the number of containers to be handled. Precedence relations may exist between tasks in the same ship hatch. QCs are identical, which means their productivity rates are equal. As is often the case with most studies on the QCSP, we are interested only in non-preemptive schedules, in which each task must be processed without interruption. The classification scheme introduced by (Graham, Lawler, Lenstra, & Kan, 1979) offers a convenient way to refer to scheduling problems. The model that we consider with the objective function described above can be

denoted by $Pm|prec|\alpha C_{max} - \beta W$ where W is the number of multi-QC double cycles, and α and β are positive numbers indicating the weights for the makespan (C_{max}) and the number of multi-QC double cycles. The problem $Pm|prec|C_{max}$ is $\mathcal{NP}$-hard in the strong sense unless the graph of precedence constraints is an *intree* or an *outtree* (Michael, 1995). Thus, converting the performance measure into the weighted sum containing the weighted C_{max} part makes our model at least as difficult as $Pm|prec|C_{max}$. Bierwirth and Meisel (2010) also offer another classification scheme dedicated to scheduling problems of container terminals. By their proposal, our model can be denoted by $Group, prec|move|cross, save|\alpha \cdot max(compl) - \beta \cdot max(W)$. That is, the travel time for QC movement is respected by *move* attribute. Also, QC's interference constraints (non-crossing and safety margins between QCs) are respected by *cross* and *save* attributes.

We consider a vessel operation during a planning horizon, which is the maximum time in which all container handlings have to be completed. The entire planning horizon is discretised and the length of each interval is the amount of time needed for a QC to perform a container handling, either a loading or an unloading. Thus, the maximum planning horizon will not be more than the workload (the number of container handlings) plus the minimum travel time of a QC (approximately the number of hatches) in the extreme scenario of a single QC operation. It is trivial to state that the required duration of the planning horizon is likely to be decreased with more QCs in operation. For each pair of multi-QC double cycles, we assume that a transport vehicle's trip between a loading QC and an unloading QC is to be completed between the release points in time from each QC. For loading, the vehicle under the loading QC is released as soon as the QC picks up a container from the vehicle. For unloading, the vehicle will be laden with a container when the QC unloading the container from ship to shore puts the container on the vehicle. Therefore, we can assume that in the planning horizon, the loading occurs during time period *t* and the unloading occurs during time period $t + 1$. One crane cycle takes about 1.5 to 2.5 minutes in a practical terminal operation; hence it is a reasonable assumption for a transport vehicle to move between two QCs on the same ship during one time period. Figure 2.1 gives an illustration for the flow of the involved equipment.

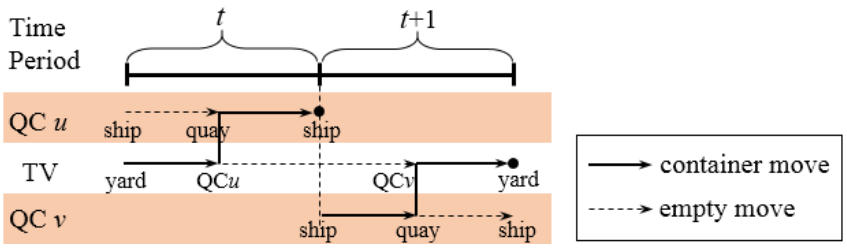


Figure 2.1 Move time of transport vehicle (TV) between the release points of QC *u* and QC *v*

The assumptions based on the features considered in this paper are summarised as follows:

- A sufficiently large number of yard cranes and transport vehicles support the QCs in order to prevent any delay in the vessel operation;
- The number of QCs allocated to the vessel is fixed and the QCs do not move away from the vessel during the entire planning horizon of vessel operation;
- *Initial position*: Initial QC positions are not given as parameters but left as decision variables. However, this assumption can be relaxed in our heuristic approach to cater for benchmark instances where initial QC positions are given as parameters;
- *Maximum travel distance per time period*: A QC can travel laterally along up to a certain number of hatches during one time period, thus the maximum distance that can be travelled within the one time period is given as a parameter. The time gap between travelling along one hatch and the maximum distance (if more than one hatch) is negligible. That is, if a QC must move anyway, moving to an adjacent hatch and moving to a hatch in the maximum distance away from the current position equally take one time period.

Also, the physical constraints for the feasibility of the model are considered as follows:

- *Safety distance between adjacent QCs*: The safety margin is given as a parameter in terms of the number of hatches for any two QCs to be away from each other at all times. By considering the safety margin, we infer that tasks within the safety margin cannot be processed simultaneously;
- *Non-crossing constraints among QCs*: QCs are positioned along the rail track on the quay, therefore the physical constraint by which QCs cannot cross each other's path is considered in our model;

- *Precedence relations of tasks*: Each hatch in a ship consists of multiple tasks in a chainlike precedence relation, and each task must consist of jobs with only one operation type among the three possible types: unloading, double cycling and loading. Let $\prec$ denote a partial order among tasks. That is, task $i \prec$ task j when task i precedes task j. In general, the following chainlike precedence relation is considered in each hatch: *unloading* $\prec$ *double cycling* $\prec$ *loading*.

2.1 Notations

In this section, we introduce the notations first before providing a detailed description of the model constraints. The complete mathematical formulation is provided in the Appendix.

Indices and sets:

i, j : Task index;

t : Time period index, which represents the time interval from the point in time $t-1$ to time t, i.e., $(t-1, t]$.

Q : The set of QCs, $Q = \{1, \cdots, q\}$;

H : The set of hatches; $H = \{1, \cdots, h\}$;

Λ : The set of unloading tasks;

Υ : The set of loading tasks;

Δ : The set of double cycling tasks;

Ω : The set of all tasks, i.e., $\Omega = \Lambda \cup \Upsilon \cup \Delta = \{1, \cdots, n\}$; each task consists of only one operation type among the three possible types: unloading, loading or double cycling;

Φ : The set of ordered pairs of tasks, representing precedence relationships between pairs of tasks; $\Phi = \{(i, j) \in \Omega^2 | \ task\ i\ must\ precede\ task\ j\}$;

Parameters:

N_i : The number of container handlings in task $i \in \Omega$;

b_i : The hatch index where task *i* is located;

T : The total number of time segments in the planning horizon, the maximum of which can be set to $\sum_{i\in\Omega} N_i + |H|$;

δ : The safety distance;

ζ : The maximum distance in terms of the number of hatches which a QC can move along during one time period;

α : The weight for the makespan;

β : The weight for the double cycling.

Decision variables:

y_i^k: 1, if task *i* is assigned to QC k; 0, otherwise;

$s_i(t)$: 1, if task *i* starts at time period *t*; 0, otherwise;

$\eta_b^k(t)$: 1, if QC k is positioned at hatch *b* at time period t; 0, otherwise;

$x_i(t)$: 1, if a container handling in task *i* is performed at time period t; 0, otherwise;

$o_i^k(t)$: 1, if a container handling in task *i* is performed by QC *k* at time period t; 0, otherwise;

$\omega^{uv}(t)$: 1, if QC *u* performs an unload handling at time period *t*, QC *v* performs a load handling at time period $t+1$, and both operations are linked together as a double cycling by the same transport vehicle $(u, v \in Q, u \neq v)$; 0, otherwise;

$\gamma(t)$: work completion flag: 1 if all tasks have been completed at time period *t*; 0, otherwise.

2.2 Fixed number of QC allocations

The model considers that the number of QCs assigned to vessel operation is fixed during the entire planning horizon such that QCs do not move away from the vessel. $\eta_b^k(t)$ denotes a binary variable that represents the assignment of QC *k* at hatch *b* at time period *t*. By

Constraint (4.1), each QC cannot move away from the vessel during the planning horizon. By Constraint (4.2), at most one QC can be positioned at a hatch at any time.

$$\sum_{b\in H} \eta_b^k(t) = 1 \qquad \forall k \in Q,\ t = 1,\cdots,T \tag{4.1}$$

$$\sum_{k\in Q} \eta_b^k(t) \le 1 \qquad \forall b \in H,\ t = 1,\cdots,T \tag{4.2}$$

2.3 Assignment constraints of QC position and task processing

$x_i(t)$ is a binary variable that represents the processing of an operation for task *i* at time period *t*. By Constraint (4.3), a QC can process its operation only when it is positioned at a hatch where the operation is.

$$x_i(t) \le \sum_{k\in Q} \eta_{b_i}^k(t) \qquad \forall i \in \Omega,\ t = 1,\cdots,T \tag{4.3}$$

2.4 Non-crossing constraints

QCs move along the same rail track on the quay, therefore they cannot cross each other's path. Constraints (4.4)-(4.6) describe the QC ordering conditions, where higher-indexed QCs must be positioned to the right of lower-indexed QCs.

$$\eta_b^k(t) + \sum_{m=1}^{b-1} \eta_m^{k+1}(t) \le 1 \qquad \forall b \in H\backslash\{1\},\ \forall k \in Q\backslash\{1\},\ t = 1,\cdots,T \tag{4.4}$$

$$\eta_b^k(t) \le \sum_{m=b+1}^{h} \eta_m^{k+1}(t) \qquad \forall b \in H\backslash\{h\},\ \forall k \in Q\backslash\{q\},\ t = 1,\cdots,T \tag{4.5}$$

$$\eta_b^k(t) \le \sum_{m=1}^{b-1} \eta_m^{k-1}(t) \qquad \forall b \in H\backslash\{1\},\ \forall k \in Q\backslash\{1\},\ t = 1,\cdots,T \tag{4.6}$$

2.5 Safety distance constraints between adjacent QCs

QCs should be apart from each other at least by the safety distance, δ. Constraints (4.7)-(4.8) ensure that any two adjacent QCs cannot be positioned within the safety distance at all times.

$$\sum_{m=\max(1,b-\delta)}^{b-1} \eta_m^k(t) \le 1 - \eta_b^{k+1}(t) \quad \forall k \in Q\backslash\{q\},\ \forall b \in H\backslash\{1\},\ t = 1,\cdots,T \tag{4.7}$$

$$\sum_{m=b+1}^{\min(b+\delta,h)} \eta_m^{k+1}(t) \le 1 - \eta_b^k(t) \quad \forall k \in Q\backslash\{q\},\ \forall b \in H\backslash\{h\},\ t = 1,\cdots,T \tag{4.8}$$

2.6 Maximum travel distance during one time period

There is a limit on the number of hatches along which a QC can travel during one time period. Constraint (4.9) limits the distance travelled by a QC during one time period to ζ. Let $o_i^k(t)$ denote a binary variable that represents the processing of an operation by QC *k* for task *i* at time period *t*. Constraint (4.10) then ensures that while travelling, a QC does not perform any operation.

$$\sum_{m=1}^{b-\zeta-1} \eta_m^k(t+1) + \sum_{n=b+\zeta+1}^{h} \eta_n^k(t+1) \le 1 - \eta_b^k(t) \quad \forall b \in H,\ \forall k \in Q,\ t = 1,\cdots,T-1 \tag{4.9}$$

$$\sum_{i\in\Omega} o_i^k(t+1) + \eta_b^k(t) \le 2 - \sum_{m=\max\{b-\zeta,1\}}^{b-1} \eta_m^k(t+1) - \sum_{n=b+1}^{\min\{b+\zeta,h\}} \eta_n^k(t+1) \quad \forall b \in H,\ \forall k \in Q,\ t = 1,\cdots,T-1 \tag{4.10}$$

2.7 Precedence relations among tasks and non-preemptive schedules

$s_i(t)$ is a binary variable that represents the start of task *i* at time period *t*. Constraint (4.12) enforces the non-preemptive scheduling assumption addressed in this model, and Constraint

(4.13) describes the precedence relations among tasks. Introducing $s_i(t)$ enables the enforcement of the non-preemptive schedules in the discretised planning horizon.

$$\sum_{t=1}^{T} s_i(t) = 1 \qquad \forall i \in \Omega \tag{4.11}$$

$$\sum_{l=1}^{t-1} x_i(l) + \sum_{l=t+N_i}^{T} x_i(l) = \big(1 - s_i(t)\big)M \qquad \forall i \in \Omega,\ t = 1, \cdots, T \tag{4.12}$$

$$\sum_{l=1}^{t-1} s_j(t) \leq 1 - s_i(t) \qquad \forall (i,j) \in \Phi,\ t = 2, \cdots, T \tag{4.13}$$

2.8 Sequence pairs for double cycles

$\omega^{uv}(t)$ is a binary variable that represents the multi-QC double cycling between QC *u* and QC *v* during time periods *t* and *t*+1. Constraint (4.14) ensures that the number of double cycling pairs that begin with QC *u* (by a loading operation) at time *t* is not greater than one. By the same token, Constraint (4.15) ensures that the number of double cycling pairs that end with QC *v* (by an unloading operation) at time period *t*+1 is not greater than one. $\omega^{uv}(t)$ is forced to be zero when there is no loading operation by QC *u* at time period *t* or there is no unloading operation by QC *v* at time period *t*+1.

$$\sum_{u \in Q \setminus \{v\}} \omega^{uv}(t) \leq 1 \qquad \forall v \in Q,\ t = 1, \cdots, T-1 \tag{4.14}$$

$$\sum_{v \in Q \setminus \{u\}} \omega^{uv}(t) \leq 1 \qquad \forall u \in Q,\ t = 1, \cdots, T-1 \tag{4.15}$$

$$\omega^{uv}(t) \leq \sum_{i \in \Lambda} o_i^u(t) \qquad \forall u, v \in Q\ (u \neq v), t = 1, \cdots, T-1 \tag{4.16}$$

$$\omega^{uv}(t) \leq \sum_{i \in \Upsilon} o_i^v(t+1) \qquad \forall u, v \in Q\ (u \neq v), t = 1, \cdots, T-1 \tag{4.17}$$

2.9 Work completion flag

Constraint (4.18) defines the work completion flag, $\gamma(t)$, that represents whether or not all container handlings have been completed by the end of the time period *t*. The objective function (4.19), which includes the term for maximising the weighted sum of $\gamma(t)$ during the planning horizon, ensures that when the value of the right-hand side of (4.18) sums to 1 for all tasks at the time period t, $\gamma(t)$ will be 1. If any of the tasks has not been completed by the end of the time period t, the right-hand side will be strictly less than 1, and thus $\gamma(t) = 0$.

$$\gamma(t) \leq \frac{\sum_{l=1}^{t} x_i(l)}{N_i} \qquad \forall i \in \Omega,\ t = 1, \cdots, T \tag{4.18}$$

2.10 Objective function

$$\text{miminize } \alpha\left(T - \sum_{t=1}^{T} \gamma(t)\right) - \beta \sum_{t=1}^{T-1} \sum_{u \in Q} \sum_{v \in Q \setminus \{u\}} \omega^{uv}(t) \tag{4.19}$$

While maximising the number of double cycles is an important goal in a double cycling problem, minimising the makespan can never be neglected as is often the case with any scheduling problem. In (4.19), therefore, we propose the objective function as the weighted sum of the makespan and the number of double cycling between QCs.

In order to tackle the complexity of this problem, we consider two-stage procedures in which the first stage is to decide the maximum number of double cycles in a single QC double cycling problem, and the second stage is to decide an appropriate number of double cycles in MQCDCP. The aim of the former is to reduce empty trips of QC cycles, and that of the latter is to reduce empty trips of transport vehicles while considering the makespan minimisation. The first-stage problem is well-studied in the existing literature (Goodchild & Daganzo, 2004, 2006, 2007; J.-H. Song, 2007), and its optimal solution can be obtained in polynomial time by reducing it to the two-machine flow shop scheduling formulation. Hence, we omit the first-stage procedure in our model and only assume that the number of single QC double

cycles per hatch is given as workload parameters to the model (as the results of the first-stage procedure).

We consider that the time spent on the tasks processed by single QC double cycles is converted into the terms of corresponding single cycles. Since each time period corresponds to the time taken for a single cycle, the time taken for a single QC double cycle should be calculated by multiplying the number of single QC double cycles by a parameter available from practice. According to the trial runs at the port of Tacoma in 2003, the average time for a single cycle was 1 minute 45 seconds, and for a double cycle it was 2 minutes 50 seconds (Goodchild & Daganzo, 2006). In this case, the time ratio of a single cycle to a double cycle is roughly 1.62, which can be the parameter for our data set.

2.11 Model size

The number of decision variables of the scheduling model has the relationship with the model size. In general, the bigger the number of decision variables in the model, the larger the model size. Let B denote the set of bays, Ω denote the set of tasks, Q denote the set of QCs and T denote the length of planning horizon. From the description of decision variables, the numbers of decision variables are:

- y_i^k: $|\Omega||Q|$;
- $s_i(t)$: $|\Omega|T$;
- $\eta_b^k(t)$: $|B||Q|T$;
- $x_i(t)$: $|\Omega|T$;
- $o_i^k(t)$: $|\Omega||Q|T$;
- $\omega^{uv}(t)$: $|Q|^2T$;
- $\gamma(t)$: T.

From this, the number of decision variables can be calculated to be $|\Omega||Q| + T(2|\Omega| + |B||Q| + |N||Q| + |Q|^2 + 1)$. The QCSP in general is in $\mathcal{NP}$-hard class, and the special case of the QCDCP (and MQCDCP) is harder than the QCSP. To overcome the computational burden of the exact solution, therefore, we study a heuristic approach in the next section.

3 Heuristic approach

In this section, we propose the use of a two-phased heuristic: construction phase and improvement phase. In the partial schedule of the construction phase, the first hatch to be engaged by each QC is decided randomly, and the successive construction of the partial schedule progresses based on a combination of the longest processing time (LPT) rule and the unidirectional schedule (UDS) rule. In the improvement phase, a local neighbourhood search is proposed based on a fast and easily implementable neighbourhood definition in the tabu search approach.

3.1 Construction phase

A set of initial solutions can be found by any heuristic methods, and for diversification we consider two heuristics: the LPT rule and the UDS rule. The algorithm based on the UDS rule is described in (Bierwirth & Meisel, 2009). The algorithm based on the LPT rule is that given the precedence relations and interference constraints, the tasks available for a freed QC are arranged in order of non-increasing processing time and the task with the longest total processing time is assigned to the QC. The assignment of a task-to-QC will be repeated iteratively until all the tasks are assigned. Further on, we present the details of the LPT rule employed in the construction.

Given the QC positions and hatches containing the tasks to be handled, we construct the processing order π of QCs with each QC starting its first task from a randomly assigned hatch. In a partial order, the QC that has the earliest finishing time will search for the next hatch to work on considering the remaining tasks available for it. The finding of such a task is based on the LPT rule, in which the processing time accounts for the travel time to the task in addition to the processing time of the task. If it fails to find the next hatch at the time of search mainly due to the interference constraints, the algorithm takes two successive measures: first, it evaluates the potential workloads of both left-hand and right-hand QCs and increases the probability of the more heavily-loaded QC finding the next hatch more away from the current QC; second, it forces the current QC to move closest to the more heavily-loaded QC and stay idle until the target QC finishes its ongoing task and moves further away.

The assignment of QC-to-task will progress iteratively until no more task is left to be assigned. The constructed schedule will be added to a candidate queue, and another schedule will be constructed from the beginning. Algorithm 3.1 describes the details of the aforementioned steps.

Algorithm 4.1 Partial order construction-phase algorithm

Step 1: Initialisation

Partition the workable hatch range of each QC;

Assign the initial hatch to each QC randomly.

Step 2: Assign Task-to-QC

Choose the QC with the earliest completion time from its partial order;

Find the suitable hatch to work next using the LPT rule;

If not found due to the interference with adjacent QCs, evaluate whether the current QC

- approaches either the left-side QC or the right-side QC depending on their workloads and stay idle until the target QC can move further away;
- moves away from the adjacent QC so that the adjacent QC can approach towards the current QC to work on the tasks that had previously been within the safety distance from the current QC;

In either case, once the current QC completes its move, it can stay idle until the completion time for the ongoing task of the adjacent QC.

Step 3: Repeat Task-to-QC Assignment

Repeat Step 2 until all tasks are assigned.

Step 4: Queue the optimistic solutions

Add the solution to a queue and repeat Step 1 iteratively for the given number of times, r;

Limit the queue size to a given number, q ($q < r$);

The queue will be prioritised by the objective value in each solution, therefore only the more optimistic solutions will remain in the queue.

The candidate queue prioritises more optimistic solutions over another. A parameter is given for the size of such queue, which will be used in the improvement phase as a pool to be used for the comparison among candidate schedules.

3.2 Improvement phase

Application of tabu search

Proposed by (Glover, 1986) and formalised by (Glover, 1989, 1990), tabu search is a global iterative optimisation method: the search moves from one solution to another, in order to improve the quality of the solutions visited. When the search arrives at a local optimum, it does not terminate but moves beyond the local optimum by choosing the best possible neighbour. Cycling back to previously visited solutions is prevented by the use of memories that record the recent visits during the search. The short-term memory framework enables us to keep the forbidden moves in a structure called *tabu list*. The type of tabu list one has chosen may forbid interesting moves, such as those that improve the best solution already found. An aspiration criterion may be employed to allow such moves. The size of the tabu list should be large enough to avoid cycling but small enough not to forbid too many moves. Although a short-term memory framework alone may be enough to obtain solutions superior to those found by conventional local search methods, a long-term mechanism can be implemented in order to diversity the search.

Neighbourhood

Among several ways of generating neighbourhoods, we follow the neighbourhood definition adapted by (Van Laarhoven, Aarts, & Lenstra, 1992) and (E. D. Taillard, 1994), which obtain a neighbourhood solution by permuting two successive and critical operations that use the same machine. To define the neighbourhood formally, we introduce the notations used in (Nowicki & Smutnicki, 1996) and modify them for our use.

Let π_k denote the processing order of tasks on machine k, and $\pi_k(i)$ denote the i^{th} task in the processing order by which machine k performs. Collectively, π is the processing order of tasks on all machines, i.e., $\pi = (\pi_k, \cdots, \pi_m)$ where m is the number of machines. Often a scheduling problem with precedence relations can be represented by a digraph. We consider the digraph $\mathrm{G}(\pi) = \{\Omega, E(\pi) \cup A_\Phi \cup E_\Psi\}$ where

$$E(\pi) = \bigcup_{k=1}^{m} \bigcup_{i} \{(\pi_k(i), \pi_k(i+1))\},$$

$$A_\Phi = \Phi \setminus E(\pi), and\ E_\Psi = \Psi \setminus E(\pi).$$

Makespan $C_{max}(\pi)$ for the processing order π equals the length of the longest path (critical path) in $G(\pi)$. A move v for the processing order π by a pair (x, y) of tasks can be defined when (i) *x* and *y* are successive tasks on some machine, and (ii) *x* and *y* are successive tasks on some critical path in $G(\pi)$. The size of this neighbourhood depends on the number of critical paths in π and the number of tasks on each critical path, and it is relatively a limited size. Obviously the CPU time of any tabu search implementation depends in principle on the computational complexity of the single neighbourhood search (and thus on the neighbourhood size) and we harness such complexity by considering only feasible interchanges of pairs for tasks on a machine in the critical path.

Let u denote a single (arbitrarily selected) critical path in $G(\pi)$. $V(\pi)$ defines a set of moves as interchanges between adjacent pair of tasks along the single critical path u. Let u_i denote the i^{th} task in the critical path u, and $|u|$ denote the number of tasks along the critical path u. Then, we can formally represent $V(\pi)$ as $V(\pi) = \bigcup_{i=1}^{|u|-1} V_i(\pi)$, where

$$V_i(\pi) = \begin{cases} \emptyset, & if\ (u_i, u_{i+1}) \in \Phi, \\ \{(u_i, u_{i+1})\}, & otherwise. \end{cases}$$

Let $Q(\pi, v)$ denote the processing order obtained by the application of move v to the processing order π. Neighbourhood $H(\pi)$ of π is then defined as all processing orders obtained by applying moves from $V(\pi)$, i.e.,

$$H(\pi) = \{Q(\pi, v) | v \in V(\pi)\}.$$

Tabu list

We define the tabu list as $T = (T_1, \cdots, T_{maxt})$ where $maxt$ is the fixed size of the tabu list and $T_i \in \Omega^2$ is a forbidden move, where $1 \leq i \leq maxt$. Tabu is introduced to keep the search from performing previously-visited moves. In its simple form, the tabu list is initialised with an empty list and a move v which leads to a local optimum is always added to tabu list T. If the tabu list is full, some element in the tabu list will expire. In general, elements expire from the list in the same order they are added. That is, all elements in tabu list T are shifted to the

left with the first element being removed, and move v is put on the position $maxt$, i.e., $T_i \coloneqq T_{i+1}$ $(i = 1, \cdots, maxt - 1)$ and $T_{maxt} \coloneqq v$.

Search process

First, we take the q number of processing orders from the queue derived by the aforementioned construction heuristic. Then, the improvement heuristic runs the q number of repeated tabu searches with each search starting with a given processing order π and with an empty tabu list $(T = \emptyset)$. At each iteration, we find the set of moves $V(\pi)$ along the then critical path u. Each move $v \in V(\pi)$ yields the corresponding processing order $\pi' = Q(\pi, v)$, which is then fed into the tabu list T. The best found solution s^* with respect to the objective function and associated processing order π^* are currently updated. The search of each iteration stops either when $V(\pi)$ is empty or when the number of iterations $(iter)$ without improving the best found solution s^* exceeds the fixed number for maximum iterations $(maxiter)$. Algorithm 3.2 gives the summary for these steps.

Algorithm 4.2 Improvement-phase 1

Step 1: Initialisation

Start with a list of candidate schedules;

Set the first candidate from the list of candidate schedules to the best incumbent π^*;

Initialise $T = \emptyset, iter = 0$.

Step 2: Generate a potential set of neighbours

Take a seed schedule π from the candidate schedules;

Set $iter \coloneqq iter + 1$;

Find $V(\pi)$ from π;

Generate a subset of neighbour $H(\pi)$ such that move v is not in the tabu list and aspiration criteria hold.

Step 3: Update the best processing order π^*

Choose a best processing order π' from Step 2;

Update the best incumbent π^* if the evaluation of π' is better than that of π^*.

Step 4: Update tabu list

If the tabu list is full, remove the first element from the tabu list;

Add π' (or the move v that leads to π') to the tabu list.

Step 5: Check stopping conditions.

If a stopping condition is met, then stop;

Otherwise, go to Step 2.

Improvement for the number of double cycles

Analogous to the concepts for the binary variable, $\omega^{uv}(t)$, the number of double cycles in the schedules generated by the heuristic is calculated as follows:

1) Take the makespan as the parameter for the number of iterations;
2) Take the number of QCs engaged in loading operation at time period *t* and the number of QCs engaged in unloading operation at time period *t*+1, then the lesser number becomes the number of double cycles linked to time segment *t*;

The above calculation rule is valid under the assumption that each task is comprised of only a single type of operation: either loading or unloading. If we have the operation type dedicated to single-QC double cycling, the number of such cycles will be weighted to the term *double cycles* in the objective function.

To demonstrate how the number of double cycles is calculated, an illustrative example is given in Figure 3.1 where QC1 is positioned at the left-most bay and QC4 at the right-most bay. The figure illustrates a schedule generated by the heuristic for instance *k*54 from (Kim & Park, 2004), the task details of which are described in Table 3.1. The schedule results of the example are analysed for each time period from 1 to [*makespan*-1]: For example, at time period *40* all four QCs are working on unloading operations, thus no double cycle is identified at this time period; At time period *80*, one QC (QC1) is engaged in an unloading operation whereas the other three QCs are engaged in loading operations, thus one double cycle is identified at this time period; At time period *160*, one double cycle is identified with QC1 being in progress of travelling, QC2 and QC4 at unloading operations, and QC3 at a loading operation; At time period *290*, two QCs (QC1, QC2) are engaged in a loading operation and the other two QCs (QC3, QC4) are engaged in unloading operations, thus two double cycles can be identified at this time period.

Table 3.1 Task details of instance *k*54

Bay	Task id	No. of containers	Operation type

1	30	42	Unloading
	1	21	Unloading
	4	22	Loading
2	18	56	Unloading
	11	48	Loading
4	7	44	Unloading
7	19	36	Unloading
	26	25	Unloading
	5	12	Loading
8	6	49	Unloading
	17	1	Loading
9	20	4	Unloading
14	23	21	Unloading
	28	21	Unloading
15	10	27	Loading
16	13	43	Loading
17	15	58	Loading
18	14	47	Unloading
19	2	26	Unloading
	3	57	Loading
20	27	55	Unloading
	12	14	Loading
21	16	37	Loading
23	21	37	Unloading
	8	15	Unloading
	9	2	Loading
24	25	53	Loading
26	24	11	Unloading
30	29	55	Unloading
	22	47	Loading

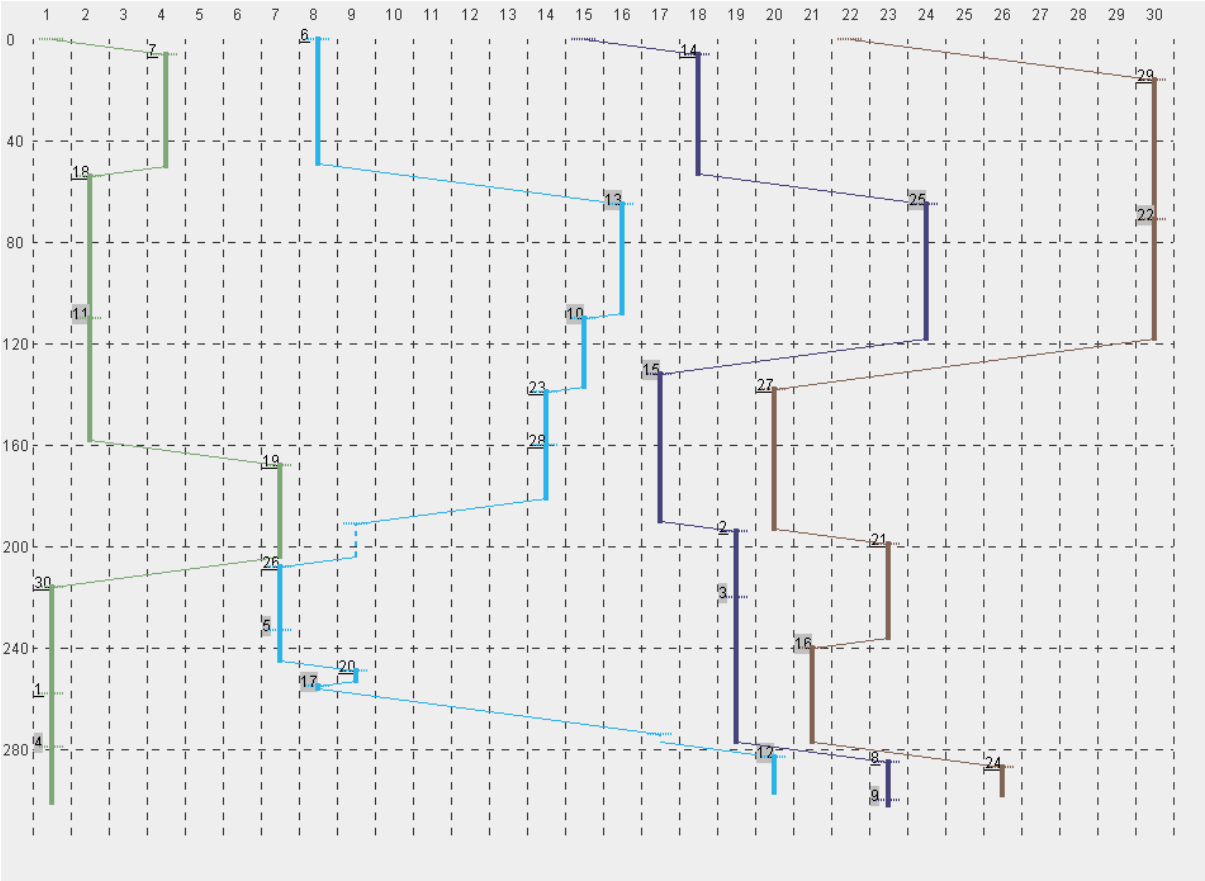


Figure 3.1 Schedule chart generated by the heuristic for instance *k*54 (Table 3.1)

In the aforementioned neighbourhood search, the swapping move between tasks may occur only among the adjacent tasks in the critical path. Now, we consider additional improvements in terms of the number of double cycles given a schedule. This can be done by shifting idle times in non-critical paths where there are tasks immediately before or after the idle times. A simple example is illustrated in Figure 3.2, in which the number of double cycles could be increased with QC1 delaying the unloading tasks in bay 5 to be completed after the idle times. This shift between tasks and idle times in non-critical paths can improve the objective value for the number of double cycles while holding the makespan the same. To save computation time, the algorithm limits this search only to the best schedule found by Algorithm 3.2. In the next section, we provide a computational analysis of this approach.

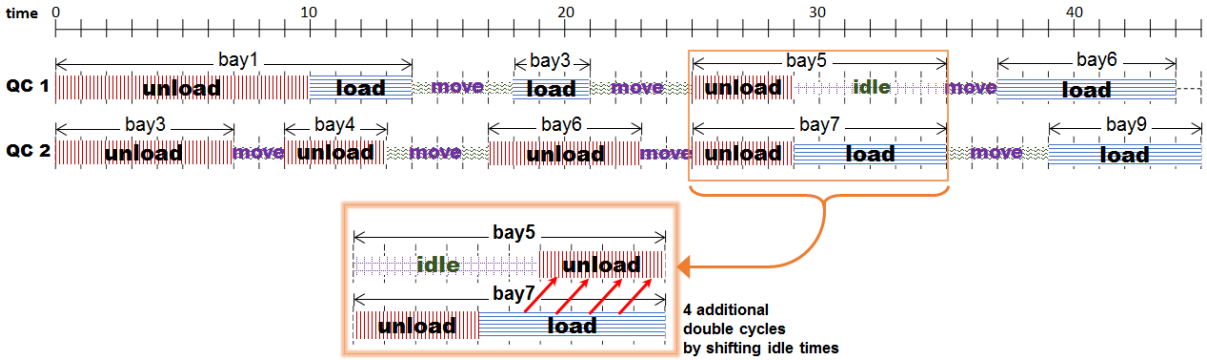


Figure 3.2 An example of improvement for the number of double cycles by shifting idle times

4 Computational experiments

We have tested our approach on the popular benchmark suite proposed by (Kim & Park, 2004). The IP formulation is implemented with a commercial solver (Gurobi 6) and is executed on a desktop PC running on 64-bit Windows 7 OS, Intel i5-2500 Quad 3.3 GHz CPU and 8GB of RAM. The programming language used for the heuristic implementation is Java (version 1.7). Due to a substantially large number of binary variables being generated, even the smallest group of instance size (10 tasks, 2 QCs) takes several hours to find the optimal solution with the proposed IP model, e.g., it takes 455 minutes to solve instance *k*13. With the IP model, the complexity in terms of the number of decision variables does not depend on the number of tasks or QCs. Instead, the processing time of each task or the makespan of the entire schedule is a significant determinant of the complexity in the IP solution because the binary variables are generated based on the discretised planning horizon. That is, the longer the makespan, the more sets of binary variables to decide.

In the first analysis, we give the comparison between the exact solution and the heuristic approach for a small-scale instance group (*k*13 – *k*22), i.e., 10 tasks and 2 QCs. We consider the objective function '$\alpha C_{max} - \beta W$' where W is the number of double cycles with weights given by $\alpha = 3$ and $\beta = 1$. Due to the objective function considering the number of double cycles, which other research does not take into account in their objective function, the makespans reported here are just for a reference, not a direct measure for benchmarking by this approach.

Table 4.1 Results of the IP and the heuristic for small-scale instances

No	IP					Heuristic			Gap*
	makespan	dbl cycle	Obj. value (Z_{best})	LB (Z_{LB})	time (min)	makespan	dbl cycle	obj. value (Z_H)	
*k*13	151	103	350		445	152	101	358	2.3%
*k*14			n/a		∞	182	130	431	n/a
*k*15			n/a		∞	175	97	452	n/a
*k*16			≤258	224.68	∞	112	58	296	≥14.7%
*k*17			n/a		∞	162	95	406	n/a
*k*18			n/a		∞	126	89	319	n/a
*k*19			n/a		∞	187	101	475	n/a
*k*20			n/a		∞	134	90	327	n/a
*k*21			n/a		∞	157	112	377	n/a
*k*22			n/a		∞	181	120	453	n/a

*: Optimality Gap = $(\boldsymbol{Z_H - Z_{IP}})/\boldsymbol{Z_{IP}}$

∞: time bounded (24 h)

LB: lower bound for the time-bounded instances

In the second analysis, the heuristic results are reported for the instances of small-to-medium scales (*k*13–*k*52). The number of double cycles cannot exceed the lesser number of the loading quantity and the unloading quantity. Thus, the ratio in the rightmost column of Table 4.2 is calculated with respect to the number of double cycles from the generated schedule into the upper bound number of possible double cycles, which is the minimum value among the loading quantity and the unloading quantity. The best known makespans are presented for a reference from the existing literature (Meisel & Bierwirth, 2011b). Since the reported solutions are the weighted makespans in which each makespan is multiplied by the weight of three, we report the best-known makespans in Table 4.2 by dividing each solution into three. On average, higher ratios for the number of double cycles are observed from the smaller scale instances.

Table 4.2 Heuristic results for the ratio of double cycles to the upper bound number of double cycles

No	best-known makespan*	heuristic makespan	**(a)** no. double cycle	obj. value	**(b)** no. loads	**(c)** no. unloads	ratio (%): **(a)**/min**(b,c)**
*k*13	151	152	101	358	104	162	97.1%
*k*14	182	182	130	431	164	183	79.3%
*k*15	171	175	97	452	219	106	91.5%
*k*16	104	112	58	296	120	67	86.6%
*k*17	151	162	95	406	114	169	83.3%
*k*18	125	126	89	319	98	135	90.8%
*k*19	181	187	101	475	224	112	90.2%
*k*20	133	134	90	327	154	97	92.8%
*k*21	155	157	112	377	174	126	88.9%
*k*22	180	181	120	453	145	197	82.8%
*k*23	192	201	55	626	69	295	79.7%
*k*24	222	239	144	633	194	231	74.2%
*k*25	246	264	128	706	247	221	57.9%
*k*26	213	224	128	595	151	256	84.8%
*k*27	219	229	131	610	188	230	69.7%
*k*28	177	184	107	523	114	218	93.9%
*k*29	269	278	143	751	315	203	70.4%
*k*30	297	312	226	872	318	259	87.3%
*k*31	190	207	96	549	179	172	55.8%
*k*32	197	212	142	557	220	157	90.4%
*k*33	201	218	167	547	253	325	66.0%
*k*34	239	249	248	613	278	413	89.2%
*k*35	228	248	177	603	230	424	77.0%
*k*36	226	254	206	589	410	240	85.8%
*k*37	170	185	156	456	255	230	67.8%
*k*38	206	222	126	615	129	455	97.7%
*k*39	171	207	124	533	345	136	91.2%
*k*40	188	206	138	546	161	372	85.7%
*k*41	196	205	137	547	183	379	74.9%
*k*42	191	206	145	596	267	266	54.5%
*k*43	292	315	221	781	383	454	57.7%
*k*44	274	295	233	736	417	370	63.0%
*k*45	278	301	215	799	510	286	75.2%
*k*46	230	269	204	708	340	317	64.4%

*k*47	264	291	243	738	452	310	78.4%
*k*48	213	233	159	669	232	366	68.5%
*k*49	298	333	245	814	474	381	64.3%
*k*50	247	261	216	804	383	325	66.5%
*k*51	266	287	213	750	372	387	57.3%
*k*52	320	345	251	859	483	440	57.0%

*: summarised in **(Meisel & Bierwirth, 2011b)**

5 Conclusion and future research

This paper developed a mathematical formulation for the MQCDCP in the discretised planning horizon. It extends the existing body of work by considering multiple QCs, and addresses non-interference constraints and safety distance between QCs. The proposed IP model takes long computation time even for instances of small scale. To tackle the computational difficulty, a two-phase heuristic is studied: construction phase and improvement phase. In the construction phase, initial solutions can be found with any heuristic methods. For diversification, we construct the initial solutions by two heuristics: the UDS rule and the LPT rule. In the improvement phase, a local search based on tabu search algorithm is employed along the critical path. Further to improve the number of double cycles, additional local search is employed along the non-critical paths of the best known solution found by the TS.

The objective of the problem is aimed at minimising the weighted sum of the makespan and the negative number of double cycles. The number of double cycles can be calculated by considering the number of QCs working on loading operation at time period t and the number of QCs working on unloading operation at time period t+1. With the objective function proposed for this solution, computational results of the benchmark comparison between the exact solution and the heuristic are available for the instances of the smallest scale. The heuristic results also provide the ratio of the number of double cycles to the upper bound number of possible double cycles. Its analysis indicates that a substantially large percentage of double cycling operation can be achieved for small scale instances and a medium-to-large percentage of double cycling operation can be done for medium scale instances. However, the higher percentage of double cycling operation may be achieved at the sacrifice of the

inflated makespan, the degree of which can be adjusted by the weights in the objective function.

Future research on the MQCDCP can go in several directions. For the exact solution, research efforts need to be made to address the limitations of the model under the discretised planning horizon. As a downside of such model, the computational complexity largely depends on the length of the planning horizon rather than the size of the tasks or QCs. Because the factor of the planning horizon tends to be substantially larger than that of the tasks or QCs, the solution approach whose complexity is based on the length of the planning horizon is better to be avoided as much as possible. For the heuristic approach, more sophisticated solution approaches exploiting the characteristics of double cycling operation need to be developed.

Acknowledgements

Authors thank Kap-Hwan Kim (Pusan National University, Korea) and Young-Man Park (Korea Naval Academy, Korea) for providing the benchmark suite.

Appendix. Multi-QC double cycling formulation

With the assumptions, notations and constraints described in Section 2, we present a full IP model for the MQCDCP as follows.

$$\text{miminize } \alpha\left(T - \sum_{t=1}^{T} \gamma(t)\right) - \beta \sum_{t=1}^{T-1} \sum_{u \in Q} \sum_{v \in Q \setminus \{u\}} \omega^{uv}(t) \tag{A.1}$$

subject to

$$\sum_{b \in H} \eta_b^k(t) = 1 \qquad \forall k \in Q,\ t = 1, \cdots, T \tag{A.2}$$

$$\sum_{k \in Q} \eta_b^k(t) \leq 1 \qquad \forall b \in H,\ t = 1, \cdots, T \tag{A.3}$$

$$x_i(t) \le \sum_{k\in Q} \eta_{b_i}^k(t) \qquad \forall i \in \Omega,\ t = 1,\cdots,T \qquad \text{(A.4)}$$

$$\sum_{k\in Q} y_i^k = 1 \qquad \forall i \in \Omega \qquad \text{(A.5)}$$

$$\sum_{u\in Q\backslash\{k\}} \eta_{b_i}^u(t) \le 2 - x_i(t) - y_i^k \qquad \forall i \in \Omega,\ \forall k \in Q,\ t = 1,\cdots,T \qquad \text{(A.6)}$$

$$\sum_{t=1}^{T} x_i(t) = N_i \qquad \forall i \in \Omega \qquad \text{(A.7)}$$

$$\sum_{i\in\Omega} x_i(t) \le |Q| \qquad t = 1,\cdots,T \qquad \text{(A.8)}$$

$$\sum_{k\in Q} o_i^k(t) = x_i(t) \qquad \forall i \in \Omega,\ t = 1,\cdots,T \qquad \text{(A.9)}$$

$$o_i^k(t) \le 0.5\{\eta_{b_i}^k(t) + x_i(t)\} \qquad \forall i \in \Omega,\ \forall k \in Q,\ t = 1,\cdots,T \qquad \text{(A.10)}$$

$$\eta_b^k(t) + \sum_{m=1}^{b-1} \eta_m^{k+1}(t) \le 1 \qquad \forall b \in H\backslash\{1\},\ \forall k \in Q\backslash\{1\},\ t = 1,\cdots,T \qquad \text{(A.11)}$$

$$\eta_b^k(t) \le \sum_{m=b+1}^{h} \eta_m^{k+1}(t) \qquad \forall b \in H\backslash\{h\},\ \forall k \in Q\backslash\{q\},\ t = 1,\cdots,T \qquad \text{(A.12)}$$

$$\eta_b^k(t) \le \sum_{m=1}^{b-1} \eta_m^{k-1}(t) \qquad \forall b \in H\backslash\{1\},\ \forall k \in Q\backslash\{1\},\ t = 1,\cdots,T \qquad \text{(A.13)}$$

$$\sum_{m=\max(1,b-\delta)}^{b-1} \eta_m^k(t) \le 1 - \eta_b^{k+1}(t) \quad \forall k \in Q\backslash\{q\},\ \forall b \in H\backslash\{1\},\ t = 1,\cdots,T \qquad \text{(A.14)}$$

$$\sum_{m=b+1}^{\min(b+\delta,h)} \eta_m^{k+1}(t) \le 1 - \eta_b^k(t) \qquad \forall k \in Q\backslash\{q\},\ \forall b \in H\backslash\{h\},\ t = 1,\cdots,T \qquad \text{(A.15)}$$

$$\sum_{m=1}^{b-\zeta-1} \eta_m^k(t+1) + \sum_{n=b+\zeta+1}^{h} \eta_n^k(t+1) \le 1 - \eta_b^k(t) \qquad \text{(A.16)}$$

$$\forall b \in H,\ \forall k \in Q,\ t = 1, \cdots, T-1$$

$$\sum_{i\in\Omega} o_i^k(t+1) + \eta_b^k(t) \le 2 - \sum_{m=\max\{b-\zeta,1\}}^{b-1} \eta_m^k(t+1) - \sum_{n=b+1}^{\min\{b+\zeta,\mathrm{h}\}} \eta_n^k(t+1) \quad (A.17)$$

$$\forall b \in H,\ \forall k \in Q,\ t = 1, \cdots, T-1$$

$$\sum_{t=1}^{T} s_i(t) = 1 \qquad \forall i \in \Omega \quad (A.18)$$

$$\sum_{l=1}^{t-1} x_i(l) + \sum_{l=t+N_i}^{T} x_i(l) = \big(1 - s_i(t)\big)M \quad (A.19)$$

$$\forall i \in \Omega,\ t = 1, \cdots, T$$

$$\sum_{l=1}^{t-1} s_j(t) \le 1 - s_i(t) \qquad \forall (i,j) \in \Phi,\ t = 2, \cdots, T \quad (A.20)$$

$$\sum_{u\in Q\setminus\{v\}} \omega^{uv}(t) \le 1 \qquad \forall v \in Q,\ t = 1, \cdots, T-1 \quad (A.21)$$

$$\sum_{v\in Q\setminus\{u\}} \omega^{uv}(t) \le 1 \qquad \forall u \in Q,\ t = 1, \cdots, T-1 \quad (A.22)$$

$$\omega^{uv}(t) \le \sum_{i\in\Lambda} o_i^u(t) \qquad \forall u, v \in Q\ (u \ne v), t = 1, \cdots, T-1 \quad (A.23)$$

$$\omega^{uv}(t) \le \sum_{i\in\Upsilon} o_i^v(t+1) \qquad \forall u, v \in Q\ (u \ne v), t = 1, \cdots, T-1 \quad (A.24)$$

$$\gamma(t) \le \frac{\sum_{l=1}^{t} x_i(l)}{N_i} \qquad \forall i \in \Omega,\ t = 1, \cdots, T \quad (A.25)$$

$$y, s, x, \eta, o, \omega, \gamma \qquad \text{binary.} \quad (A.26)$$

where M is a sufficiently large positive number.

The objective function given in (A.1) aims at minimising the weighted sum of the makespan and the number of double cycling between QCs. Constraints (A.2) and (A.3) ensure that all QCs cannot move away from the vessel at all times and that at most one QC can be positioned at a hatch at any time. Constraint (A.4) ensures that a QC must be positioned at a hatch if it is working there. Constraints (A.5) and (A.6) ensure that a task is assigned to only one QC and that while a QC is working on a task, no other QCs can be positioned at the hatch of the task. Constraints (A.7) and (A.8) give the definition of $x_i(t)$, and Constraints (A.9) and (A.10) give the definition of $o_i^k(t)$. Constraints (A.11)-(A.13) satisfy non-crossing constraints among QCs, and Constraints (A.14) and (A.15) enforce the safety distance between adjacent QCs at all times. Constraint (A.16) specifies a maximum number of hatches along which a QC can travel during a time period, and Constraint (A.17) enforces that the QC is not allowed to work during the travel time. Constraint (A.18)-(A.20) define precedence relations among tasks. Constraints (A.21)-(A.24) define $\omega^{uv}(t)$ for counting the number of double cycling. Constraint (A.25) defines $\gamma(t)$, which is the work completion flag at time period *t*. The aim of maximising the objective function (A.1) with $\alpha > 0$ enforces $\gamma(t) = 1$ when all tasks have been completed by the end of time period *t*. Constraint (A.26) restricts all decision variables to binary domains.

References

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.

Bierwirth, C., & Meisel, F. (2014). A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research, 13*(5), 533-549.

Glover, F. (1989). Tabu search-part I. *ORSA Journal on computing, 1*(3), 190-206.

Glover, F. (1990). Tabu search-part II. *ORSA Journal on computing, 2*(1), 4-32.

Goodchild, A. V. (2005). *Crane double cycling in container ports: algorithms, evaluation, and planning.* (PhD thesis), University of California at Berkeley, CA.

Goodchild, A. V., & Daganzo, C. F. (2004). *Reducing ship turn-around time using double-cycling*. Institue for Transportation Studies Research Report (UCB).Retrieved from http://escholarship.org/uc/item/86r4p6sc

Goodchild, A. V., & Daganzo, C. F. (2006). Double-cycling strategies for container ships and their effect on ship loading and unloading operations. *Transportation Science, 40*(4), 473-483.

Goodchild, A. V., & Daganzo, C. F. (2007). Crane double cycling in container ports: planning methods and evaluation. *Transportation Research Part B: Methodological, 41*(8), 875-891.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics, 5*, 287-326.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval research logistics quarterly, 1*(1), 61-68.

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

Ku, D., & Arthanari, T. S. (2014). On double cycling for container port productivity improvement. *Annals of Operations Research*, 1-16. 10.1007/s10479-014-1645-z

Lee, C.-Y., Liu, M., & Chu, C. (2014). Optimal Algorithm for the General Quay Crane Double-Cycling Problem. *Transportation Science*

Meisel, F., & Bierwirth, C. (2011b). A unified approach for the evaluation of quay crane scheduling models and algorithms. *Computers & Operations Research, 38*(3), 683-693.

Meisel, F., & Wichmann, M. (2010). Container sequencing for quay cranes with internal reshuffles. *OR spectrum, 32*(3), 569-591.

Michael, P. (1995). Scheduling, theory, algorithms, and systems. *Englewood Cli s, New Jersey*

Nowicki, E., & Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Management science, 42*(6), 797-813.

Sidney, J. B. (1979). The two-machine maximum flow time problem with series parallel precedence relations. *Operations research, 27*(4), 782-791.

Song, J.-H. (2007). A study for optimization of double cycling in container ports. *Port Technology International, Edition 36*, 50-52. Retrieved from http://www.porttechnology.org/technical_papers/a_study_for_optimisation_of_double_cycling_in_container_ports/

Taillard, E. D. (1994). Parallel taboo search techniques for the job shop scheduling problem. *ORSA Journal on computing*, *6*(2), 108-117.

Van Laarhoven, P. J., Aarts, E. H., & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations research, 40*(1), 113-125.

Zhang, H., & Kim, K. H. (2009). Maximizing the number of dual-cycle operations of quay cranes in container terminals. *Computers & Industrial Engineering*, *56*(3), 979-992.

Conclusion

We have so far investigated the seaside planning and operation problems in general and the crane double cycling models specifically. In this chapter, we provide a summary of research results and contributions. Future research directions are also discussed.

1 Summary of research results

Paper I provides the literature review on the QCSP in a comprehensive manner. It then categorises the QC scheduling problems by the granularity of tasks and whether or not the schedules are preemptible. Some of the studies classified as tasks of individual containers in the survey by (Bierwirth & Meisel, 2010) are reclassified into preemptive version of bay-wise tasks. As the model proposal, the paper fills the gap in the body of the literature by developing an MIP model for a preemptive version of bay operation. No such exact model has been presented in the literature except for heuristics as in (Lu et al., 2012). The relaxed formulations using Lagrangian relaxation (and LP relaxation) are compared in computational experiments. As a complementary study to **Paper II** and **Paper III**, this paper provides a general perspective on the QCSP with an exact formulation for the model being considered in the recent research trend.

Paper II is the paper published in a journal as the motivation of this thesis. It investigates double cycling models in the existing literature and finds that a breakthrough in modelling the multi-QC double cycling formulation is needed. It also addresses that in the existing literature, the granularity of tasks for double cycling is too much focused on the stack-based QCSP. Other types of granularity need to be examined as a potential towards the breakthrough. A

follow-up research which identifies and fills the gaps presented in this paper is linked to **Paper III**.

Paper III finally proposes a model for the multi-QC double cycling operation that this whole thesis is aimed for. The IP model is formulated under the discretised planning horizon for a vessel operation and each time interval in the planning horizon corresponds to the time period during which a container handling takes place. The objective is represented by a function of weighted sum with weights assigned to the makespan and the number of double cycling pairs between QCs. In the discretised planning horizon, all decision variables are binary variables. In the objective function, for example, $\gamma(t)$ is a binary variable that represents whether or not all container handlings have been completed by the end of time period t, and $\omega^{uv}(t)$ is a binary variable that represents the multi-QC double cycling pair between QC u (for loading) and QC v (for unloading) during time periods t and $t+1$. The tractable problem size is very limited with this approach as is often the case with the model on a discretised planning horizon. Proposed heuristic approaches using tabu search attempt to mitigate such computational challenge.

2 Contributions

First, the contribution of **Paper I** is in providing an exact model in the bay-wise, preemptive operation under unidirectional QC schedules. The development of an exact model enables several ways of relaxations, including the Lagrangian relaxation included in this paper, to be investigated in the follow-up research. We have found that the use of Lagrangian relaxation to the model gives a very tight lower bound to some instances that were previously hard to solve. This improvement became only possible with some modifications in the traditional model configuration, i.e., non-preemptive cluster-based tasks → preemptive bay-wise tasks, and bidirectional crane movement → unidirectional crane movement. Although the modification in the model configuration implies some form of relaxations or adjustments from the traditional model configuration, this type of modification is still valid by meeting the constraints originating from real-world requirements and in most cases, this modified model reflects even better the practice in the real-world terminal operation. This claim is

justifiable in the most up-to-date research trends using this configuration: see (Bierwirth & Meisel, 2009; Chen et al., 2014; Legato & Trunfio, 2014; Lu et al., 2012).

Second, as much as is the case for **Paper I**, the main contribution in **Paper III** is also the development of an exact formulation for crane double cycling strategy, and specifically multi-QC double cycling model. This contribution was made possible as a follow-up research addressed to the problem identified in **Paper II**. As a trade-off between the solution quality and computation time, the applied meta-heuristic approach presents a way to tackle the computational difficulty implied in the nature of the model.

3 Future research direction

The objective of a mathematical model in the QCSP is usually the minimisation of the makespan of the vessel operation or the minimisation of the delay of the vessel departure. If the model implies a planning horizon of multiple vessels, the corresponding objectives are the minimisation of the sum of the makespan or the sum of the delays for the multiple vessels. However, as studied by (J. Liu et al., 2006), the relative tardiness criterion rather than the total tardiness criterion can be better at least for a type of a rolling horizon model. It avoids scenarios that a few vessels suffer most of the tardiness, which may happen if the objective is to minimise the total tardiness. This idea fits the reality: Captains of ships tend not to complain about a small amount of delays whereas any captain whose ship is seriously delayed at a marine terminal will complain severely no matter how much the total tardiness of his or her ship has been reduced. Hence, it is conceivable to reflect a psychological factor in formulating the mathematical objective for the optimisation problems such as terminal operation.

The use of the discretised planning horizon in **Paper III** increases the number of decision variables in the model faster than the model of not doing so. Hence, we note that this phenomenon complicates the optimisation model very severely, the problem class of which is already complex enough by its $\mathcal{NP}$-completeness. As such, it is considerable to develop a model whose main decision variables are task-dependent, not time-dependent. In fact, adding the term for double cycling to the objective function complicates this line of research

direction. It is hoped that follow-up research can address this difficulty adequately and complement the current form of research. Also, the efficiency and/or effectiveness of the proposed heuristics need to be supported or rebutted with further evidence. Hence, another direction of future research lies in developing more advanced heuristic approaches. In this regard, it is planned to study the formulated model with more sophisticated meta-heuristics in future.

Bibliography

Agrawal, M., Kayal, N., & Saxena, N. (2004). PRIMES is in P. *Annals of mathematics*, 781-793.

Applegate, D., Bixby, R., Chvátal, V., & Cook, W. (2003). Concorde TSP Solver. Retrieved 28 June, 2015, from http://www.math.uwaterloo.ca/tsp/concorde/index.html

Avazbeigi, M. (2009). An Overview of Complexity Theory *Facility Location* (pp. 19-36): Springer.

Avery, P. (2011). LinkedIn Discussions on Double-Cycling Technique. Retrieved 11 June, 2012, from http://www.linkedin.com/groups/Doublecycling-technique-1947860.S.82027223

Avriel, M., Penn, M., Shpirer, N., & Witteboon, S. (1998). Stowage planning for container ships to reduce the number of shifts. *Annals of Operations Research, 76*, 55-71.

Baker, K. R., & Trietsch, D. (2009). *Principles of sequencing and scheduling*: Wiley.

Bazaraa, M. S., & Sherali, H. D. (1981). On the choice of step size in subgradient optimization. *European Journal of Operational Research, 7*(4), 380-388.

Bendall, H. B., & Stent, A. F. (1996). Hatchcoverless container ships: Productivity gains from a new technology. *Maritime Policy & Management, 23*(2), 187-199.

Bertsekas, D. P. (1999). *Nonlinear programming* (2nd ed. ed.). Belmont, MA: Athena Scientific.

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.

Bierwirth, C., & Meisel, F. (2014). A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR), 35*(3), 268-308.

Boyd, S., Xiao, L., & Mutapcic, A. (2003). *Subgradient Methods*. Stanford University: Retrieved from https://web.stanford.edu/class/ee392o/subgrad_method.pdf

Boysen, N., Emde, S., & Fliedner, M. (2012). Determining crane areas for balancing workload among interfering and noninterfering cranes. *Naval Research Logistics (NRL), 59*(8), 656-662.

Bremermann, H. J. (1958). *The evolution of intelligence: The nervous system as a model of its environment*. Seattle, WA: University of Washington, Department of Mathematics.

Brown, G. G., Lawphongpanich, S., & Thurman, K. P. (1994). Optimizing ship berthing. *Naval Research Logistics (NRL), 41*(1), 1-15.

Btażewicz, J., Salvador, A. M., & Walkowiak, R. (2002). Tabu search for two-dimensional irregular cutting. In C. C. Ribeiro & P. Hansen (Eds.), *Essays and Surveys in Metaheuristics* (pp. 101-128): Kluwer Academic Publishers.

Cao, J., Shi, Q., & Lee, D.-H. (2010). Integrated quay crane and yard truck schedule problem in container terminals. *Tsinghua Science & Technology, 15*(4), 467-474.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2013). Seaside operations in container terminals: Literature overview, trends, and research directions. *Flexible Services and Manufacturing Journal*, 1-39.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications, 45*(1), 41-51.

Chen, J. H., Lee, D.-H., & Cao, J. X. (2011). Heuristics for quay crane scheduling at indented berth. *Transportation Research Part E: Logistics and Transportation Review, 47*(6), 1005-1020.

Chen, J. H., Lee, D.-H., & Goh, M. (2014). An effective mathematical formulation for the unidirectional cluster-based quay crane scheduling problem. *European Journal of Operational Research, 232*(1), 198-208.

Choo, S., Klabjan, D., & Simchi-Levi, D. (2010). Multiship crane sequencing with yard congestion constraints. *Transportation Science, 44*(1), 98-115.

Christofides, N., & Whitlock, C. (1977). An algorithm for two-dimensional cutting problems. *Operations research*, 30-44.

Chung, S., & Chan, F. T. (2013). A workload balancing genetic algorithm for the quay crane scheduling problem. *International Journal of Production Research, 51*(16), 4820-4834.

Chung, S. H., & Choy, K. L. (2012). A modified genetic algorithm for quay crane scheduling operations. *Expert Systems with Applications, 39*(4), 4213-4221.

Clausen, J. (1999). Branch and bound algorithms-principles and examples. from Department of Computer Science, University of Copenhagen

Cook, S. A. (1971). *The complexity of theorem-proving procedures.* Paper presented at the Proceedings of the third annual ACM symposium on Theory of computing.

Cook, W. (2012). Markowitz and Manne + Eastman + Land and Doig = Branch and Bound. *Optimization Stories, Documenta Mathematica book series, Extra volume,* 227-238.

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

Daskin, M. S. (2011). *Network and discrete location: models, algorithms, and applications*: John Wiley & Sons.

De Werra, D., & Hertz, A. (1989). Tabu search techniques. *Operations-Research-Spektrum, 11*(3), 131-141.

Degano, C., & Pellegrino, A. (2002. *Multi-Agent Coordination and Collaboration for Control and Optimization Strategies in an Intermodal Container Terminal*. Paper presented at the IEEE International Engineering Management Conference (IEMC-2002), Cambridge, UK

Demirkol, E., Mehta, S., & Uzsoy, R. (1998). Benchmarks for shop scheduling problems. *European Journal of Operational Research, 109*(1), 137-141.

Diabat, A., & Theodorou, E. (2014). An integrated quay crane assignment and scheduling problem. *Computers & Industrial Engineering, 73*, 115-123.

Dowd, T. J., & Leschine, T. M. (1990). Container terminal productivity: a perspective. *Maritime Policy and Management, 17*(2), 107-112.

Dowsland, K. A. (1993). Simulated annealing. In C. R. Reeves (Ed.), *Modern heuristic techniques for combinatorial problems* (pp. 20-69). New York: John Wiley & Sons, Inc.

Eastman, W. L. (1958). *Linear programming with pattern constraints.* (PhD PhD Thesis), Harvard University, MA.

Etcheberry, J. (1977). The set-covering problem: A new implicit enumeration algorithm. *Operations research, 25*(5), 760-772.

Everett III, H. (1963). Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations research, 11*(3), 399-417.

Expósito-Izquierdo, C., González-Velarde, J. L., Melián-Batista, B., & Moreno-Vega, J. M. (2013). Hybrid estimation of distribution algorithm for the quay crane scheduling problem. *Applied Soft Computing, 13*(10), 4063-4076.

Feo, T. A., & Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization, 6*(2), 109-133.

Fisher, M. L. (1981). The Lagrangian relaxation method for solving integer programming problems. *Management science, 27*(1), 581-615.

Fisher, M. L., & Hochbaum, D. S. (1980). Database location in computer networks. *Journal of the ACM (JACM), 27*(4), 718-735.

Fisher, M. L., Jaikumar, R., & Van Wassenhove, L. N. (1986). A multiplier adjustment method for the generalized assignment problem. *Management science, 32*(9), 1095-1103.

Fortnow, L., & Homer, S. (2003). A short history of computational complexity. *Bulletin of the EATCS, 80*, 95-133.

Fraser, A. S. (1957). Simulation of Genetic Systems by Automatic Digital Computers I. Introduction. *Australian Journal of Biological Sciences, 10*(4), 484-491.

Froyland, G., Koch, T., Megow, N., Duane, E., & Wren, H. (2008). Optimizing the landside operation of a container terminal. *OR spectrum, 30*(1), 53-75.

Fu, Y.-M., Diabat, A., & Tsai, I.-T. (2014). A multi-vessel quay crane assignment and scheduling problem: Formulation and heuristic solution approach. *Expert Systems with Applications, 41*(15), 6959-6965.

Gadeyne, B., & Verhamme, P. (2011). *Optimizing Maritime Container Terminal Operations.* (Master of Business Economics Master Thesis), Ghent University.

Gambardella, L. M., Rizzoli, A. E., & Zaffalon, M. (1998). Simulation and planning of an intermodal container terminal. *Simulation, 71*(2), 107-116.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-Completeness*. San Francisco: W. H. Freeman.

Garfinkel, R. S., & Nemhauser, G. L. (1972). *Integer programming*. New York: Wiley.

GDV. (2012). Container Handbook: Cargo loss prevention information from German marine insurers. In *GDV, Berlin.* Retrieved 22 June, 2012, from http://www.containerhandbuch.de/chb_e/stra/index.html?/chb_e/stra/stra_01_03_03.html

Gendreau, M., & Potvin, J.-Y. (2005). Tabu search. In E. K. Burke & G. Kendall (Eds.), *Search methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* (pp. 165-186): Springer.

Geoffrion, A. M. (1974). Lagrangean relaxation for integer programming. *Approaches to Integer Programming*, 82-114.

Gilmore, P. C., & Gomory, R. E. (1963). A linear programming approach to the cutting stock problem-Part II. *Operations research*, *11*(6), 863-888.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, *8*(1), 156-166.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, *13*(5), 533-549.

Glover, F. (1989). Tabu search-part I. *ORSA Journal on computing*, *1*(3), 190-206.

Glover, F. (1990). Tabu search-part II. *ORSA Journal on computing*, *2*(1), 4-32.

Glover, F., & Laguna, M. (1997). *Tabu search*. Norwell, MA: Kluwer Academic Publishers.

Goffin, J.-L. (1977). On convergence rates of subgradient optimization methods. *Mathematical Programming*, *13*(1), 329-347.

Goffin, J.-L. (2012). Subgradient Optimization in Nonsmooth Optimization (including the Soviet Revolution). *Optimization Stories, Documenta Mathematica book series, Extra volume*, 277-290.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, *1989*

Goodchild, A. V. (2005). *Crane double cycling in container ports: algorithms, evaluation, and planning.* (PhD thesis), University of California at Berkeley, CA.

Goodchild, A. V., & Daganzo, C. F. (2004). *Reducing ship turn-around time using double-cycling*. Institue for Transportation Studies Research Report (UCB).Retrieved from http://escholarship.org/uc/item/86r4p6sc

Goodchild, A. V., & Daganzo, C. F. (2005a). *Crane double cycling in container ports: effect on ship dwell time*. Institue for Transportation Studies Research Report (UCB).Retrieved from http://escholarship.org/uc/item/9qp7p7jq

Goodchild, A. V., & Daganzo, C. F. (2005b). *Performance Comparison of Crane Double Cycling Strategies*: Institute of Transportation Studies, University of California, Berkeley.

Goodchild, A. V., & Daganzo, C. F. (2006). Double-cycling strategies for container ships and their effect on ship loading and unloading operations. *Transportation Science*, *40*(4), 473-483.

Goodchild, A. V., & Daganzo, C. F. (2007). Crane double cycling in container ports: planning methods and evaluation. *Transportation Research Part B: Methodological*, *41*(8), 875-891.

Gottwald Port Technology. (2012). Mobile Harbour Cranes. Retrieved 23 June, 2012, from http://www.gottwald.com/gottwald/site/gottwald/en/products/harbour-cranes/mobile-harbour-cranes.html

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics, 5*, 287-326.

Grötschel, M., Jünger, M., & Reinelt, G. (1984). A cutting plane algorithm for the linear ordering problem. *Operations research, 32*(6), 1195-1220.

Guan, Y., Yang, K.-H., & Zhou, Z. (2013). The crane scheduling problem: models and solution approaches. *Annals of Operations Research, 203*(1), 119-139.

Günther, H.-O., & Kim, K. H. (2006). Container terminals and terminal operations. *OR spectrum, 28*(4), 437-445.

Guo, P., Cheng, W., & Wang, Y. (2014). A modified generalized extremal optimization algorithm for the quay crane scheduling problem with interference constraints. *Engineering Optimization, 46*(10), 1411-1429.

Han, X.-l., Lu, Z.-q., & Xi, L.-f. (2010). A proactive approach for simultaneous berth and quay crane scheduling problem with stochastic arrival and handling time. *European Journal of Operational Research, 207*(3), 1327-1340.

Hapag-Lloyd. Container Specification. Retrieved 22 June, 2012, from http://www.hapag-lloyd.com/downloads/press_and_media/publications/Brochure_Container_Specification_en.pdf

Hayut, Y. (1980). Inland container terminal—function and rationale. *Maritime Policy and Management, 7*(4), 283-289.

He, X., Wang, S., & Zheng, J. (2011). *A hybrid heuristic algorithm for integrated large-capacity quay crane scheduling problem.* Paper presented at the Computer Research and Development (ICCRD), 2011 3rd International Conference on.

Held, M., & Karp, R. M. (1970). The traveling-salesman problem and minimum spanning trees. *Operations research, 18*(6), 1138-1162.

Held, M., & Karp, R. M. (1971). The traveling-salesman problem and minimum spanning trees: Part II. *Mathematical Programming, 1*(1), 6-25.

Held, M., Wolfe, P., & Crowder, H. P. (1974). Validation of subgradient optimization. *Mathematical Programming, 6*(1), 62-88.

Henesey, L. E. (2006). *Multi-agent systems for container terminal management.* (PhD thesis), Blekinge Institute of Technology, Karlskrona, Sweden.

Hertz, A., & de Werra, D. (1990). The tabu search metaheuristic: how we used it. *Annals of Mathematics and Artificial Intelligence, 1*(1), 111-121.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*: U Michigan Press.

Hornik, K., & Grün, B. (2007). TSP-Infrastructure for the traveling salesperson problem. *Journal of Statistical Software, 23*(2), 1-21.

Imai, A., Nagaiwa, K. I., & Tat, C. 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: Methodological, 35*(4), 401-417.

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

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: Methodological, 39*(3), 199-221.

Javanshir, H., & Seyedalizadeh Ganji, S. (2010). Yard crane scheduling in port container terminals using genetic algorithm. *Journal of Industrial Engineering International, 6*(11), 39-50.

Jin, Z., & Li, N. (2011). Optimization of Quay Crane Dynamic Scheduling Based on Berth Schedules in Container Terminal. *Journal of Transportation Systems Engineering and Information Technology, 11*(3), 58-64.

Johnson, D. S. (2012). A Brief History of NP-Completeness, 1954–2012. *Optimization Stories, Documenta Mathematica book series, Extra volume*, 359-376.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval research logistics quarterly, 1*(1), 61-68.

Karp, R. M. (1972). *Reducibility among combinatorial problems*. New York: Plenum Press.

Kaveshgar, N., & Huynh, N. (2015). Integrated quay crane and yard truck scheduling for unloading inbound containers. *International Journal of Production Economics, 159*, 168-177.

Kaveshgar, N., Huynh, N., & Rahimian, S. K. (2012). An efficient genetic algorithm for solving the quay crane scheduling problem. *Expert Systems with Applications, 39*(18), 13108-13117.

Kenndy, J., & Eberhart, R. (1995). *Particle swarm optimization.* Paper presented at the Proceedings of IEEE International Conference on Neural Networks.

Kim, K. H., Kang, J. S., & Ryu, K. R. (2004). A beam search algorithm for the load sequencing of outbound containers in port container terminals. *OR spectrum, 26*(1), 93-116.

Kim, K. H., & Kim, H. B. (1999). Segregating space allocation models for container inventories in port container terminals. *International Journal of Production Economics, 59*(1), 415-423.

Kim, K. H., & Kim, K. Y. (1999). Routing straddle carriers for the loading operation of containers using a beam search algorithm. *Computers and Industrial Engineering, 36*(1), 109-136.

Kim, K. H., & Moon, K. C. (2003). Berth scheduling by simulated annealing. *Transportation Research Part B: Methodological, 37*(6), 541-560.

Kim, K. H., & Park, K. T. (2003). A note on a dynamic space-allocation method for outbound containers. *European Journal of Operational Research, 148*(1), 92-101.

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

Kim, K. H., Park, Y.-M., & Ryu, K. R. (2000). Deriving decision rules to locate export containers in container yards. *European Journal of Operational Research, 124*(1), 89-101.

Kim, K. Y., & Kim, K. H. (1999). A routing algorithm for a single straddle carrier to load export containers onto a containership. *International Journal of Production Economics, 59*(1), 425-433.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science, 220*(4598), 671-680.

Kolisch, R., & Sprecher, A. (1997). PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. *European Journal of Operational Research, 96*(1), 205-216.

Kolisch, R., Sprecher, A., & Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management science*, 1693-1703.

Ku, D. (2009). *CPM Approach for the Analysis of Loading Plan in RoRo Terminal.* (Master thesis), Pukyong National University, Busan, Korea. Retrieved from http://dcollection.pknu.ac.kr/jsp/common/DcLoOrgPer.jsp?sItemId=000000017452

Ku, D. (2014). Rehandling Problem of Pickup Containers under Truck Appointment System. *Lecture Notes in Computer Science, 8733*, 272-281. 10.1007/978-3-319-11289-3_28

Ku, D., & Arthanari, T. S. (2014). On double cycling for container port productivity improvement. *Annals of Operations Research*, 1-16. 10.1007/s10479-014-1645-z

Lai, K., & Shih, K. (1992). A study of container berth allocation. *Journal of Advanced Transportation, 26*(1), 45-60.

Land, A. H., & Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, 497-520.

Le-Griffin, H. D., & Murphy, M. (2006). *Container terminal productivity: Experiences at the ports of Los Angeles and Long Beach*.

Lee, C.-Y., Liu, M., & Chu, C. (2014). Optimal Algorithm for the General Quay Crane Double-Cycling Problem. *Transportation Science*

Lee, D.-H., Cao, Z., & Meng, Q. (2007). Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm. *International Journal of Production Economics, 107*(1), 115-124.

Lee, D.-H., & Chen, J. H. (2010). An improved approach for quay crane scheduling with non-crossing constraints. *Engineering Optimization, 42*(1), 1-15.

Lee, D.-H., Chen, J. H., & Cao, J. X. (2011). Quay crane scheduling for an indented berth. *Engineering Optimization, 43*(9), 985-998.

Lee, D.-H., & Qiu Wang, H. (2010). Integrated discrete berth allocation and quay crane scheduling in port container terminals. *Engineering Optimization, 42*(8), 747-761.

Lee, D.-H., & Wang, H. Q. (2010). An approximation algorithm for quay crane scheduling with handling priority in port container terminals. *Engineering Optimization, 42*(12), 1151-1161.

Lee, D.-H., Wang, H. Q., & Miao, L. (2007). An approximation algorithm for quay crane scheduling with non-interference constraints in port container terminals. *Tristan VI, Phuket, June*, 10-15.

Lee, D.-H., Wang, H. Q., & Miao, L. (2008a). Quay crane scheduling with handling priority in port container terminals. *Engineering Optimization, 40*(2), 179-189.

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

Legato, P., Mazza, R. M., & Trunfio, R. (2008). *Simulation-based optimization for the quay crane scheduling problem.* Paper presented at the Simulation Conference, 2008. WSC 2008. Winter.

Legato, P., Mazza, R. M., & Trunfio, R. (2010). Simulation-based optimization for discharge/loading operations at a maritime container terminal. *OR spectrum, 32*(3), 543-567.

Legato, P., & Trunfio, R. (2014). A local branching-based algorithm for the quay crane scheduling problem under unidirectional schedules. *4OR, 12*(2), 123-156.

Legato, P., Trunfio, R., & Meisel, F. (2012). Modeling and solving rich quay crane scheduling problems. *Computers & Operations Research, 39*(9), 2063-2078.

Levin, L. A. (1973). Universal sequential search problems. *Problemy Peredachi Informatsii, 9*(3), 115-116.

Lieberman, R., & Turksen, I. (1981). Crane scheduling problems. *AIIE transactions, 13*(4), 304-311.

Lieberman, R., & Turksen, I. (1982). Two-operation crane scheduling problems. *IIE Transactions, 14*(3), 147-155.

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

Lim, A., Rodrigues, B., Xiao, F., & Zhu, Y. (2002). *Crane scheduling using tabu search.* Paper presented at the 14th IEEE International Conference on Tools with Artificial Intelligence, 2002 (ICTAI 2002), Washington, DC.

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

Lim, A., Rodrigues, B., & Xu, Z. (2004). Approximation Schemes for the Crane Scheduling Problem. In T. Hagerup & J. Katajainen (Eds.), *Algorithm Theory - SWAT 2004* (Vol. 3111, pp. 323-335): Springer Berlin / Heidelberg.

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

LinkedIn Discussion. (n.d.). Dual cycling at quay crane. Retrieved 25 June, 2012, from http://www.linkedin.com/groups/Dual-cycling-quay-crane-1947860.S.73716732

Little, J. D., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations research, 11*(6), 972-989.

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

Liu, M., Zheng, F., & Li, J. (2014). Scheduling small number of quay cranes with non-interference constraint. *Optimization Letters*, 1-10.

Løkketangen, A., & Glover, F. (1996). Probabilistic move selection in tabu search for zero-one mixed integer programming problems. In I. H. Osman & J. P. Kelly (Eds.), *Meta-Heuristics: Theory and Applications* (pp. 467-487). Norwell, MA: Kluwer Academics Publishers.

Lorie, J. H., & Savage, L. J. (1955). Three problems in rationing capital. *The Journal of Business, 28*(4), 229-239.

Lu, Z., Han, X., Xi, L., & Erera, A. L. (2012). A Heuristic for the Quay Crane Scheduling Problem Based on Contiguous Bay Crane Operations. *Computers & Operations Research*

Markowitz, H. M., & Manne, A. S. (1957). On the solution of discrete programming problems. *Econometrica: Journal of the Econometric Society*, 84-110.

Martello, S., & Vigo, D. (1998). Exact solution of the two-dimensional finite bin packing problem. *Management science*, 388-399.

Martin, R. K. (2012). *Large scale linear and integer optimization: a unified approach*: Springer Science & Business Media.

Meer, J. R. (2000). *Operational control of internal transport.* (PhD thesis), Erasmus University Rotterdam.

Meersmans, P. J. M., & Dekker, R. (2001). *Operations research supports container handling*. Erasmus School of Economics (ESE).

Meisel, F. (2009). *Seaside operations planning in container terminals*: Springer.

Meisel, F. (2011). The quay crane scheduling problem with time windows. *Naval Research Logistics (NRL), 58*(7), 619-636.

Meisel, F., & Bierwirth, C. (2011a). QCSPgen - A Benchmark Generator for Quay Crane Scheduling Problems. Retrieved 15 Jun, 2012, from http://prodlog.wiwi.uni-halle.de/qcspgen/

Meisel, F., & Bierwirth, C. (2011b). A unified approach for the evaluation of quay crane scheduling models and algorithms. *Computers & Operations Research, 38*(3), 683-693.

Meisel, F., & Bierwirth, C. (2013). A framework for integrated berth allocation and crane operations planning in seaport container terminals. *Transportation Science, 47*(2), 131-147.

Meisel, F., & Wichmann, M. (2010). Container sequencing for quay cranes with internal reshuffles. *OR spectrum, 32*(3), 569-591.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics, 21*(6), 1087-1092.

Michael, P. (1995). Scheduling, theory, algorithms, and systems. *Englewood Cli s, New Jersey*

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 (NRL), 53*(1), 45-59.

Monaco, M. F., & Sammarra, M. (2011). Quay crane scheduling with time windows, one-way and spatial constraints. *International Journal of Shipping and Transport Logistics, 3*(4), 454-474.

Murty, K. G. (2007). Yard crane pools and optimum layouts for storage yards of container terminals. *Journal of Industrial and Systems Engineering, 1*(3), 190-199.

Murty, K. G., Wan, Y., Liu, J., Tseng, M. M., Leung, E., Lai, K. K., & Chiu, H. W. C. (2005). Hongkong international terminals gains elastic capacity using a data-intensive decision-support system. *Interfaces*, 61-75.

Nam, H., & Lee, T. (2013). A scheduling problem for a novel container transport system: a case of mobile harbor operation schedule. *Flexible Services and Manufacturing Journal, 25*(4), 576-608.

Ng, W., & Mak, K. (2006). Quay crane scheduling in container terminals. *Engineering Optimization, 38*(6), 723-737.

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2013). Hybrid evolutionary computation methods for quay crane scheduling problems. *Computers & Operations Research, 40*(8), 2083-2093.

Nguyen, V. D., & Kim, K. H. (2009). A dispatching method for automated lifting vehicles in automated port container terminals. *Computers & Industrial Engineering, 56*(3), 1002-1020.

Nikjoofar, A., & Zarghami, M. (2013). 5 Water Distribution Networks Designing by the Multiobjective Genetic Algorithm and Game Theory. *Metaheuristics in Water, Geotechnical and Transport Engineering*, 99-119.

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.

Nowicki, E., & Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Management science, 42*(6), 797-813.

Padberg, M., & Rinaldi, G. (1987). Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Operations Research Letters, 6*(1), 1-7.

Padberg, M., & Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review, 33*(1), 60-100.

Papadimitriou, C. H., & Steiglitz, K. (1982). *Combinatorial optimization : algorithms and complexity*. Englewood Cliffs, N.J.: Prentice Hall.

Park, K. T., & Kim, K. H. (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.-M., & Kim, K. H. (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.

Rani, D., Jain, S. K., Srivastava, D. K., & Perumal, M. (2012). 3 Genetic Algorithms and Their Applications to Water Resources Systems. *Metaheuristics in Water, Geotechnical and Transport Engineering*, 43.

Rebollo, M., Julian, V., Carrascosa, C., & Botti, V. (2000). *A multi-agent system for the automation of a port container terminal*.

Reinelt, G. (1991). TSPLIB - A traveling salesman problem library. *ORSA Journal on computing, 3*(4), 376-384.

Ryu, K. R., Kim, K. H., Lee, Y. H., & Park, Y.-M. (2001). *Load sequencing algorithms for container ships by using metaheuristics.* Paper presented at the Proceedings of 16th International Conference on Production Research (CD-ROM), Prague, Czech Republic.

Sahni, S., & Gonzalez, T. (1976). P-complete approximation problems. *Journal of the ACM (JACM), 23*(3), 555-565.

Salhi, S. (2002). Defining tabu list size and aspiration criterion within tabu search methods. *Computers & Operations Research, 29*(1), 67-86.

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

Sherali, H. D., & Myers, D. C. (1988). Dual formulations and subgradient optimization strategies for linear programming relaxations of mixed-integer programs. *Discrete Applied Mathematics, 20*(1), 51-68.

Shor, N. (1962). An application of the method of gradient descent to the solution of the network transportation problem. *Materialy Naucnovo Seminara po Teoret i Priklad. Voprosam Kibernet. i Issted. Operacii, Nucnyi Sov. po Kibernet, Akad. Nauk Ukrain. SSSR, vyp, 1*, 9-17.

Shor, N. (1964). On the structure of algorithms for numerical solution of problems of optimal planning and design. *Diss. Doctor Philos. Kiev*

Shor, N. Z. (1985). *Minimization methods for non-differentiable functions* (Vol. 3): Springer.

Sidney, J. B. (1979). The two-machine maximum flow time problem with series parallel precedence relations. *Operations research, 27*(4), 782-791.

Sipser, M. (2006). *Introduction to the theory of computation* (2nd ed.). Boston: Thomson Course Technology.

Song, J.-H. (2007). A study for optimization of double cycling in container ports. *Port Technology International, Edition 36*, 50-52. Retrieved from http://www.porttechnology.org/technical_papers/a_study_for_optimisation_of_double_cycling_in_container_ports/

Song, L., Cherrett, T., & Guan, W. (2012). Study on berth planning problem in a container seaport: Using an integrated programming approach. *Computers & Industrial Engineering, 62*(1), 119-128.

Stahlbock, R., & Voß, S. (2008). Operations research at container terminals: a literature update. *OR spectrum, 30*(1), 1-52.

Steenken, D., Voß, S., & Stahlbock, R. (2004). Container terminal operation and operations research-a classification and literature review. *OR spectrum, 26*(1), 3-49.

Steenken, D., Winter, T., & Zimmermann, U. T. (2001). Stowage and transport optimization in ship planning. *Online optimization of large scale systems*, 731-745.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*(2), 278-285.

Taillard, E. D. (1994). Parallel taboo search techniques for the job shop scheduling problem. *ORSA Journal on computing, 6*(2), 108-117.

Taleb-Ibrahimi, M., De Castilho, B., & Daganzo, C. F. (1993). Storage space vs handling work in container terminals. *Transportation Research Part B: Methodological, 27*(1), 13-32.

Tang, L., Zhao, J., & Liu, J. (2014). Modeling and solution of the joint quay crane and truck scheduling problem. *European Journal of Operational Research, 236*(3), 978-990.

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.

Theodorou, E., & Diabat, A. (2014). A joint quay crane assignment and scheduling problem: formulation, solution algorithm and computational results. *Optimization Letters*, 1-19.

Thurston, T., & Hu, H. (2002). *Distributed agent architecture for port automation.* Paper presented at the Computer Software and Applications Conference, 2002.

UNCTAD/RMT. (2014). *Review of Maritime Transport*. New York, Geneva: United Nations.Retrieved from http://unctad.org/en/PublicationsLibrary/rmt2014_en.pdf

Unsal, O., & Oguz, C. (2013). Constraint programming approach to quay crane scheduling problem. *Transportation Research Part E: Logistics and Transportation Review, 59*, 108-122.

Van Laarhoven, P. J., Aarts, E. H., & Lenstra, J. K. (1992). Job shop scheduling by simulated annealing. *Operations research, 40*(1), 113-125.

Vazirani, V. V. (2001). *Approximation algorithms*: Springer Science & Business Media.

Vis, I. F., & Harika, I. (2004). Comparison of vehicle types at an automated container terminal. *OR spectrum, 26*(1), 117-143.

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

Wang, D., Li, X., & Wang, Q. (2011). *A two-stage composite heuristic for dual cycling quay crane scheduling problem.* Paper presented at the Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on.

Wang, S., & Hu, W. (2009). *Multi quay crane scheduling problem based on ACO in container terminals.* Paper presented at the Management and Service Science, 2009. MASS'09. International Conference on.

Wang, S., Zheng, J., Zheng, K., Guo, J., & Liu, X. (2012). Multi Resource Scheduling Problem Based on an Improved Discrete Particle Swarm Optimization. *Physics Procedia, 25*, 576-582.

Wang, Y., & Kim, K. H. (2011). A quay crane scheduling algorithm considering the workload of yard cranes in a container yard. *Journal of Intelligent Manufacturing, 22*(3), 459-470.

Wikipedia. (n.d.). List of largest container ships. Retrieved 4 April, 2015, from http://en.wikipedia.org/wiki/List_of_largest_container_ships

Yang, C. H., Choi, Y. S., & Ha, T. Y. (2004). Simulation-based performance evaluation of transport vehicles at automated container terminals. *OR spectrum, 26*(2), 149-170.

Yin, R. K. (2014). *Case study research: Design and methods* (5th ed.). LA: Sage publications.

Zhang, C., Liu, J., Wan, Y., Murty, K. G., & Linn, R. J. (2003). Storage space allocation in container terminals. *Transportation Research Part B: Methodological, 37*(10), 883-903.

Zhang, H., & Kim, K. H. (2009). Maximizing the number of dual-cycle operations of quay cranes in container terminals. *Computers & Industrial Engineering, 56*(3), 979-992.

Zhao, W., & Goodchild, A. V. (2010). The impact of truck arrival information on container terminal rehandling. *Transportation Research Part E: Logistics and Transportation Review, 46*(3), 327-343.

Zhu, Y., & Lim, A. (2004). *Crane Scheduling with Spatial Constraints: Mathematical Models and Solving Approaches.* Paper presented at the AMAI.

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