Models and solution algorithms for container terminal operations

Shi, Yanfeng

2018

Shi, Y. (2018). Models and solution algorithms for container terminal operations. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/82590>

<https://doi.org/10.32657/10220/47515>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**Models and solution algorithms for
container terminal operations**

SHI YANFENG

**SCHOOL OF CIVIL AND ENVIRONMENTAL
ENGINEERING**

2018

Models and solution algorithms for container terminal operations

SHI YANFENG

School of Civil and Environmental Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2018

Abstract

Containerization technique greatly improves the efficiency of international cargo shipping. Container terminals, as the service gateways of hinterland transport and maritime transport, play a crucial role in the process of cargo transport. In presence of growing demand for containerized marine shipping and increasing sizes of container vessels, a swift service at container terminal is vital. To improve service productivity of a container terminal, one option is to increase the capacity of container terminal or purchase new facility. However, the obvious drawback of this option is the high investment cost involved, which discourages terminal operators from taking this option. In contrast, a more viable option is taking advantage of planning tools such as the planning of quay side operations or land side operations, which will not incur a significant amount of investment. In this thesis, we focus on developing more realistic models and efficient solution algorithms for these types of problems in optimizing container terminal.

Integer programming is widely used in many application areas and plays a fundamental role in supporting managerial decisions or modeling hard to solve problem. Most of ocean shipping problems can be modeled as integer programming or mixed integer programming, such as cargo routing, fleet deployment, crane scheduling, stowage planning, etc. Different from linear programs which is solvable in polynomial time, integer programming problems are normally NP-hard. Many solution methods have been proposed by the previous studies, one of the well-known and successfully applied methodology is Benders decomposition. This method was mostly used for LP problems or MIP problems.

In these problems, the subproblem is a linear program for which the strong dual holds and valid Benders cut can be derived using dual theory. However, when the subproblem is integer problem, the strong dual does not hold and hence the cuts cannot be generated accordingly. To overcome this, this thesis proposes a Benders decomposition approach for generic integer programming problem. We first formulate a relaxed cut generation problem to elicit a Benders cut when the subproblem has integer constraints. If the obtained Benders cut can not cut off infeasible solution, two fixed integer problem are invoked to add solution elimination constraint. This guarantees the convergence of our proposed approach.

In containerized ocean shipping, one of the prominent operations is loading and unloading during calling a sequence of ports according to a planned route. Positions of containers within container ship, known as stowage plan, greatly affect the operation efficiency in container terminal. Poor stowage plan would cause a large number of container shifts at a container terminal. This container shifts operations are unproductive, costly and time consuming transportation activities for both shipping company and terminal operators. However, arranging containers into proper locations within container ship is a very complicated task.

Currently, a successfully used approach to solve stowage planning problem is to decompose the problem into master planning phase and slot planning phase. Our study focuses on container stowage slot planning phase. We first extend the container stowage slot planning problem by including the constraints that the weight of containers stowed in the same stack of a container vessel does not increases from the bottom of the stack to the top of the stack. Indeed, this constraint is practically required in real loading operations but was not fulfilled in the existing model formulations. Then, we investigate container stowage slot planning problem from a multi-port view. A multi-port view of slot planning is essential in reducing the negative interactive effect of stowing containers in

the subsequent port of call. A nonlinear integer programming model is developed for this problem, which is flexible to incorporate realistic stacking rules in comparison with those in previous studies.

Another challenging problem in container terminal is quay crane scheduling problem. Quay crane productivity has always been one of the critical components of terminal productivity, optimal scheduling quay cranes can increase their efficiency substantially. This problem aims to assign the loading and unloading tasks of a container ship to the particular quay crane and determine the schedule for the quay crane to process the loading and discharging tasks.

To improve the efficiency of quay cranes, another strategy is adopting double cycling technique, which enables quay cranes to carry out unloading after loading activity or loading after unloading activity in each cycle. This indeed converts the idle moves of quay cranes with only single cycling into productive moves. In the thesis, we modeled the quay crane scheduling problem with incorporation of double cycling strategy. Most of existing studies focused on double cycling with single QC, i.e., only one single quay crane performs loading and unloading simultaneously on a given bay of container ship. In this case, the interaction among quay cranes are not considered and the efficiency of double cycling is underestimated. In our study, we model the quay crane scheduling problem while considering double cycling strategy, in this case crane interference, safety margin between different QCs, precedence relationship among tasks and non-simultaneous handling among tasks all considered in a much more realistic and complicated problem formulation. As the problem is hard to solve, we develop a logic based Benders decomposition method for the problem. The logic based decomposition method in our paper can guarantee global optimal solution, meanwhile reduce computation time substantially.

Acknowledgements

Foremost, I would like to express my sincere thanks of gratitude to my supervisor Professor Wang Zhiwei. He offered me valuable and constructive suggestions in my research work. He has taught me more than I could ever give him credit for here. I believe without his endless help and support, I am not able to finish this journey.

My special thanks are extended to all the committee members of my Qualification Examination. Each of them has provided me extensive personal and professional guidance in my research. I learned a lot from their insightful comments.

I am also grateful to all of my friends with whom I have had the pleasure in these years. We played together, laughed together and studied together. I cherish these moments a lot. Thanks for their concerns and help in my life.

Nobody has been more important to me than my family and girlfriends in my pursuit of PhD journey. I would like to thank my parents, their love are with me no matter what I pursue. Most importantly, I am very grateful to my girlfriends, who provides me unending support. Because all of you, my world has become beautiful.

Contents

ABSTRACT	i
ACKNOWLEDGEMENTS	iv
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF ABBREVIATIONS	xi
CHAPTER 1 INTRODUCTION	1
1.1 Background	1
1.2 Research scope and objectives	2
1.3 Outline of this thesis	3
CHAPTER 2 LITERATURE REVIEW	4
2.1 Container stowage planning problem	4
2.2 Quay crane scheduling problem	8
2.3 Solution methods for integer programming problems	11
2.4 Concluding remarks	17
CHAPTER 3 SOLVING INTEGER LINEAR PROGRAMMING PROBLEMS USING BENDERS DECOMPOSITION METHOD	19
3.1 Introduction	19
3.2 Generic integer linear programming and Benders decomposition ..	21
3.2.1 Fixed subproblem and its dual	24
3.2.2 Complementary slackness condition	26
3.2.3 Relaxed cut generation problem	26
3.3 Benders Decomposition Applied to Integer Programs	29
3.3.1 Integer Benders Decomposition Algorithm	29
3.4 Computational Experiments	30

3.4.1	Test instances	31
3.4.2	Computational results	32
3.5	Conclusion	34
CHAPTER 4 MODELING AND SOLVING MULTI-PORT CONTAINER STOWAGE SLOT PLANNING PROBLEM		36
4.1	Introduction	36
4.2	Problem description and mathematical formulation	39
4.2.1	Integer linear programming model	43
4.3	Solution algorithms	45
4.3.1	Definition, sets and parameters	45
4.3.2	Slot attributes based heuristic method	48
4.3.3	Tabu search algorithm	51
4.4	Numerical experiments	54
4.4.1	Analysis of results	56
4.5	Conclusion	62
CHAPTER 5 MODELING AND SOLVING QUAY CRANE SCHEDULING PROBLEM CONSIDERING DOUBLE CYCLING		64
5.1	Introduction	64
5.2	Notation and model formulation	67
5.2.1	Model formulation	68
5.2.2	Graph representation of feasible solution	72
5.3	An enhanced combinatorial Benders cuts method	73
5.3.1	Idea and outline	73
5.3.2	Lower bound problem	75
5.3.2.1	Infeasible cuts for the Lower bound problem	76
5.3.3	Upper bound problem	78
5.3.3.1	Non-optimal cuts for the Lower bound problem	79
5.4	Heuristic solution method	81
5.4.1	Assigning and sequencing tasks to quay crane	81
5.4.2	Scheduling of quay crane tasks	82
5.5	Numerical examples	85
5.5.1	Evaluate the performance of LBD and TIH	86

5.5.2 Evaluate the efficiency of QCSP considering double cycling strategies	88
5.6 Conclusion	90
CHAPTER 6 CONCLUSIONS	92
6.1 Thesis summary	92
6.2 Future research	94
REFERENCES	95

List of Figures

2.1	Dimensions of the commonly used containers.	5
2.2	The basic structure of a container ship	6
4.1	The master planning phase and slot planning phase of stowage planning	38
4.2	Optimal slot plan for the example problem upon leaving each port of call, wherein slot highlighted with dark colour denotes container at that slot was shifted at port 3, the maximum weight limits for each stack is 100, POD denotes the port of destination	45
4.3	Comparison of the performance of SAM and TSA for experiment 1	61
4.4	Comparison of the performance of SAM and TSA for experiment 2	61
4.5	Comparison of the performance of SAM and TSA for experiment 3	62
5.1	Feasible work areas for quay cranes with safety margin set to 1 bay	69
5.2	A graph representation of feasible solution of the model	72
5.3	procedure for logic based decomposition method	74
5.4	type of infeasible solution obtained by lower bound problem: (a) sub-cycle with no precedence violation; (b) sub-cycle with prece- dence violation within a quay crane task processing sequence; (c) sub-cycle with precedence violation among different quay cranes .	76
5.5	Example of disjunctive graph	83
5.6	Flow chart for TIH method	85

List of Tables

3.1	The sizes of the test problems	32
3.2	Computational results for problems p1--p8	32
3.3	Computational results for problems p9--p15	33
3.4	Computational results for problems p16--p20	33
4.1	Sets, parameters and variables in the model	41
4.2	Transportation matrix for the port call	44
4.3	Index set and constants in the solution method	46
4.4	Characteristic of experiment 1 instances	54
4.5	Characteristic of Experiment 2 instances	55
4.6	Characteristic of Experiment 3 instances	55
4.7	Computation results on instances of experiment 1, ``S" represents the container weight scenarios	57
4.8	Computation results on instances of experiment 2, ``--" means no corresponding value were obtained, ``S" represents the container weight scenarios	58
4.9	Computation results on instances of experiment 3, ``--" means no corresponding value were obtained, ``S" represents the container weight scenarios	59
5.1	Sets, parameters and variables in the MIP model	67
5.2	Parameter settings for numerical study	86
5.3	Results obtained by CPLEX, LBD and TIH	87

5.4 The comparison of efficiency for each quay crane handling method, where $\tau = 1$, $a = 2\tau$ and $\bar{t} = 2\tau$, r =number of loading tasks/all the tasks	89
---	----

List of Abbreviations

B&B	Branch and bound
BAP	Berth allocation problem
CBC	Combinatorial Benders cut
CSP	Container stowage planning
GA	Genetic algorithm
GRASP	Greedy randomized adaptive search procedure
ILP	Integer linear programming
IP	Integer programming
LP	Linear programming
MIP	Mixed integer programming
MILP	Mixed integer linear programming
MINP	Mixed integer nonlinear programming
PSO	particle swarm optimization
qUFL	Quadratic Uncapacitated facility location
QCAP	Quay crane assignment problem
QCSP	Quay crane scheduling problem
QC	Quay crane
QCs	Quay cranes
TEU	Twenty foot equivalent unit
TSA	Tabu search algorithm

UDS	unidirectional scheduling
UFL	Uncapacitated facility location
VRP	Vehicle routing problem

Chapter 1

INTRODUCTION

1.1 Background

Modern shipping techniques play a vital role in globalized trade and are the backbone of the world economy. They connect new suppliers and consumers thousands of kilometers away from each other, and thus have transformed business models around the world. In this transformation, the introduction of containerization is an indispensable factor. Containerization, served as a standard interface between the different parties involved in the intermodal freight transport chain, facilitates the development of an integrated transport system.

Global container trade in 2017 had an annual growth of 6.1% and was estimated at 138.5 million of TEUs (Twenty-Foot Equivalent Units) , it is expected by BIMCO ¹ that demand will grow by 4%-4.5% in 2018 ². In addition, vessel size for transporting containers have increased to nearly 21413 TEUs vessels, the container ship still have the mega-trends ³. Besides, container ship operating costs are expected to rise in 2018.

The spectacular growth of demand for containerized marine shipping and notable increase of container vessels sizes have created many problems for ports

¹Baltic and International Maritime Council

²<http://www.marinepoland.com/news/trends-and-challenges-for-the-global-container-market-in-2018.html>

³<https://www.oocl.com/eng/pressandmedia/pressreleases/2017/Pages/12may17.aspx>

leading to higher requirements on operation efficiency in container terminals. It is observed that many container terminals are reaching their capacity limits leading to traffic and port congestion. The port congestion has a lot of negative effect on the major stakeholders, it can cause ship delays at port, extra manpower in container terminal, longer leader time for shipper, etc. To resolve this problem, one way is to increase port capacity. Capacity can be increased either by physical expansion or improved utilization of the available resources. However, the obvious drawback of this way are the high investment cost involved and the availability of lands, which discourages terminal operators from taking this option. In contrast, a more viable option is taking advantage of planning tools such as the planning of quay side operations or land side operations, which will not incur a significant amount of investment.

1.2 Research scope and objectives

The objective of this thesis research are two main parts: firstly, to develop more realistic mathematical programming models for quayside operations in container terminal; secondly, to propose efficient solution method for the considered problem.

More specifically, the following research tasks are conducted:

1. To provide a Benders decomposition approach for solving generic integer programming problems
2. To develop mathematical models for container stowage slot planning problem with more realistic constraints and propose efficient solution algorithm to solve the corresponding problems
3. To develop a mathematical model for quay crane scheduling problem consider double cycling and investigate its efficiency, meanwhile propose solution algorithm to solve this problem efficient.

1.3 Outline of this thesis

The thesis consists of six chapters. Chapter 1 presents a brief introduction to the background and motivations of this study. The research objective are also given in this chapter.

Chapter 2 reviews previous work on container stowage planning problem and quay crane scheduling problem, the solution methods for integer programming problem are also briefly reviewed, and make a summarization about the development of corresponding problem.

In Chapter 3, we present a Benders decomposition method for a generic integer linear programming problem, by introduce auxiliary variables into master problem, we formulate the subproblem as feasibility problem, a relaxed cut generation problem is formulated based on this. Meanwhile, we also used two fixed integer problems to add solution elimination constraint when the Benders cut obtained by the relaxed cut generation problem is not valid. This guarantee the convergence of the proposed method.

Chapter 4 studies the container stowage slot planning problem from a multi-port view. A multi-port view of slot planning is essential in reducing the negative effect of stowing containers in the subsequent port of call. Due to the NP-hard property of the problem, we propose two specific solution algorithms to solve it, i.e., a slot attributes based heuristic method and Tabu search method.

Chapter 5 investigates the efficiency of quay crane scheduling problem considering double cycling. Because of the NP-hard property of the problem, we used a logic based decomposition method to solve it.

In Chapter 6, we present a conclusion on the research of this thesis. In this conclusion, we discuss the findings and contributions of this research. Meanwhile, the recommendations for future work are presented.

Chapter 2

LITERATURE REVIEW

Container stowage planning problem and quay crane scheduling problem, as the typical problems in container shipping, received increasing research attention in the past few years. Because of the combinatorial nature in these problems, most of the models in the existing literatures are formulated as integer program which are very difficult to solve, especially when the problem size increases. Besides, the solution methods for these problems play a central roles for the application of the models.

Therefore, in this chapter, we review the literature on container stowage planning problem, quay crane scheduling problem and the development of solution methods for integer programming. Section 2.1 focuses on the container stowage planning problem. In section 2.2, the reviews of the QCSP and their related solution methods are presented. In section 2.3, the solution methods for integer programming are briefly reviewed. Section 2.4 presents a summary of this chapter.

2.1 Container stowage planning problem

Container stowage plan is the plan and method by which different types of container vessels are loaded with containers of specific standard sizes or types. The

most common containers are 20ft or 40ft long, and 8ft wide (see Figure 2.1). They are either 8'6" high called standard container or 9'6" high called high-cube container. Standard 20ft and 40ft containers, together with 40ft high-cube containers accounts for almost 90% of the world's container inventory in circulation (Delgado, 2013). Besides, containers can also be distinguished as reefer containers, hazardous containers, etc.

Figure 2.1: Dimensions of the commonly used containers.

Container vessel where containers are stowed is longitudinally divided into sections called bays (See Figure 2.2). Bays are divided in two sections, on deck and below deck, by a number of flat leak-proof structures called hatch covers. Containers are stowed on top of or below the hatch cover. A bay is further divided transversely into stacks that are one container wide holding a Twenty-foot Equivalent Unit container or a Forty-foot Equivalent Unit container, a bay is vertically indexed by tiers. Bay, stack and tier uniquely define a slot within container ship.

Figure 2.2: The basic structure of a container ship

Container stowage plan is important for both container terminals and liner shipping company, as good stowage plan can reduce container shifts at each port. Container shifts are unproductive, costly and time consuming transportation activities. To date, due to the importance of container stowage planning problem, many research have been conducted in the past few years. The complexity of container stowage planning problem was investigated by [Avriel et al. \(2000\)](#) and [Tierney et al. \(2014\)](#). [Avriel et al. \(2000\)](#) proved that the container ship stowage problem is NP-complete and demonstrated a relation of the stowage problem to the coloring of circle graphs problem. [Tierney et al. \(2014\)](#) showed that the complexity of the container ship stowage problem considered in [Avriel et al. \(2000\)](#) is polynomial if changing uncapacitated stacks of the problem to capacitated. The existing studies can be roughly classified into two categories in terms of solution approaches: single phase approaches and multi-phase approaches.

The single phase approaches treat the stowage planning problem as one entire problem, solving ship stability and slot planning simultaneously. In this category, [Avriel & Penn \(1993\)](#) proposed an exact binary linear programming model for multi-port single rectangular bay stowage planning, they focused on minimizing the total shifts of containers during the port rotations. However, their model was inflexible to incorporate practical aspects like stability and weight constraints. [Avriel et al. \(1998\)](#) developed a suspensory heuristic procedure to solve large instances of container stowage planning problem. [Dubrovsky et al. \(2002\)](#) considered a similar problem described by [Avriel & Penn \(1993\)](#) with

including stability constraints. To solve the problem fastly, they developed a genetic algorithm with an efficient encoding scheme. [Ambrosino et al. \(2004\)](#) presented a model meant to stow a complete vessel. They focused on minimizing the stowage time at the first port. The limitation is that containers were only loaded at the first port and unloaded only for the subsequent port rotations, there are no containers to be loaded on the container ship for the subsequent port of calls. [Ambrosino et al. \(2010\)](#) presented a constructive loading heuristic method and an ant colony optimization algorithm for stowing a complete vessel. [Sciomachen & Tanfani \(2003\)](#) and [Li et al. \(2008\)](#) considered the same problem described by [Ambrosino et al. \(2004\)](#). [Sciomachen & Tanfani \(2003\)](#) used a heuristic method based on the 3D bin packing problem for solving the problem. Containers with different type and size were considered in their method like reefer container, high cube container and open top container etc. [Li et al. \(2008\)](#) presented a binary linear model seeking to maximize space utilization while minimize the operative cost for multi-port master bay plan. However, their model can not truly represent the multi-port stowage planning problem as they did not consider the relation of stowage plan between each port. [Ding & Chou \(2015\)](#) studied stowage planning problem for a single rectangular bar, they proposed a heuristic algorithm to solve the problem. However, weight constraints were not considered. [Araújo et al. \(2016\)](#) developed a Pareto clustering search method to solve 3D container ship loading plan problem. In their problem setting, the containers were treated with the same size and weight.

In the category of multi-phase approaches, slot planning problem is solved after master planning phase for complete vessel stowage planning. [Wilson & Roach \(2000\)](#) and [Wilson et al. \(2001\)](#) described the decomposition approach to solve stowage planning problem, they proposed a branch and bound search method for master planning phase and a tabu search method for slot planning phase. [Kang & Kim \(2002\)](#) presented a heuristic method for the master planning phase called group to bay subproblem and a tree search method for slot planning

phase named containers to slots subproblem. [Pacino et al. \(2011\)](#) and [Delgado et al. \(2012b\)](#) also exploited the same decomposition approach. In [Pacino et al. \(2011\)](#), an integer programming model for multi-port master planning was proposed taking into account detailed constraints like containers with different type and weight, shear force and hatch over-stowage, etc., the results of the model can be used to generate an input for slot planning problem presented in [Delgado et al. \(2012b\)](#). [Francisco et al. \(2016\)](#) explicitly considered the separations rules for dangerous cargo and handling of rolled out containers when making the container stowage slot planning. To solve the problem efficiently, they developed a greedy randomized adaptive search procedure. However, the weight constraints were not included in the model (e.g. a heavier container cannot be stacked on a lighter one were not considered). [Christensen & Pacino \(2017\)](#) investigated a cargo mix problem which can be seen as master planning phase problem. The problem aims at selecting the amount of containers to load on a vessel so as to maximizes the value of the containers loaded.

2.2 Quay crane scheduling problem

The quayside operations at a container terminal are vital, among which QCSP is a core task. One can refer to [Bierwirth & Meisel \(2010\)](#) and [Bierwirth & Meisel \(2015\)](#) for a complete view of quayside operations review. In this section, we focus on the review of QCSP. In QCSP, discharging and loading containers of a container ship are scheduled on a set of deployed quay cranes to minimize the make-span of container vessel.

As one of the most challenging topics in container terminals, QCSP is widely studied because of its complexity and practical applicability. The problem of scheduling quay cranes received increasing research attention after the seminal work by [Daganzo \(1989\)](#), who presented a MIP formulation for the QCSP to minimize the weighted delay cost of vessels. Tasks are referred to as container

groups of a bay and the processing time is uniform in their work. [Peterkofsky & Daganzo \(1990\)](#) later developed a B&B method which can be used to solve large instances of QCSP.

[Kim & Park \(2004\)](#) proposed a MIP model for QCSP. In the formulation, they considered the quay crane interference and precedence relations among tasks as well as non-simultaneity constraint between tasks, wherein the tasks are referred to as container groups. A branch and bound method and a GRASP were proposed to solve the problem. [Ng & Mak \(2006\)](#) proposed a heuristic method to solve quay crane scheduling problem. In their method, the cranes are assigned to a set of non-overlapping zones, which reduce the difficult of the mult-crane scheduling problem. Their computational results showed the method is able to find effective solutions. [Moccia et al. \(2006\)](#) formulated the quay cane scheduling problem as VRP with side constraints. Their model can be solved efficiently by CPLEX for small instance problem. As for large instances, they developed a branch and cut algorithm. In [Sammarra et al. \(2007\)](#), QCSP was decomposed into routing problem and scheduling problem, they used tabu search method to solve the routing problem, and used a local search for scheduling problem. The algorithm proposed outperformed the GRASP (Kim and Park 2004). [Lee et al. \(2008\)](#) studied a QCSP in which the task is defined on the basis of hold, they proved the considered QCSP is NP-complete in nature, and proposed a genetic algorithm to obtain sub-optimal solution. [Beens & Ursavas \(2016\)](#) investigated a QCSP at indented berth structure in a container terminal and solved the problem using the branch and price method.

Although crane interference constraint are formulated by [Kim & Park \(2004\)](#) and [Sammarra et al. \(2007\)](#), the treatment of crane interference constraint was not correct in their model formulation. To make a correction, [Bierwirth & Meisel \(2009\)](#) introduced the minimum time span for two different quay cranes into the model formulation, and presented a revised MIP model for QCSP. The authors proposed a unidirectional scheduling (UDS) heuristic to search the solution, and

the results show that UDS performs better than the algorithms proposed by [Kim & Park \(2004\)](#) and [Sammorra et al. \(2007\)](#). UDS can obtain better solution with shorter time. [Al-Dhaheri & Diabat \(2015\)](#) developed a novel yet simple formulation for the QCSP. The main novelty of the formulation lies in the objective function, which is to minimize the relative difference of the workload between bays. However, the computations show that the model is ideal for small and medium sized problem, an exponential CPU time is required with the increase of the problem size. [Al-Dhaheri et al. \(2016a\)](#) studied a QCSP in which the uncertainty of quay crane handling time, straddle carrier round-trip time and container pick up time is considered, they proposed a stochastic MIP model and a simulation-based GA for the considered problem. Considering the importance of vessel stability when scheduling discharge and load operations, [Al-Dhaheri et al. \(2016a\)](#) incorporated the constraints into the model formulation, and a GA is proposed for the problem. [Tavakkoli-Moghaddam et al. \(2009\)](#) presented a MIP model for QCSAP. As the NP-hard property of the problem, they used genetic algorithm to solve real world situation.

For QC double cycling, [Goodchild & Daganzo \(2006\)](#) took the first step to study the double cycling problem mathematically. They formulated QC double cycling problem as a two machine flow shop problem and showed that double cycling can significantly increase crane productivity, about 20% cycles and 10% operational time can be reduced. [Goodchild & Daganzo \(2007\)](#) studied the longer term impact of double cycling on port operations including crane, vessel, and berth productivity, as well as impact on land side transportation operations. They developed a formula to predict the impact of quay crane double cycling on turn-around time. [Zhang & Kim \(2009\)](#) extended a single crane double cycling model to the case of multiple hatches with the objective of maximizing the number of dual cycles, and proposed a hybrid heuristic approach to solve the model. [Lee et al. \(2015\)](#) investigated the computational complexity for QC

double cycling problem. They proved that the problem can be solved polynomially by formulating it as a flow shop scheduling problem with series-parallel precedence constraints. [Liu et al. \(2015\)](#) studied QC double cycling considering internal-reshuffles in a bay, the internal-reshuffling operations help improve quay crane efficiency and decrease ship turn-around time at port. [Ku & Arthanari \(2016\)](#) pointed out that there is a flaw with the QC double cycling model by [Zhang & Kim \(2009\)](#).

Another representative studies of QCSP are to scheduling quay cranes with consideration of scheduling yard truck and yard crane simultaneously. [He et al. \(2015\)](#) studied this types of problem, they also incorporated the energy consumption of transporting tasks into their model. A hybrid of GA and PSO algorithm to solve the problem. The efficiency of the algorithm is illustrated through a computational experiment. [Al-Dhaheri et al. \(2016b\)](#) considered a uncertain issues in scheduling quay crane, the uncertainty stems from container handling process. They also considered the transfer operations between the quay and the stacking yard. As the problem is difficult to solve, they used a simulation-based GA to solve it.

2.3 Solution methods for integer programming problems

An integer programming problem means that some of the variables or all the variables in the problem are restrict to integer values. When a programming problem includes both integer variables and continuous variables, we call it

mixed integer program. This types of problems have the following general form,

$$\begin{aligned}
& \min \quad \mathbf{c}\mathbf{x} \\
& \text{s.t.} \quad \mathbf{A}\mathbf{x} \geq \mathbf{b} \\
& \quad \quad \mathbf{B}\mathbf{x} \geq \mathbf{d} \\
& \quad \quad \mathbf{x}_j \in \mathbb{Z} \\
& \quad \quad j \in N_1 \subseteq N
\end{aligned}$$

in which $\mathbf{A}$ and $\mathbf{B}$ denotes matrix with conformable dimensions. $\mathbf{c}$, $\mathbf{b}$ and $\mathbf{d}$ are given vectors. $\mathbf{x}$ is vector variable. When $N_1 = N$, it is a pure integer program. Otherwise, it is a mixed integer program.

In practice, integer programming is used in many applications areas, such as production planning, routing, scheduling and set covering, etc. Different from linear programs which are relatively easy to solve, integer programming problems are normally NP-hard. Although integer programming is hard to solve, in some cases relaxing the integer restriction does not affect the solution to be integer. This special case has the form ([Hoffman & Kruskal, 2010](#)): $\max \mathbf{c}^T \mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ have all integer entries and $\mathbf{A}$ is totally unimodular.

In general, the solution methods for integer programming can be classified into three main categories: (1) Exact algorithms. This types of solution algorithms guarantee to find an optimal solution. But they may take an exponential number of iterations. Cutting-planes and branch-and-bound are such algorithms. (3) Approximation algorithms. These solution algorithms can provide a sub-optimal solution together in polynomial time. Meanwhile, they also provide a bound on the degree of sub-optimality. (2) Heuristic algorithms. They also provide a suboptimal solution, but the solution quality is not guaranteed. One of the advantages of these algorithms is the running time. [Tili et al. \(2013\)](#) used a particle swarm optimization to solve container loading problem, their computation results shows the performance of particle swarm optimization is related

to parameter setting, hence the solution quality is not always guaranteed. [Xue et al. \(2013\)](#) and [Hernández et al. \(2013\)](#) employed ant colony algorithm to solve yard truck scheduling and containership Stowage Problem respectively. Their computational results show the proposed ant colony algorithm perform rather good in their problem setting. These evidence suggests that this types of algorithms can obtainer good solutions fast. Other typical heuristic examples are tabu search method ([Glover et al., 1993, 2007](#)), genetic method, simulated annealing method, etc. In the following, we focus on the review of solution methods in the first category and second category.

In the first category, branch and bound and the cutting planes methods are two classic way of solving integer programs. As the difficulty of solving integer programs is due to the non-convexity of the feasible set, these two solution approaches employ two different ways to circumvent this difficulty. In B&B method, the feasible set is partitioned into smaller subsets denoted as nodes of B&B tree ([Floudas & Pardalos, 2001a](#)). In each node, we can calculate the objective value and obtain the corresponding bounds. For example, in minimization problem, the lower bound can be obtained by solving LP relaxation at a given node , and the upper bound is obtained when some of the nodes return integer values. This procedure proceeds until some criterion are met, one can refer to ([Floudas & Pardalos, 2001a](#)) for more information. In cutting plane method, the feasible integer set is gradually approximated by adding valid inequalities. These valid inequalities exclude part of the non-integer solutions and do not remove any feasible integer point ([Kelley, 1960](#)).

In the second category, lagrangian relaxation method is widely used for certain types of integer programming problems. One can refer to [Fisher \(1981\)](#) and [Geoffrion & Bride \(1978\)](#) for an introduction and review of lagrangian relaxation method. The basic idea of this method is to dualize the hard constraints of the origin problem. This produces a lagrangian problem that can be solved easily.

Consider the following integer problem,

$$\begin{aligned}
& \min \quad \mathbf{c}\mathbf{x} \\
& \text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \\
& \quad \quad \mathbf{D}\mathbf{x} \leq \mathbf{e} \\
& \quad \quad \mathbf{x} \in \mathcal{Z}^+
\end{aligned}$$

in which $\mathbf{A}$ is matrix, $\mathbf{x}$ is variable vector, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{e}$ are vectors in conformable dimensions. The constraints can be treated as two sets. Assume $\mathbf{A}\mathbf{x} = \mathbf{b}$ are hard constraints, dual this constraint we have the following lagrangian problem,

$$\begin{aligned}
Z_D(\boldsymbol{\mu}) = \min \quad & \mathbf{c}\mathbf{x} + \boldsymbol{\mu}(\mathbf{A}\mathbf{x} - \mathbf{b}) \\
& \text{s.t.} \quad \mathbf{D}\mathbf{x} \leq \mathbf{e} \\
& \quad \quad \mathbf{x} \in \mathcal{Z}^+
\end{aligned}$$

where $\boldsymbol{\mu}$ is a vector of lagrangian multipliers. When given $\boldsymbol{\mu}$, lagrangian problem can be solved easily of which the optimal value is a lower bound of the original problem (Fisher, 1981). For lagrangian relaxation method, we are more concerned with finding best lower bound of origin problem. Hence, we have the following dual problem,

$$Z_D = \max_{\boldsymbol{\mu}} Z_D(\boldsymbol{\mu})$$

Now, the problem is to find best $\boldsymbol{\mu}$ so a near optimal solution can be obtained. One of the commonly used method is subgradient method. The subgradient method is a adaptation of the gradient method, in which subgradients is used to replace gradients. The choice of step size was studied by Bazaraa & Sherali (1981). The subgradient method almost always work, however for some problem it may cause zigzag search which result in a slow convergence. Therefore, many other others have been proposed. Zhao et al. (1999) developed a surrogate subgradient method,. In this method, the proper direction of $\boldsymbol{\mu}$ can be obtained

without solving optimally all the subproblems. This means it takes little efforts to find better directions. [Bragin et al. \(2015\)](#) proof the convergence of surrogate subgradient method. [Belgacem & Amir \(2017\)](#) proposed a new deflected subgradient method, they used a convex combination of modified gradient technique and average direction strategy, the convergence is proved.

Another solution approach is Benders decomposition method. The classic Benders decomposition algorithm was mostly used for linear programming problems or mixed integer programming problems. Consider MIP as follows,

$$\begin{aligned} P : \quad & \min_{\mathbf{x}, \mathbf{y}} \quad \mathbf{c}^T \mathbf{x} + \mathbf{d}^T \mathbf{y} \\ \text{s.t.} \quad & \mathbf{Ax} \geq \mathbf{b} \\ & \mathbf{Dx} + \mathbf{Fy} \geq \mathbf{e} \\ & \mathbf{x}, \mathbf{y} \in \mathcal{Z}^+ \end{aligned}$$

In Benders decomposition algorithm, we first introduce an artificial variable $\mathbf{z} = \mathbf{d}^T \mathbf{y}$ assuming it has a lower bound $\underline{\mathbf{z}}$. Problem P is decomposed into master problem solving the master variables (assumed to be bounded),

$$\begin{aligned} MP : \quad & \min_{\mathbf{x}, \mathbf{z}} \quad \mathbf{c}^T \mathbf{x} + \mathbf{z} \\ \text{s.t.} \quad & \mathbf{Ax} \geq \mathbf{b} \\ & \mathbf{z} \geq \bar{\mathbf{z}} \\ & \mathbf{x} \in \mathcal{Z}^+ \end{aligned}$$

and subproblem solving the $\mathbf{y}$ variables with given tentative value $\bar{\mathbf{x}}$ from master problem,

$$\begin{aligned} SP(\bar{\mathbf{x}}) : \quad & \min_{\mathbf{y}} \quad \mathbf{d}^T \mathbf{y} \\ \text{s.t.} \quad & \mathbf{Fy} \geq \mathbf{e} - \mathbf{D}\bar{\mathbf{x}} \\ & \mathbf{y} \in \mathcal{Z}^+ \end{aligned}$$

The dual problem of $SP(\bar{\mathbf{x}})$ is used to generate cut added to MP,

$$\begin{aligned} DSP(\bar{\mathbf{x}}) : \quad & \min_{\boldsymbol{\lambda}} \quad \boldsymbol{\lambda}^T(\mathbf{e} - \mathbf{D}\bar{\mathbf{x}}) \\ & \text{s.t.} \quad \boldsymbol{\lambda}^T \mathbf{F} \leq \mathbf{d}^T \\ & \quad \boldsymbol{\lambda} \geq 0 \end{aligned}$$

Suppose the optimal solution of MP is $(\mathbf{x}^*, \mathbf{z}^*)$ in a iteration. Take this solution as an input to $DSP(\mathbf{x}^*)$ and solve $DSP(\mathbf{x}^*)$. If $DSP(\mathbf{x}^*)$ is unbounded, we obtain an unbounded ray denoted by $\bar{\boldsymbol{\lambda}}$, then the Benders' feasibility cut $\bar{\boldsymbol{\lambda}}^T(\mathbf{e} - \mathbf{D}\mathbf{x}) \leq 0$ is derived and added to the MP and solve the MP again. If the $DSP(\mathbf{x}^*)$ is bounded, we can obtain an optimality cut or denote the problem is optimal. Let η^* be the optimal value of $DSP(\mathbf{x}^*)$ and $\bar{\boldsymbol{\lambda}}$ be the optimal vertex of $DSP(\mathbf{x}^*)$. When $\eta^* \leq \mathbf{z}^*$ then current $(\mathbf{x}^*, \mathbf{z}^*)$ is optimal. Otherwise, Benders optimality cuts $\mathbf{z} \geq \bar{\boldsymbol{\lambda}}^T(\mathbf{e} - \mathbf{D}\mathbf{x})$ is added to the MP, and then solve the MP. The method is repeated until optimal solution is found. Note that in Benders decomposition method the subproblem should be linear program which strong dual holds.

The Benders decomposition method has attracted much research attention in the past few years. Recently, [Rahmaniani et al. \(2017\)](#) provided a state-of-the-art survey of the Benders decomposition algorithm. [Hooker & Ottosson \(2003\)](#) proposed a logic-based Benders decomposition. They Generalizes the linear programming dual to a so-called inference dual. [Codato & Fischetti \(2006\)](#) proposed a combinatorial Benders cuts method for MILP formulation, in which the generated cuts do not depend on the big-M formulation. The method can produce the LP relaxation that is considerably tighter than that associated with the origin MILP formulation. [Benders \(1962\)](#). [Rei et al. \(2009\)](#). [Fischetti et al. \(2017\)](#) redesigned Benders decomposition method for UFL problem and qUFL, the redesigned approaches are more scalable for large-scale input data. [Fakhri et al. \(2017\)](#) studied Benders decomposition method in which the subproblem has integer variables. In their approach, the subproblems are relaxed and solved in a

B&B framework. In each node of B&B tree, Benders cuts are generated if this node is proved to be valid for all its descendants.

2.4 Concluding remarks

This chapter presents a comprehensive literature review on container stowage problem, quay crane scheduling problem and integer programming problems. Since container stowage planning problem and quay crane scheduling problem are often formulated as integer programming or mixed integer programming which is difficult to solve, it is imperative to develop efficient algorithm to solve these problems. The existing research of solution methods for integer programming are also stated in this chapter.

From literature review, we have the following observations: (1) for container stowage problem, very few previous works studied it from a multi-port view, and most of them did not consider practical constraints, such as the stacking rule that heavy containers cannot be stowed on top of lighter containers, reefer containers shall be stowed in a location with power plug, etc. A multi-port view of slot planning is able to reduce the negative effect of stowing containers towards the subsequent port of call, and practical constraints ensure the model formulation is more realistic. Therefore, it is imperative for the operators to figure out the optimal container stowage plan considering the interactions among multiple ports. However, the inclusion of multi-port view and practical constraints complicates the model formulation and makes the problem even harder to solve, as we need to explicitly consider the weight relations and location properties among containers, as well as to handle the loading and discharging operations during port of calls in the model formulation. Besides, the inclusion of multi-port view makes the search space for optimal solution more complicated, as optimal solution for single port may not be global optimal for all ports. (2) for quay crane scheduling problem, many practical aspects have

been handled in the existing studies, like crane interference constraints and task precedence constraints. Besides, the integration of quay crane scheduling problem with other quay side problems, such as berth allocation problem, the quay crane assignment problem, etc., have also been proposed. However, there still exists a research gap when investigating crane double cycling strategies especially for multiple quay cranes. Basically, crane double cycling strategies can reduce the ineffective idle spreader moves substantially, and thus scheduling multiple quay crane in this way may increase the quay side operation efficiency significantly; (3) for integer programming problems, the solution methods are widely studied. As a widely applied solution method for solving large-scale integer programming problem, Benders decomposition method can not be directly implemented for solving integer programming with only integer variables. No previous studies have ever studied such a problem. Therefore, we focus on studying these types of problem in this thesis.

In the following chapter, a solution method for integer programming is firstly presented. Then, a model incorporated with practical constraints from a multi-port view are formulated for container stowage slot planning problem. Lastly, we investigate the quay crane scheduling problem with consideration of double cycling.

Chapter 3

SOLVING INTEGER LINEAR PROGRAMMING PROBLEMS USING BENDERS DECOMPOSITION METHOD

3.1 Introduction

Integer programming plays a fundamental role in supporting managerial decisions or modeling hard to solve problem. The simplest way to solve an ILP is to remove the integer restrictions on variables, and solve the corresponding LP. This approach is called the LP relaxation of the ILP. However, the solution obtained through this approach may not be optimal or may not even be feasible.

Benders decomposition is a well-known and successfully applied methodology for solving large-scale optimization problems, proposed by [Benders \(1962\)](#). The benefits of using Benders decomposition are: (1) by decomposition, we reduce the problem size or complexity by exploiting the problem structure; (2) the decomposed problems are independent, which allows solving master problem and subproblem with different methods. The classic Benders decomposition algorithm was mostly used for LP problems or MIP problems. In these problems, the subproblem is is a LP for which we can use standard duality theory to develop cuts. [Geoffrion \(1972\)](#) extended the Benders decomposition method for

nonlinear programming problem. The efficiency of Benders algorithm is greatly affected by the generated cuts.

For integer programming with only integer variables, the straightforward implementation of Benders algorithm and the generation of valid Benders cuts becomes complicated due to the duality gap of integer subproblems (Geoffrion, 1972; Chu & Xia, 2004; Floudas & Pardalos, 2001b). If the variables of integer programming is binary, one can use the no-good cuts to force at least one of the variables to change value (Benoist et al., 2002; Sadykov & Wolsey, 2006; Bockmayr & Pinar, 2006). However, no-good cuts will lead to an enumerative search in the solution space. This will lead to a slow convergence. For some specific problems, one can obtain better Benders cut. For example, in quay crane scheduling problem or traveling sales man problem, the cuts that break cycle results is generally better (Applegate et al., 1998, 2003; Chvátal et al., 2010; Pferschy & Staněk, 2017). As for more general integer programming, Hooker & Ottosson (2003) proposed a logic-based Benders decomposition method to generate valid Benders cuts. However, a large number of disjunction cuts are used in their methods, which complicate the master problem significantly.

As for integer programming involving only integer decision variables, Chu & Xia (2004) proposed a method to generate Benders cuts for the following type of problem,

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{y}} \quad & \mathbf{c}^T \mathbf{y} \\ \text{s.t.} \quad & \mathbf{D}\mathbf{y} \leq \mathbf{d} \\ & \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \{0, 1\}^n, \mathbf{y} \in \{0, 1\}^m \end{aligned}$$

This problem only contains $\mathbf{y}$ variables in the objective function. For the case where objective function involving both $\mathbf{x}$ and $\mathbf{y}$ variables, the method is not applicable. Hence, it is desirable to develop a more general solution method. To

resolve this problem, we develop a Benders decomposition method for generic integer programming problems in this chapter.

The main contributions of this chapter is to extend the Benders decomposition method to a more general pure integer programming problem, we introduce auxiliary variables into master problem and then formulate the subproblem as feasibility problem. Thereafter, using fixed subproblem and its dual, we can formulate a relaxed cut generation problem using complementary slackness condition, however the relaxed cut generation problem not always obtain valid Benders cut, in this situation we apply solution-elimination constraints to ensure the convergence of our method.

The rest of this chapter has the following structure. In section 3.2, we present a generic integer programs and decompose the problem into master problem and subproblem using Benders decomposition method. Section 3.3 introduces the selection of valid Benders cuts and present a cut generation problem to elicit it. Section 3.4 gives computational results. In section 3.5, we make the conclusion.

3.2 Generic integer linear programming and Benders decomposition

The method here can extends to integer programming with bounded variables, for notational simplicity we consider binary integer programming. Consider a

generic integer programming model as the following form,

$$\begin{aligned}
P : \quad & \min_{\mathbf{x}, \mathbf{y}} \quad \mathbf{c}^T \mathbf{x} + \mathbf{d}^T \mathbf{y} \\
& \text{s.t.} \quad \mathbf{D} \mathbf{y} \geq \mathbf{e} \\
& \quad \mathbf{F} \mathbf{x} \leq \mathbf{g} \\
& \quad \mathbf{A} \mathbf{x} + \mathbf{B} \mathbf{y} \geq \mathbf{b} \\
& \quad \mathbf{x} \in \{0, 1\}^n, \mathbf{y} \in \{0, 1\}^m
\end{aligned} \tag{3.1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are vectors of binary integer variables. Note that the differences of this formulation from that of [Chu & Xia \(2004\)](#) are the objective function contains both integer variables $\mathbf{x}$ and $\mathbf{y}$. Hence, this formulation represents a more general case which cannot be handled by the method of [Kim & Hooker \(2002\)](#) and [Chu & Xia \(2004\)](#).

Following the work flows of Benders decomposition, the origin problem P can be decomposed into the master problem MP solving the $\mathbf{y}$ variables and $\mathbf{z}$ variables,

$$\begin{aligned}
MP : \quad & \min_{\mathbf{y}} \quad \mathbf{d}^T \mathbf{y} + \mathbf{z} \\
& \text{s.t.} \quad \mathbf{D} \mathbf{y} \geq \mathbf{e} \\
& \quad \phi(\boldsymbol{\lambda}, \mathbf{y}) \leq 0, \boldsymbol{\lambda} \in \mathcal{F} \\
& \quad \phi'(\boldsymbol{\lambda}, \mathbf{y}, \mathbf{z}) \leq 0, \boldsymbol{\lambda} \in \mathcal{O} \\
& \quad \mathbf{y} \in \{0, 1\}^m
\end{aligned} \tag{3.2}$$

and subproblem $SP'(\bar{\mathbf{y}})$ solving the $\mathbf{x}$ variables with fixed variables $\bar{\mathbf{y}}$ from the master problem solution,

$$\begin{aligned}
SP'(\bar{\mathbf{y}}) : \quad & \min_{\mathbf{x}} \quad \mathbf{c}^T \mathbf{x} \\
& \text{s.t.} \quad \mathbf{F} \mathbf{x} \leq \mathbf{g} \\
& \quad \mathbf{A} \mathbf{x} \geq \mathbf{b} - \mathbf{B} \bar{\mathbf{y}} \\
& \quad \mathbf{x} \in \{0, 1\}^n
\end{aligned} \tag{3.3}$$

In the formulation of MP, $\mathcal{F}$ and $\mathcal{O}$ are sets of coefficient vectors for feasibility cuts and optimality cuts respectively. As noted in [Fischetti et al. \(2010\)](#), Benders' separation can always be rephrased as a pure feasibility problem. So, we can rewrite $SP'(\bar{\mathbf{y}})$ as follows,

$$\begin{aligned}
SP(\bar{\mathbf{y}}, \bar{\mathbf{z}}) : \quad & \min_{\mathbf{x}} \quad 0 \\
\text{s.t.} \quad & \mathbf{c}^T \mathbf{x} \leq \bar{\mathbf{z}} \\
& \mathbf{F}\mathbf{x} \leq \mathbf{g} \\
& \mathbf{A}\mathbf{x} \geq \mathbf{b} - \mathbf{B}\bar{\mathbf{y}} \\
& \mathbf{x} \in \{0, 1\}^n
\end{aligned} \tag{3.4}$$

where $\bar{\mathbf{z}}$ is the solution of MP. Thus, MP only involves feasibility cuts. In Benders algorithm, MP and $SP(\bar{\mathbf{y}})$ are iteratively solved. In each iteration, the information from $SP(\bar{\mathbf{y}})$, which is expressed as Benders cuts, is added to MP. When $SP(\bar{\mathbf{y}})$ is feasible, we obtain the optimal solution of problem P.

The core issue here is the generation of valid Benders cuts from $SP(\bar{\mathbf{y}}, \bar{\mathbf{z}})$. As noted in [Chu & Xia \(2004\)](#), a valid Benders cut should satisfy: (1) if the current tentative value of variables $\bar{\mathbf{y}}$ for master problem is an infeasible assignment, then this cut must cut off at least $\bar{\mathbf{y}}$; (2) any feasible solution of $\mathbf{y}$ variables in master problem must satisfy this cut. Since $SP(\bar{\mathbf{y}})$ is integer program, strong dual theory can not be applied directly to develop cuts.

3.2.1 Fixed subproblem and its dual

Suppose we know the value of $\mathbf{x}$ as $\bar{\mathbf{x}}$, we can formulate feasible subproblem of $\text{SP}(\bar{\mathbf{y}})$ with fixed value $\bar{\mathbf{x}}$ (Chu & Xia, 2004) as follows,

$$\begin{aligned}
 \text{SP}_f(\bar{\mathbf{y}}, \bar{\mathbf{z}}, \bar{\mathbf{x}}) : \quad & \min_{\mathbf{x}} \quad \mathbf{1}^T \mathbf{r}_0 + \mathbf{1}^T \mathbf{r}_1 + \mathbf{1}^T \mathbf{r}_2 \\
 & \text{s.t.} \quad \mathbf{c}^T \mathbf{x} - \mathbf{r}_0 \leq \bar{\mathbf{z}} \\
 & \quad \quad \mathbf{F}\mathbf{x} - \mathbf{r}_1 \leq \mathbf{g} \\
 & \quad \quad \mathbf{A}\mathbf{x} \geq \mathbf{b} - \mathbf{B}\bar{\mathbf{y}} - \mathbf{r}_2 \\
 & \quad \quad \mathbf{x} = \bar{\mathbf{x}} \\
 & \quad \quad \mathbf{x} \text{ free}; \mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2 \geq 0
 \end{aligned} \tag{3.5}$$

where $\mathbf{r}_0, \mathbf{r}_1$ and $\mathbf{r}_2$ are slack variables to the corresponding constraints of $\text{SP}(\bar{\mathbf{y}}, \bar{\mathbf{z}})$. $\bar{\mathbf{x}}$ represents a feasible solution in the domain, $\bar{\mathbf{x}} \in \{0, 1\}^n$. Hence, $\text{SP}_f(\bar{\mathbf{y}}, \bar{\mathbf{z}}, \bar{\mathbf{x}})$ is always feasible and is a continuous linear programming. Therefore, strong dual theory is applicable to $\text{SP}_f(\bar{\mathbf{y}}, \bar{\mathbf{x}})$.

Now, we can apply dual theory to $\text{SP}_f(\bar{\mathbf{y}}, \bar{\mathbf{x}})$ to derive cuts for master problem. The dual formulation of $\text{SP}_f(\bar{\mathbf{y}}, \bar{\mathbf{x}})$ is given as follows:

$$\begin{aligned}
 \text{DSP}(\bar{\mathbf{y}}, \bar{\mathbf{z}}, \bar{\mathbf{x}}) : \quad & \max_{\lambda_0, \lambda_1, \lambda_2, \mu} \quad -\lambda_0 \bar{\mathbf{z}} - \mathbf{g}^T \lambda_1 + (\mathbf{b} - \mathbf{B}\bar{\mathbf{y}})^T \lambda_2 + \bar{\mathbf{x}}^T \mu \\
 & \text{s.t.} \quad \lambda_0 \mathbf{c} + \mathbf{F}^T \lambda_1 - \mathbf{A}^T \lambda_2 - \mu = 0 \\
 & \quad \quad \lambda_0, \lambda_1, \lambda_2 \leq 1 \\
 & \quad \quad \mu \text{ free}; \lambda_0, \lambda_1, \lambda_2 \leq 0
 \end{aligned} \tag{3.6}$$

Denote $\mathbf{r} = (\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2)$, $\lambda = (\lambda_0, \lambda_1, \lambda_2)$ and $\Phi(\bar{\mathbf{y}}, \bar{\mathbf{z}}) = (-\bar{\mathbf{z}}, -\mathbf{g}^T, (\mathbf{b} - \mathbf{B}\bar{\mathbf{y}})^T)$ for ease of presentation. We rewrite dual objective (3.6) as $\Phi(\bar{\mathbf{y}}, \bar{\mathbf{z}})\lambda + \bar{\mathbf{x}}^T \mu$.

We can observe that the optimal value of $\text{DSP}(\bar{\mathbf{y}}, \bar{\mathbf{z}}, \bar{\mathbf{x}})$ is correlated with the value of $\bar{\mathbf{x}}$. Since $\bar{\mathbf{x}} \in \{0, 1\}^n$, thus we have $\mathcal{N} = 2^n$ possible fixed subproblems and corresponding dual problems. As $\text{SP}_f(\bar{\mathbf{y}}, \bar{\mathbf{x}})$ is linear programming, the strong

dual holds for $\text{SP}_f(\bar{\mathbf{y}}, \bar{\mathbf{x}})$ and $\text{DSP}(\bar{\mathbf{y}}, \bar{\mathbf{z}}, \bar{\mathbf{x}})$. Therefore, we can use these solutions of N fixed subproblems to obtain a generic Benders cut, which is expressed as the disjunction of N linear inequalities:

$$\begin{aligned} \Phi(\mathbf{y}, \mathbf{z})\tilde{\boldsymbol{\lambda}}^1 + (\tilde{\mathbf{x}}^1)^T \tilde{\boldsymbol{\mu}}^1 &\leq 0 \\ \Phi(\mathbf{y}, \mathbf{z})\tilde{\boldsymbol{\lambda}}^2 + (\tilde{\mathbf{x}}^2)^T \tilde{\boldsymbol{\mu}}^2 &\leq 0 \\ \dots & \\ \Phi(\mathbf{y}, \mathbf{z})\tilde{\boldsymbol{\lambda}}^N + (\tilde{\mathbf{x}}^N)^T \tilde{\boldsymbol{\mu}}^N &\leq 0 \end{aligned} \tag{3.7}$$

where $(\tilde{\mathbf{x}}^1, \tilde{\mathbf{x}}^2, \dots, \tilde{\mathbf{x}}^N)$ are all the possible $\mathbf{x}$ values, $\tilde{\boldsymbol{\lambda}}$ and $\tilde{\boldsymbol{\mu}}$ are corresponding dual values. Cuts (3.7) formulate a valid Benders cut; however it needs to solve a large number of dual fixed subproblem.

Lemma 1. *Cuts (3.7) is a valid Benders cut, and there exists a unique linear inequality out of N linear inequalities that (1) this unique cut can excludes infeasible solution $\bar{\mathbf{y}}$; (2) any feasible solution of $\mathbf{y}$ satisfy this cut.*

Proof. As Cuts (3.7) contains all the possible $\mathbf{x}$ values, it is reasonable to suppose $\mathbf{x}^*$ is a feasible solution corresponding to infeasible solution $\bar{\mathbf{y}}$ and $\bar{\mathbf{z}}$, denotes its dual variable as $\bar{\boldsymbol{\lambda}}$ and $\bar{\boldsymbol{\mu}}$. It satisfies,

$$\Phi(\bar{\mathbf{y}}, \bar{\mathbf{z}})\bar{\boldsymbol{\lambda}} + (\mathbf{x}^*)^T \bar{\boldsymbol{\mu}} > 0 \tag{3.8}$$

Since $\mathbf{x}^*$ is feasible, there must exists a feasible $\mathbf{y}^*$ and $\mathbf{z}^*$, denotes its dual variables as $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$. Since $\mathbf{x}^*$ and $\mathbf{y}^*$ are feasible solution, it satisfy,

$$\Phi(\mathbf{y}^*, \mathbf{z}^*)\boldsymbol{\lambda}^* + (\mathbf{x}^*)^T \boldsymbol{\mu}^* = 0 \tag{3.9}$$

As $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are independent of $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, hence the following inequality holds,

$$\Phi(\mathbf{y}^*, \mathbf{z}^*)\bar{\boldsymbol{\lambda}} + (\mathbf{x}^*)^T \bar{\boldsymbol{\mu}} \leq \Phi(\mathbf{y}^*, \mathbf{z}^*)\boldsymbol{\lambda}^* + (\mathbf{x}^*)^T \boldsymbol{\mu}^* = 0 \tag{3.10}$$

Therefore, cuts $\Phi(\mathbf{y}, \mathbf{z})\bar{\boldsymbol{\lambda}} + (\mathbf{x}^*)^T \bar{\boldsymbol{\mu}} \leq 0$ is a valid Benders cut. $\square$

3.2.2 Complementary slackness condition

The optimal solution of $\text{SP}_f(\bar{\mathbf{y}}, \bar{\mathbf{x}})$ and $\text{DSP}(\bar{\mathbf{y}}, \bar{\mathbf{z}}, \bar{\mathbf{x}})$ are related by complementary slackness condition (Lasdon, 2002). Hence, we have the following complementary slackness constraints,

$$\begin{cases} (\mathbf{c}^T \mathbf{x} - \mathbf{r}_0 - \bar{z})^T \boldsymbol{\lambda}_0 = 0 \\ (\mathbf{F}\mathbf{x} - \mathbf{r}_1 - \mathbf{g})^T \boldsymbol{\lambda}_1 = 0 \\ (\mathbf{b} - \mathbf{B}\bar{\mathbf{y}} - \mathbf{r}_2 - \mathbf{A}\mathbf{x})^T \boldsymbol{\lambda}_2 = 0 \\ \mathbf{r}_i(\boldsymbol{\lambda}_i - \mathbf{1}) = 0, i = 0, 1, 2 \end{cases} \quad (3.11)$$

together with constraints (3.5) and (3.6).

$\mathbf{r}_i(\boldsymbol{\lambda}_i - \mathbf{1}) = 0$ is redundancy in (3.11), which is proved in Chu & Xia (2004).

Therefore, the complementary condition can be simplified as follows,

$$\begin{cases} \mathbf{c}^T \mathbf{x} - \mathbf{r}_0 \leq \bar{z} \\ \mathbf{F}\mathbf{x} - \mathbf{r}_1 \leq \mathbf{g} \\ \mathbf{A}\mathbf{x} \geq \mathbf{b} - \mathbf{B}\bar{\mathbf{y}} - \mathbf{r}_2 \\ \boldsymbol{\lambda}_0 \mathbf{c}^T \mathbf{x} - \mathbf{r}_0 = \boldsymbol{\lambda}_0 \bar{z} \\ \boldsymbol{\lambda}_1 \mathbf{F}\mathbf{x} - \mathbf{r}_1 = \mathbf{g} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \mathbf{A}\mathbf{x} = \mathbf{b} \boldsymbol{\lambda}_2 - \mathbf{B}\bar{\mathbf{y}} \boldsymbol{\lambda}_2 - \mathbf{r}_2 \\ \mathbf{r} \geq 0, \boldsymbol{\lambda} \leq 1 \end{cases} \quad (3.12)$$

3.2.3 Relaxed cut generation problem

Although Lemma 1 indicates there exist a unique Benders cut in (3.7), it can not be directly derived. But if a Benders cut in (3.7) satisfies the following sufficient

condition, then it is a valid Benders cut, for ease of understanding, I present the proof procedure of [Chu & Xia \(2004\)](#) here

Theorem 1. Denote feasible solution pools of problem $SP_f(\bar{\mathbf{y}}, \bar{\mathbf{z}}, \bar{\mathbf{x}})$ as $\mathbf{X}$, suppose a solution $\tilde{\mathbf{x}} \in \mathbf{X}$ such that $\tilde{\mathbf{x}}$ and the corresponding dual $\tilde{\boldsymbol{\mu}}$ satisfy

$$\tilde{\mathbf{x}}^T \tilde{\boldsymbol{\mu}} \leq \mathbf{x}^T \tilde{\boldsymbol{\mu}}, \quad \forall \mathbf{x} \in \mathbf{X} \quad (3.13)$$

then the linear inequality

$$\Phi(\mathbf{y}, \mathbf{z}) \tilde{\boldsymbol{\lambda}} + \tilde{\mathbf{x}}^T \tilde{\boldsymbol{\mu}} \leq 0 \quad (3.14)$$

is a valid Benders cut.

Proof. First, we proof the infeasible solution $(\bar{\mathbf{y}}, \bar{\mathbf{z}})$ can be excluded by cut (3.14). Since $(\bar{\mathbf{y}}, \bar{\mathbf{z}})$ is infeasible for the subproblem $SP(\bar{\mathbf{y}}, \bar{\mathbf{z}})$, then the objective value of $SP_f(\bar{\mathbf{y}}, \bar{\mathbf{z}}, \bar{\mathbf{x}})$ is greater than zero, so do the $DSP(\bar{\mathbf{y}}, \bar{\mathbf{z}}, \bar{\mathbf{x}})$. Hence, any of the linear inequalities stated in (3.7) are violated by $(\bar{\mathbf{y}}, \bar{\mathbf{z}})$, so cut (3.14) excludeds infeasible $(\bar{\mathbf{y}}, \bar{\mathbf{z}})$.

Second, we proof the feasible solution $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ satisfy cut (3.14). Denote $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ as any feasible assignment from MP. Hence, we must have corresponding $\hat{\mathbf{x}}$ making the objective value of $SP_f(\hat{\mathbf{y}}, \hat{\mathbf{z}}, \hat{\mathbf{x}})$ 0 and corresponding $(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}})$ making the objective value of $DSP(\hat{\mathbf{y}}, \hat{\mathbf{z}}, \hat{\mathbf{x}})$ 0. Hence, we have,

$$\Phi(\hat{\mathbf{y}}, \hat{\mathbf{z}}) \hat{\boldsymbol{\lambda}} + \hat{\mathbf{x}}^T \hat{\boldsymbol{\mu}} = 0$$

Since the feasible domain of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are independent of the values of $\mathbf{y}$, $\mathbf{z}$ and $\mathbf{x}$, the optimal solution of $DSP(\bar{\mathbf{y}}, \bar{\mathbf{z}}, \bar{\mathbf{x}})$ denoted by $(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}})$ is also a feasible solution to $DSP(\hat{\mathbf{y}}, \hat{\mathbf{z}}, \hat{\mathbf{x}})$. Therefore, we have,

$$\Phi(\hat{\mathbf{y}}, \hat{\mathbf{z}}) \bar{\boldsymbol{\lambda}} + \hat{\mathbf{x}}^T \bar{\boldsymbol{\mu}} \leq \Phi(\hat{\mathbf{y}}, \hat{\mathbf{z}}) \hat{\boldsymbol{\lambda}} + \hat{\mathbf{x}}^T \hat{\boldsymbol{\mu}} = 0$$

From condition (3.13), we have $\bar{\mathbf{x}}^T \bar{\boldsymbol{\mu}} \leq \hat{\mathbf{x}}^T \bar{\boldsymbol{\mu}}$

$$\Phi(\hat{\mathbf{y}}, \hat{\mathbf{z}}) \bar{\boldsymbol{\lambda}} + \bar{\mathbf{x}}^T \bar{\boldsymbol{\mu}} \leq \Phi(\hat{\mathbf{y}}, \hat{\mathbf{z}}) \bar{\boldsymbol{\lambda}} + \hat{\mathbf{x}}^T \bar{\boldsymbol{\mu}} \leq 0$$

Hence, feasible solution $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ satisfy the cut. $\square$

Inequality (3.13) can be satisfied with enforce the following constraints,

$$\begin{cases} \mathbf{x}_i \boldsymbol{\mu}_i \leq 0 \\ (1 - \mathbf{x}_i) \boldsymbol{\mu}_i \geq 0 \end{cases} \quad (3.15)$$

However, with enforce (3.15), the problem may become infeasible, consider a relax of (3.15), we have the following relaxed benders cut generation problem,

$$\begin{aligned} \text{CGP}(\bar{\mathbf{y}}, \bar{\mathbf{z}}) : \quad & \min_{\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}} \quad \mathbf{1}^T \mathbf{p} + \mathbf{1}^T \mathbf{q} \\ \text{s.t} \quad & \mathbf{c}^T \mathbf{x} - \mathbf{r}_0 \leq \bar{\mathbf{z}} \\ & \mathbf{F} \mathbf{x} - \mathbf{r}_1 \leq \mathbf{g} \\ & \mathbf{A} \mathbf{x} \geq \mathbf{b} - \mathbf{B} \bar{\mathbf{y}} - \mathbf{r}_2 \\ & \boldsymbol{\lambda}_0 \mathbf{c}^T \mathbf{x} - \mathbf{r}_0 = \boldsymbol{\lambda}_0 \bar{\mathbf{z}} \\ & \boldsymbol{\lambda}_1 \mathbf{F} \mathbf{x} - \mathbf{r}_1 = \mathbf{g} \boldsymbol{\lambda}_1 \\ & \boldsymbol{\lambda}_2 \mathbf{A} \mathbf{x} = \mathbf{b} \boldsymbol{\lambda}_2 - \mathbf{B} \bar{\mathbf{y}} \boldsymbol{\lambda}_2 - \mathbf{r}_2 \\ & \mathbf{x}_i \boldsymbol{\mu}_i - \mathbf{p}_i \leq 0, \forall i \\ & (1 - \mathbf{x}_i) \boldsymbol{\mu}_i + \mathbf{q}_i \geq 0, \forall i \\ & \mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{p}_i, \mathbf{q}_i \geq 0, \boldsymbol{\lambda} \leq 1 \end{aligned} \quad (3.16)$$

In the formulaiton of $\text{CGP}(\bar{\mathbf{y}}, \bar{\mathbf{z}})$, $\mathbf{x}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and $\mathbf{r}$ are variables to be determined. Denote the objective of $\text{CGP}(\bar{\mathbf{y}}, \bar{\mathbf{z}})$ as Φ_{CGP} , the generated cut have the form as following,

$$\Phi(\mathbf{y}, \mathbf{z}) \tilde{\boldsymbol{\lambda}} + \tilde{\mathbf{x}}^T \tilde{\boldsymbol{\mu}} - \Phi_{CGP} \leq 0 \quad (3.17)$$

3.3 Benders Decomposition Applied to Integer Programs

Section 3.2 presents the relaxed Benders cut generation problem, however the Benders cut generated by Benders cut generation problem may not necessarily exclude infeasible solution. Since $\mathbf{z}$ in the master problem is not binary variables, so when the Benders cut obtained through cut generation problem cannot cut off infeasible solution in master problem, we cannot directly used no-good cuts to exclude incumbent solution.

3.3.1 Integer Benders Decomposition Algorithm

To handle the above-mentioned situation, we introduce two related integer programs: (1) $P(\bar{\mathbf{x}}, \mathbf{Y})$ be the integer program $P(\mathbf{X}, \mathbf{Y})$, with $\mathbf{x} = \bar{\mathbf{x}}$; (2) $P(\mathbf{X}, \bar{\mathbf{y}})$ be the integer program $P(\mathbf{X}, \mathbf{Y})$, with $\mathbf{y} = \bar{\mathbf{y}}$. $\mathbf{X}$ and $\mathbf{Y}$ are the domains of variable $\mathbf{x}$ and $\mathbf{y}$.

Based on the relaxed integer Benders cut problem and the two related integer programs, the integer Benders decomposition algorithm can be specified as follows,

Step 1: initialize, decompose the origin problem into master problem $MP(\mathbf{y}, \mathbf{z})$ and subproblem $SP(\mathbf{x})_{\bar{\mathbf{y}}, \bar{\mathbf{z}}}$, construct relaxed Benders cut problem $CGP(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})_{\bar{\mathbf{y}}, \bar{\mathbf{z}}}$.

Let lower bound value $\underline{z} = -\infty$ and upper bound value $\bar{z} = \infty$

Step 2: solve the master problem $MP(\mathbf{y}, \mathbf{z})$ to optimal, obtain the solution $(\bar{\mathbf{y}}, \bar{\mathbf{z}})$ and objective value z' , if $\underline{z} \geq z'$ then let $\underline{z} = z'$.

Step 3: solve $CGP(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})_{\bar{\mathbf{y}}, \bar{\mathbf{z}}}$ to obtain optimal solution $(\tilde{\mathbf{x}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}})$, generate the relaxed cut,

$$\Phi(\mathbf{y}, \mathbf{z})\tilde{\boldsymbol{\lambda}} + (\tilde{\mathbf{x}}')^T \tilde{\boldsymbol{\mu}} \leq 0 \quad (3.18)$$

- (a) if the objective value of $\text{CGP}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})_{\bar{\mathbf{y}}, \bar{\mathbf{z}}}$ equals 0, the above cut is valid cut, add it to $\text{MP}(\mathbf{y}, \mathbf{z})$, go to Step 2.
- (b) if the objective value of $\text{CGP}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})_{\bar{\mathbf{y}}, \bar{\mathbf{z}}}$ is greater than 0, and the incumbent $(\bar{\mathbf{y}}, \bar{\mathbf{z}})$ violates the above cut, then the above cut is valid cut, add it to $\text{MP}(\mathbf{y}, \mathbf{z})$, go to Step 2.
- (c) if the objective value of $\text{CGP}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})_{\bar{\mathbf{y}}, \bar{\mathbf{z}}}$ is greater than 0, and the incumbent $(\bar{\mathbf{y}}, \bar{\mathbf{z}})$ does not violate the above cut, then go to Step 4.

Step 4: solve $P(\mathbf{X}, \bar{\mathbf{y}})$, let $\mathbf{Y} \setminus \{\bar{\mathbf{y}}\}$. If $P(\mathbf{X}, \bar{\mathbf{y}})$ is infeasible, go to step 2; denote $\hat{z}$ as the optimal solution of $P(\mathbf{X}, \bar{\mathbf{y}})$, if $\hat{z} < \bar{z}$, let $\bar{z} = \hat{z}$, go to Step 5. if $\bar{z} \leq \underline{z}$, go to Step 6.

Step 5: solve $P(\bar{\mathbf{x}}, \mathbf{Y})$, let $\mathbf{X} \setminus \{\bar{\mathbf{x}}\}$. If $P(\bar{\mathbf{x}}, \mathbf{Y})$ is infeasible, go to step 2; denote $\hat{z}$ as the optimal solution of $P(\bar{\mathbf{x}}, \mathbf{Y})$, if $\hat{z} < \bar{z}$, let $\bar{z} = \hat{z}$; if $\bar{z} \leq \underline{z}$, go to Step 6.

Step 6: return the solution.

Theorem 2. *The Integer Benders Decomposition Algorithm terminates in finite step and return the optimal solution of the origin problem.*

Proof. The proof is trivial, since Step 3, Step 4 or Step 5 is guaranteed to introduce a solution elimination constraint that eliminates a partial solution, the algorithm is finite. $\square$

3.4 Computational Experiments

In this section, we present our solution approach for the capacitated facility location problem in which each customer is served by a single facility. We use the

following mathematical formulation for single source capacitated facility location problems (Holmberg et al., 1999),

$$\begin{aligned}
\min z &= \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \\
\text{s.t. } &\sum_{j \in J} a_j x_{ij} \leq b_i y_i, \forall i \in I \\
&\sum_{i \in I} x_{ij} = 1, \forall j \in J \\
&x_{ij} - y_i \leq 0, \forall i \in I, j \in J \\
&x_{ij} \in \{0, 1\}, y_i \in \{0, 1\}
\end{aligned} \tag{3.19}$$

The computational experiments are performed on a personal computer with 16 gigabytes of RAM, Mac operating system and Intel Core i7 processors with 2.3 gigahertz cores. The ILP model (3.19) and our Benders solution approach are implemented in Python and embedded within the ILOG Concert Technology frame work, based on ILOG CPLEX 12.7.

3.4.1 Test instances

We have randomly generated three sets of test problems, each with different sizes and properties. Table 3.1 gives a summary of the sizes and the quotients between the total capacity and total demand. The coordinates of locations in three sets were generated as $U(10, 200)$ (We use the notation $U(a, b)$ for a uniform distribution in the interval $[a, b]$). The cost are determined as $c_{ij} = 4e_{ij}$ where e_{ij} is the Euclidean distance from facility i to customer j . The fixed cost are generate as $U(500, 1500)$ for all three sets.

Table 3.1
The sizes of the test problems

Set	Problems	$ I $	$ J $	Ratio
1	p1--p8	10	50	3.5--5.2
2	p9--p15	20	50	5.4--6.9
3	p16--p20	30	100	5.1--5.9

Note: $|I|$ and $|J|$ are the numbers of facilities and customers respectively.

3.4.2 Computational results

Table 3.2, Table 3.3 and Table 3.4 give the results of the computation tests. For each of the problem, we give the results of direct solution with CPLEX and our integer Benders decomposition method (IBD). For CPLEX, we report the optimal objective value, computation time and the solution gap. For IBD, we report the optimal objective value, computation time and number of iterations to obtain the optimal solution.

Table 3.2
Computational results for problems p1--p8

Instance	CPLEX			IBD		
No.	z	time	gap	z	time	#iter
1	12159.75	0.46	0%	12159.75	2.2	24
2	13019.54	0.37	0%	13019.54	0.57	14
3	11262.02	0.11	0%	11262.02	0.15	10
4	13739.73	0.59	0%	13739.73	0.92	14
5	16619.37	0.61	0%	16619.37	0.25	10
6	12211.91	0.07	0%	12211.91	0.02	6
7	15965.07	0.28	0%	15965.07	12.0	25
8	13407.760	0.19	0%	13407.760	8.50	23

Table 3.3
Computational results for problems p9--p15

Instance	CPLEX			IBD		
No.	z	time	gap	z	time	#iter
9	13432.7	0.55	0%	13432.7	15.0	65
10	11436.59	0.43	0%	11436.59	0.24	17
11	10624.27	0.10	0%	10624.27	0.0	1
12	12356.0	0.32	0%	12356.0	1.82	28
13	12503.4	0.22	0%	12503.4	0.02	6
14	10932.56	0.54	0%	10932.56	0.01	4
15	10588.48	0.10	0%	10588.48	0.01	2

Table 3.4
Computational results for problems p16--p20

Instance	CPLEX			IBD		
No.	z	time	gap	z	time	#iter
16	28703.29	374.33	0.03%	28703.29	127.3	28
17	26800.05	76.5	0.06%	26800.05	35.4	26
18	27614.17	66.45	0.0%	27614.17	33.2	32
19	26509.64	51.09	0.08%	26509.64	43.2	17
20	24245.45	10.48	0.0%	24245.45	6.34	12

Table 3.2 summarize the computation results for relative easy case (set 1 case). We can observed that IBD obtain optimal solution for all 8 instances, although it does not outperform the directly solving through CPLEX. One of the reasons is that the formulation structure is not so suitable for using Benders decomposition method. Table 3.3 presents the computation results for all the instances in set 2. As we can see, most of time LBD outperforms the directly solving, but

there is a case IBD is far worse than directly solving by CPLEX, that may be because the master problem takes a long time to obtain optimal solution as there are much more variables in master problem than that of subproblem. Table 3.4 gives the computation results for instances in set 3 which represent a set of hard instance. Also, IBD requires less computation time than CPLEX to obtain the optimal value.

In summary, Integer Benders decomposition method is competitive with direct solving through CPLEX. In particular, in some case IBD obtain optimal solution through few iterations. Future work will improve the implementation of master problem solving which in the single capacitated facility location takes a long time to obtain optimal solution.

3.5 Conclusion

In this chapter, we present a Benders decomposition method for generic integer programming problems. For the considered pure integer problems, the objective function both involve in x variable and y variable. we introduce auxiliary variables into master problem and then formulate the subproblem as feasibility problem. Then we used fixed subproblem and its dual to formulate a relaxed cut generation problem, since the Benders cut obtained by relaxed cut generation problem may not always cut off infeasible problem, and because of introduction of auxiliary variables in master problem, no-good cut can not directly added to master problem when the Benders cut is not valid. To resolve this problem, we introduced two fixed integer problem, this two fixed integer problem will be invoked to add solution elimination constraint when the Benders cut is not valid. The solution approach have been test through single source capacitated facility location problem. Although the computation results shows that our method not

always outperform CPLEX, there are still a lot we can do to improve the algorithm. For example, the master problem can be solved with proper algorithm like Constraint Programming.

Chapter 4

MODELING AND SOLVING MULTI-PORT CONTAINER STOWAGE SLOT PLANNING PROBLEM

4.1 Introduction

Containerization technique greatly improved the efficiency of international cargo shipping. Around 90% of non-bulk cargo worldwide is moved by container ships ([Secretariat, 2015](#)). In 2017, global containerized trade had an annual growth of 6.1% and was estimated at 138.5 million of TEUs (Twenty-Foot Equivalent Units), it is expected that demand will grow by 4%-4.5% in 2018¹. In keeping pace with global containerized trade, the number and total capacity of global container ships climb up to a new level. Over the period 1985--2015, the global container ship fleet increased by 5.2%, and the nominal fleet growth of 5.5% for 2018 is estimated². Today, the capacity of largest container ship in the world reaches 21413 TEUs³.

One of the prominent operations for container shipping is loading and unloading during calling a sequence of ports according to a planned route. At a port terminal, containers destined for this port are unloaded and containers with port

¹<http://www.marinepoland.com/news/trends-and-challenges-for-the-global-container-market-in-2018.html>

²https://www.bimco.org/news/market_analysis/2018/20180903-container

³<https://www.oocl.com/eng/pressandmedia/pressreleases/2017/Pages/12may17.aspx>

of destination beyond this port are loaded thereafter. Due to the fact that containers can only be accessed and retrieved from the top of storage area within container ship, some containers may have to be removed from the ship temporarily in order to access the containers underneath it. This kind of containers removing from container ship are called shifts. In real operations, shifts may arise at a port terminal from the reasons that containers destined for this port were stowed below the containers with port of destination beyond this port, or the container location has to be reordered at this port to reduce the shifts in the subsequent port of call. The first case is called necessary shift and the second case voluntary shift. For the voluntary shifts, it is imperative for shipping company and terminal operators to make optimal decision on whether it is profitable or not to do so.

Shifting operations are unproductive, costly and time consuming transportation activities for both shipping company and terminal operators. During the port of rotations, container ship may encounter a large number of shifting operations if the stowage plans, which determines the arrangement of containers on container vessel, are not properly designed. As noted in [Delgado et al. \(2012a\)](#), efficient stowage plan can shorten the container ship staying time at port due to the improvement of loading and unloading activities, hence the port congestion is eased as well as the port fees for shipping company is reduced. Moreover, shipping company can slow down the speed of container ship at sea to reduce the bunker consumption with the reduction of port staying time while the ship turnaround time is maintained, which is also beneficial to the environment because the reduction of sailing speed decreases the consumption of bunker fuel and thus the amount of gas emission ([Wang et al., 2013](#)).

In this chapter, we focus on container stowage slot planning problem. Arranging containers into proper locations within container ship is a very complicated task. Currently, a successfully used approach to solve stowage planning problem is to decompose the problem into master planning phase and slot planning

phase, as illustrated in Figure 4.1. In the master planning phase, containers are distributed to bay sections of the vessels. Ship stability and hatch overstowage minimization are explicitly considered in this phase. With the information from master planning phase, the slot planning phase assigns the containers to specific slots in each bay sections. However, a practical slot planning problems should take into account more constraints from container ship structures. Specifically, there are a wide variety of container types and a set of stacking rules to be considered. For example, reefer containers shall be stowed in a location with power plug, hazardous containers must be placed at a location based on a complex set of separation and heavy container can not be stowed on top of light container, etc. The practical constraints represent the necessary rules when placing the containers on container ship. Such practical constraints could be stability constraints, hazardous containers constraints, stack weight constraints, etc. A generated stowage plan should satisfy these constraints, as the violation of these constraints could either bring safety issues or induce higher operational costs.

Figure 4.1: The master planning phase and slot planning phase of stowage planning

The main contributions of this chapter are three folds. First, the slot planning problem are modeled from a multi-port view with incorporation of more practical constraints. A multi-port view of slot planning is essential in reducing the negative effect of stowing containers in the subsequent port of call. The works by [Avriel & Penn \(1993\)](#), [Avriel et al. \(1998\)](#) and [de Azevedo et al. \(2014\)](#) solved

the problem from a multi-port view, however their model is inflexible in capturing more practical constraints of real slot planning. A more realistic slot planning problem was modelled by [Delgado et al. \(2012b\)](#) and [Francisco et al. \(2016\)](#), but they only considered loading containers at a single port. Second, we extend the slot planning problem by explicitly including the constraint that the weight of containers stowed in the same ship-stack does not increase from the bottom to the top. Indeed, this constraint is practically required in real loading operations but was not fulfilled in the existing model formulations like [Delgado et al. \(2012b\)](#) and [Francisco et al. \(2016\)](#). Third, as the slot planning problem considered in this study is NP-hard ([Avriel et al., 2000](#); [Delgado et al., 2012b](#)), we propose two specific solution algorithms to solve it, i.e., a slot attributes based heuristic method and tabu search method.

The reminder of this chapter is organized as follows. In Section [4.2](#), we describes the considered problem in this paper and present a mathematical model. Section [4.3](#) introduce the solution methods for the formulated model. In Section [4.4](#), some computational experiments are conducted and final conclusions are discussed in Section [4.5](#).

4.2 Problem description and mathematical formulation

As mentioned previously, slot planning problem focuses on assigning specific slots to containers with a solution given by master planning. Thus, the exogenously given informations in slot planning problem include: the port of calls of container ship, the configuration of block cargo space in terms of stack J and tier $K(j)$, the set of containers I , the number of containers to be loaded and unloaded at each port of call, the weight of each container and the maximum weight limits of each stack.

Normally, containers can be classified into 20ft, 40ft and high cube containers according to the size. Besides, containers can also be distinguished as reefer containers, hazardous containers, etc. A practical slot planning problem should include stacking rules for all these types of containers. For simplicity, we only consider stacking rules for general 40ft containers in model formulation, as an observable trend of rapidly growing usage of the 40ft container supports the expectation that 40ft container would replace its 20ft counterpart as the most commonly used container worldwide. Later, we will demonstrate that stacking rules for reefer containers and hazardous containers can be easily incorporated in our model.

In generating slot planning with consideration of multi-port interaction, containers can be voluntarily shifted at any intermediate port so as to minimize the total incurred shifts along the entire shipping route. This can not be resolved in the formulation by [Delgado et al. \(2012b\)](#) and [Francisco et al. \(2016\)](#), which is a major motivation of this study to propose a model formulation to acquire the optimal multi-port stowage slot plan.

Let us now introduce the mathematical notation. Table [4.1](#) shows the notation of Sets, parameters and decision variables used in the model formulation.

Table 4.1
Sets, parameters and variables in the model

Sets	
P	Index set of the order of the port rotations of a ship route, $P = \{1, 2, \dots, N\}$
I	Set of all containers
$N(p)$	Set of containers whose origin port is p , $N(p) \subseteq I, p \in P$,
$D(p)$	Set of container whose port of destination is port p , $D(p) \subseteq I, p \in P$
J	Set of stacks
$K(j)$	Set of tiers within the stack j , $j \in J$, $k_0(j)$ the lowest tier of stack j and $\bar{k}(j)$ the highest tier of stack j
W_i^c	The weight of container i , $i \in I$
W_j^s	The maximum allowed weight of stack j , $j \in J$
c_p	Cost of shifting a container at port p
Variables	
$x_{jki}^p \in \{0, 1\}$	Equals 1 when container i is stowed at stack j tier k at port p , otherwise 0
$y_{ip} \in \{0, 1\}$	Equals 1 when container i is shifted at port p , otherwise 0

To facilitate our formulation, we further define the following container set based on the notation given in Table 4.1. Let $G(p)$ be the container set of port p denoting the containers loaded at port before p and destined to port beyond port p . Hence, we have $G(p) = (\cup_{m=1}^{p-1} N(m)) \cap (\cup_{m=p+1}^N D(m))$. Denote $H(p)$ as all the containers on the container ship after loading operations at port p , $H(p) = G(p) \cup N(p)$. Note that $G(1) = \emptyset$ and $H(N) = \emptyset$ according to the definition.

The integer programming model F for the MP-SPP is defined as ,

$$[\text{Model } F] \min \sum_{p \in P} \sum_{i \in G(p)} c_p y_{ip} \quad (4.1)$$

$$\text{s.t.} \quad \sum_{k'=1}^{k-1} \sum_{i' \in D(p)} x_{jk'i'}^{p-1} \leq (k-1) (y_{ip} - x_{jki}^{p-1} + 1), \quad (4.2)$$

$$\forall p \in P \setminus \{1\}, \forall j \in J, \forall k \in K(j) \setminus \{1\}, \forall i \in G(p)$$

$$\sum_{k'=1}^{k-1} \sum_{i' \in G(p)} y_{i'p} x_{jk'i'}^{p-1} \leq (k-1) (y_{ip} - x_{jki}^{p-1} + 1), \quad (4.3)$$

$$\forall p \in P \setminus \{1\}, \forall j \in J, \forall k \in K(j) \setminus \{1\}, \forall i \in G(p)$$

$$|x_{jki}^p - x_{jki}^{p-1}| \leq y_{ip}, \forall p \in P \setminus \{1\}, \forall j \in J, \forall k \in K(j), \forall i \in G(p) \quad (4.4)$$

$$\sum_{j \in J} \sum_{k \in K(j)} x_{jki}^p = 1, \forall p \in P, \forall i \in H(p) \quad (4.5)$$

$$\sum_{i \in H(p)} x_{jki}^p \leq 1, \quad \forall p \in P, \forall j \in J, \forall k \in K(j) \quad (4.6)$$

$$\sum_{i \in H(p)} x_{jki}^p - \sum_{i \in H(p)} x_{j,k-1,i}^p \leq 0, \forall p \in P, \forall j \in J, \forall k \in K(j) \quad (4.7)$$

$$\sum_{i \in H(p)} W_i^c x_{jki}^p - \sum_{i \in H(p)} W_i^c x_{j,k-1,i}^p \leq 0, \quad \forall p \in P, \forall j \in J, \forall k \in K(j) \quad (4.8)$$

$$\sum_{k \in K(j)} \sum_{i \in H(p)} W_i^c x_{jki}^p \leq W_j^s, \forall p \in P, \forall j \in J \quad (4.9)$$

The objective function (4.1) calculates the cost of total shifted containers during the journey. Constraints (4.2) denotes that container i ($i \in G(p)$) will be shifted at port p if there are containers with port of destination p stowed under container i . Constraints (4.3) entails that, if container i ($i \in G(p)$) is shifted then containers stowed above it also have to be shifted. Containers that are not shifted and destined to port beyond p should maintain in the same slot, which is ensured by constraints (4.4). Constraints (4.5) guarantees all the containers are loaded on the container ship. Constraints (4.6) forces that each slot can only be occupied by one container. Constraints (4.7) ensures the allocated containers form stacks, which means that no containers are hanging in the air. Constraints (4.8) imposes that the heavy container are not stowed on top of light container. Constraints (4.9) requires the weight of loaded containers can not exceed the maximum weight limits of each stack.

Model F is flexible to incorporate other constraints. For example, for reefer containers, they should be placed into slots with power plug. Let R_i^c indicates whether container i is reefer or not, and R_{jk} denotes whether slots (j,k) can be

stowed reefer container or not, we have,

$$\sum_{j \in J} \sum_{k \in K(j)} x_{jki}^p R_{jk} \geq R_i^c, \quad p \in P, i \in H(p)$$

For stowing high cube containers, the maximum height limits of each stack can not be exceeded. Denoting H_i^c as the height of container i and H_j^s the maximum height limits of stack j , we have,

$$\sum_{k \in K(j)} \sum_{i \in H(p)} H_i^c x_{jki}^p \leq H_j^s, \quad p \in P, j \in J$$

In realistic stowage plan, we may have many other container placement requirements. For example, container can be required to be stowed into a specific slot. In this case, we can simply set $x_{j'k'i'}^p = 1$ to indicate container i' has to be placed in slot (j', k') .

4.2.1 Integer linear programming model

One can find that Model F is a nonlinear integer programming model. The nonlinearity stems from constraint (4.3) and (4.4). Indeed, there is no efficient general solution algorithm for solving such a nonlinear integer programming problem. Using linearization technique, we can transform the nonlinear model equivalently into a linear model. let $\eta_{jki}^p = x_{jki}^{p-1} y_{ip}$

$$\begin{cases} \eta_{jki}^p \geq x_{jki}^{p-1} + y_{ip} - 1, & p \in S, j \in J, k \in K(j), i \in G(p) \\ \eta_{jki}^p \leq x_{jki}^{p-1}, & p \in S, j \in J, k \in K(j), i \in G(p) \\ \eta_{jki}^p \leq y_{ip}, & p \in S, j \in J, k \in K(j), i \in G(p) \end{cases} \quad (4.10)$$

Constraint (4.4) can be equivalently rewritten as follows,

$$\begin{cases} y_{ip} \geq x_{jki}^p - x_{jki}^{p-1}, & \forall p \in P, \forall j \in J, k \in K(j), i \in G(p) \\ y_{ip} \geq x_{jki}^{p-1} - x_{jki}^p, & \forall p \in P, \forall j \in J, k \in K(j), i \in G(p) \end{cases} \quad (4.11)$$

Then, the nonlinear integer programming model can be transformed into integer linear programming model Γ equivalently, which is given by:

$$[\text{Model } \Gamma] \min \sum_{p \in S} \sum_{i \in G(p)} c_p y_{ip} \quad (4.12)$$

Subject to constraints (4.2), (4.5)--(4.9), (4.10)--(4.11) and,

$$\begin{aligned} \sum_{k'=1}^{k-1} \sum_{i' \in G(p)} \eta_{jk'i'}^p &\leq (k-1) (y_{ip} - x_{jki}^{p-1} + 1), \\ \forall p \in P \setminus \{1\}, \forall j \in J, \forall k \in K(j) \setminus \{1\}, \forall i \in G(p) \end{aligned} \quad (4.13)$$

Model Γ is an integer linear programming model with respect to decision variables x_{jki}^p , y_{ip} and $\eta_{jk'i'}^p$. To illustrate the correctness of the model, a small instance is given as following,

Example 1. Given the transportation matrix in Table 4.2. Assume $|P| = 5$, $|J| = 3$, $|k(j)| = 5$, $c_p = 1 (\forall p \in P)$, the result of weight of each container is shown in Figure 4.2.

Table 4.2
Transportation matrix for the port call

Source/Dest	2	3	4	5
1	3	7	2	0
2	0	0	1	2
3	0	0	6	3
4	0	0	0	6

The slot plan for each of port call generated by model Γ are shown in Figure 4.2. The total number of shifts for the problem is 3, the shifts happened at port 3, the dark gray slots show the shifted containers.

Weight: 10 POD:4			Weight: 10 POD:4			Weight: 10 POD:4	Weight: 12 POD:5			Weight: 12 POD:5	
Weight: 11 POD:3		Weight: 11 POD:3	Weight: 11 POD:3		Weight: 11 POD:3	Weight: 15 POD:4	Weight: 12 POD:5	Weight: 10 POD:4		Weight: 12 POD:5	Weight: 20 POD:5
Weight: 23 POD:3	Weight: 19 POD:2	Weight: 11 POD:3	Weight: 23 POD:3	Weight: 12 POD:4	Weight: 11 POD:3	Weight: 17 POD:4	Weight: 17 POD:5	Weight: 12 POD:4		Weight: 17 POD:5	Weight: 22 POD:5
Weight: 26 POD:3	Weight: 19 POD:2	Weight: 17 POD:3	Weight: 26 POD:3	Weight: 12 POD:5	Weight: 17 POD:3	Weight: 24 POD:4	Weight: 17 POD:5	Weight: 12 POD:4	Weight: 19 POD:5	Weight: 17 POD:5	Weight: 23 POD:5
Weight: 29 POD:3	Weight: 29 POD:2	Weight: 18 POD:4	Weight: 29 POD:3	Weight: 19 POD:5	Weight: 18 POD:4	Weight: 28 POD:4	Weight: 19 POD:5	Weight: 18 POD:4	Weight: 26 POD:5	Weight: 19 POD:5	Weight: 28 POD:5
Departing port 1			Departing port 2			Departing port 3			Departing port 4		

Figure 4.2: Optimal slot plan for the example problem upon leaving each port of call, wherein slot highlighted with dark colour denotes container at that slot was shifted at port 3, the maximum weight limits for each stack is 100, POD denotes the port of destination

4.3 Solution algorithms

The slot planning problem of a single port is NP-hard (Delgado et al., 2012b). The multi-port slot planning is even more complicated as the interaction of the container placement constraints among all the ports has to be explicitly considered. Indeed, most of the solution methods in the literature on this type of problems are heuristic algorithms. In this section, we propose two solution methods for the multi-port slot planning problem (MP-SPP) by exploiting the special model structure of this problem: dynamic slot attributes method and tabu search algorithm.

4.3.1 Definition, sets and parameters

To facilitate the description of the methods, we present notations used in the method in this section. Table 4.3 together with Table 4.1 define the sets and parameters used to describe our solution methods. Before further elaborating on the solution algorithms, we firstly introduce the concept of active empty slot,

Definition 1 (*Active empty slot*). An active empty slot means an empty slot immediately upon an occupied slot or refers to the lowest slot of the block; active empty slot ensures the container after loading is not hang in the air.

Table 4.3
Index set and constants in the solution method

$S(p)$	Set of shifted containers at port p
$NS(p)$	Set of containers including shifted containers at port p and containers whose origin port is p , $NS(p) = N(p) \cup S(p)$
$\bar{G}(p)$	Set of on board containers after discharging operations at port p , $\bar{G}(p) = G(p) \setminus S(p)$
OS	Set of active empty slot
ES	Set of occupied slot
EES_i	Set of eligible active empty slot for container i ($i \in NS(p)$), $EES_i \subseteq OS$
EOS_i	Set of eligible occupied slot for container i ($i \in NS(p)$), $EOS_i \subseteq ES$
$\bar{J}(k)$	Set of stack in which tier k could be indexed
$\bar{W}_{jk}$	The relative weight requirement of a container to be stowed at active empty slot (j, k) which means the weight of container to be loaded cannot be heavier than the weight of already loaded container underneath the slot
$\bar{W}_{jk}^s$	The weight requirement of a container to be stowed into active empty slot (j, k) corresponding to the maximum stack weight limits, which means after loading the container the total weight of the container stowed within the stack should not exceed the maximum stack weight limits
POD_{jk}	Records the port of destination of the container immediately underneath the active empty slot (j, k)

$\hat{W}_{jk}^u$	The relative weight requirement of the container to be loaded correlated to the upper slot of (j, k) which means the weight of container to be loaded should be heavier than the weight of container stowed at the upper slot
$\hat{W}_{jk}^d$	The relative weight requirement of the container to be loaded corresponding to the underneath slot of (j, k) , it denotes the weight of the container to be loaded have to be lighter than the weight of container stowed underneath the slot of (j, k)
$\hat{W}_{jk}^s$	The weight requirement of the container to be loaded corresponding to the maximum weight limits of stack j
$\hat{W}_{jk}$	Records the weight of container currently stowed at slot (j, k)
POD_{jk}	Records the port of destination of container currently stowed at slot (j, k)
W_{dm}^{sort}	Denotes the weight that rank m th of all the weight of the containers with port of destination d
W_{dk}^{tier}	Denotes the suggested total weight of containers with port of destination d for tier k
A_{id}	Equals to 1 when container i is destined for port d , otherwise 0
S_i	Binary, equals to 1 when container i ($i \in G(p)$) is shifted at port p , otherwise 0
O_i	Binary, equals to 1 when container i ($i \in H(p)$) is overstowed at port p , otherwise 0
P_{jd}	Binary, equals to 1 when at least one container in stack j is destined for port d , otherwise 0
E_j	Binary, equals to 1 when there exists containers stowed at stack j , otherwise 0

4.3.2 Slot attributes based heuristic method

In constructing solution algorithm for solving MP-SPP, we divide the original problem into $|P| - 1$ subproblems, in which a subproblem p is used to generate high-quality slot plan ($\mathbf{x}^p$) for port p with the given slot plan of previous port ($\mathbf{x}^{p-1}$) and container set $N(p)$, $\cup_{d=p+1}^N G(d)$ and $D(p)$, $\mathbf{x}^p = \{x_{jki}^p | j \in J, k \in K(j), i \in H(p)\}$. After all subproblems are constructed, the solution for MP-SPP is obtained, $\mathbf{x} = \{\mathbf{x}^p | p = 1, 2, \dots, N - 1\}$.

In each step of the method, we firstly update the set of occupied slots OS and active empty slots ES according Equations (4.14) and Equation (4.15). Note that only one container is determined to be loaded in each step. The set of active empty slot OS and the set of occupied slot ES are defined as follows:

$$\text{OS} = \left\{ (j, k) | \sum_{i \in H(p)} x_{jki}^p = 1, \forall j \in J, \forall k \in K(j) \right\} \quad (4.14)$$

$$\begin{aligned} \text{ES} = & \left\{ (j, k) | \sum_{i \in H(p)} x_{jki}^p = 0, \forall j \in J, k = k_0(j) \right\} \cup \\ & \left\{ (j, k) | \sum_{i \in H(p)} x_{jki}^p < \sum_{i \in H(p)} x_{j, k-1, i}^p, \forall j \in J, k_0(j) < k \in K(j) \right\} \end{aligned} \quad (4.15)$$

Initially, for port p , we have $x_{jki}^p = x_{jki}^{p-1}$, $\forall i \in \bar{G}(p)$, which means unshifted containers stay at the same position as it is stowed in the previous port. Meanwhile, $x_{jki}^p = 0$, $\forall i \in NS(p)$, means containers i is not stowed; $x_{jki}^p = 1$, $\forall i \in NS(p)$, means containers i is stowed into slot (j, k) .

As defined in section 4.3.1, $\bar{W}_{jk}$, POD_{jk} and $\bar{W}_{jk}^s$ represent the attributes of active empty slot (j, k) ; $\hat{W}_{jk}^u$, $\hat{W}_{jk}^d$, $\hat{W}_{jk}^s$, $\hat{\text{POD}}_{jk}$ and $\hat{W}_{jk}$ together denote the attributes of occupied slot (j, k) . In each step, the attributes of active empty slots and occupied slots are updated based on the following equations,

$$\bar{W}_{jk} = \begin{cases} W_j, & \text{if } k = k_0(j) \\ \sum_{i \in H(p)} W_i^c x_{j, k-1, i}^p, & \text{Otherwise} \end{cases} \quad \forall (j, k) \in \text{ES} \quad (4.16)$$

$$\text{POD}_{jk} = \begin{cases} \infty, \text{ if } k = k_0(j) \\ \sum_{i \in H(p)} \text{POD}_i x_{j,k-1,i}^p, \text{ Otherwise} \end{cases} \quad \forall (j, k) \in \text{ES} \quad (4.17)$$

$$\bar{W}_{jk}^s = W_j - \sum_{k' \in K(j)} \sum_{i \in H(p)} W_i^c x_{jk'i}^p, \quad \forall (j, k) \in \text{ES} \quad (4.18)$$

$$\hat{W}_{jk}^u = \begin{cases} 0, \text{ if } \sum_{i \in H(p)} x_{j,k+1,i}^p = 0 \text{ or } k = \bar{k}(j) \\ \sum_{i \in H(p)} W_i^c x_{j,k+1,i}^p, \text{ Otherwise} \end{cases} \quad \forall (j, k) \in \text{OS} \quad (4.19)$$

$$\hat{W}_{jk}^d = \begin{cases} W_j, & \text{if } k = k_0(j) \\ \sum_{i \in H(p)} W_i^c x_{j,k-1,i}^p, & \text{Otherwise} \end{cases} \quad \forall (j, k) \in \text{OS} \quad (4.20)$$

$$\hat{W}_{jk} = \sum_{i \in H(p)} W_i^c x_{jki}^p, \quad \forall (j, k) \in \text{OS} \quad (4.21)$$

$$\hat{\text{POD}}_{jk} = \sum_{i \in H(p)} \text{POD}_i^c x_{jki}^p, \quad \forall (j, k) \in \text{OS} \quad (4.22)$$

$$\hat{W}_{jk}^s = W_j - \sum_{k' \in K(j)} \sum_{i \in H(p)} W_i^c x_{jk'i}^p, \quad \forall (j, k) \in \text{OS} \quad (4.23)$$

Equation (4.16)–(4.18) present the expressions for updating the attributes of active empty slot in each step. It should be noted when the slot is the lowest slot of the stack, there is no relative weight requirement for the lowest slot; however, it should satisfy the maximum stack weight limits, and $\text{POD}_{jk} = \infty$ denotes there is no port of destination requirement for a container to be stowed into slot (j, k) . Equation (4.19)–(4.23) are used to update the attributes of occupied slot in each step, which is similar to the attributes definition of active empty slot except that $\hat{W}_{jk}^u$ is set to 0 when the slot (j, k) is the highest slot of stack j or the upper slot is active empty slot. (i.e., that there is no container weight constraints of upper slots.)

After the set and attributes of occupied slots OS and active empty slots ES are updated, we determine the set of eligible active empty slots EES and eligible occupied slot EOS. A container i ($i \in NS(p)$) can be stowed into slot (j, k) ($(j, k) \in \text{ES}$) only if it satisfies the attributes of slot (j, k) , or be placed into (j', k')

$((j', k') \in \text{OS})$ where container i' is currently stowed if it fulfills the attributes of slot (j', k') . Note that when container i is placed into occupied slot (j', k') , container i' will become unassigned and will be re-stowed later. In each step, the EES and EOS for container i is updated by Equation (4.24) and (4.25).

$$\text{EES}_i = \{(j, k) | (j, k) \in \text{ES} \text{ and } W_i^c \leq \bar{W}_{jk}, W_i^c \leq \bar{W}_{jk}^s\} \quad (4.24)$$

$$\text{EOS}_i = \{(j, k) | (j, k) \in \text{OS} \text{ and } W_i^c \leq W_{jk}^d, W_i^c \geq W_{jk}^u, W_i^c \leq W_{jk}^s, W_i^c > \hat{W}_{jk}\} \quad (4.25)$$

Equation (4.24) defines the set of eligible active empty slot for the container i which is to be loaded, and Equation (4.25) defines the set of eligible occupied slot for the container to be loaded. Note $W_i^c > \hat{W}_{jk}$ in Equation (4.25) means a lighter container should be swapped out.

Slot attributes based heuristic method (SAM) comprises three procedures: sorting procedure aimed to determine container loading order, loading procedure designed to select most suitable slot to stow the container, and swapping procedure used to handle exceptions occurred in loading procedure, i.e., $\text{EES}_i = \emptyset$. In the sorting procedure, container set $NS(p)$ are arranged according to their port of destination and container weight in descending order, i.e., the farther and heavier container is loaded prior to closer and lighter container. In loading procedure, we determine the most suitable slot $(j, k) \in \text{EES}_i$ for container i according to three conditions: (1) choose (j, k) such that $\bar{W}_{jk}^s = \max\{\bar{W}_{j'k'}^s | (j', k') \in \text{EES}_i, \text{ and } \text{POD}_{j'k'} \geq \text{POD}_i^c\}$, which guarantees container i is not overstowed and smaller number of stack is used; (2) if condition 1 is not satisfied, choose (j, k) such that $\bar{W}_{jk}^s = \max\{\bar{W}_{j'k'}^s | (j', k') \in \text{EES}_i, \text{ and } \text{POD}_{j'k'} = \infty\}$, which also ensures container i is not overstowed but through using a new stack; (3) if condition 1 and condition 2 are not satisfied, choose (j, k) such that $\bar{W}_{jk}^s = \max\{\bar{W}_{j'k'}^s | (j', k') \in \text{EES}_i\}$ which ensures container i can be stowed into a slot successfully. However, loading procedure cannot always assure that there exists eligible active empty slot for container i . Hence, when $\text{EES}_i = \emptyset$, we use swapping procedure to stow the container i . Note that swapping procedure will cause an already

loaded container i' transformed into the state of unloaded, which needs to be restowed later through loading procedure. In swapping procedure, the best slot $(j, k) \in \text{EOS}_i$ for container i to be placed into is determined through the following condition: (1) choose the slot (j, k) such that $\hat{W}_{jk} = \min\{\hat{W}_{j'k'} | (j', k') \in \text{EOS}_i \text{ and } \text{POD}_i^c \leq \hat{\text{POD}}_{j'k'}, \text{ meanwhile } (j', k') \text{ the highest slot}\}$; (2) if condition 1 is not satisfied, choose slot (j, k) such that $\hat{W}_{jk} = \min\{\hat{W}_{j'k'} | (j', k') \in \text{EOS}_i \text{ and } (j', k') \text{ the highest slot}\}$. The three procedures are repeated until all the containers of port p are loaded.

4.3.3 Tabu search algorithm

Tabu search algorithm (TSA) is a meta heuristic method which iteratively explores the problem solutions by making moves from a current solution $\mathbf{x}$ to another solution $\mathbf{x}'$ within the neighborhood of $\mathbf{x}$ (Glover et al., 1993). The moves are performed with the goal of minimizing or maximizing an evaluation function $F(\mathbf{x})$. In our problem, Tabu search is employed to generate better solution for each subproblem p , which can be viewed as the improvement phase of the solution obtained by SAM. A better solution for each subproblem p is evaluated in the following two aspects: (1) the solution $\mathbf{x}^p$ should be most consistent with the solution $\mathbf{x}^{p-1}$ which reduces the unnecessary shifts caused by swapping procedure in SAM; (2) the solution leaves more operating space for generating slot plan of subsequent port rotations so as to reduce the possibility of container overstorage or shifts in future ports, which can be achieved by minimizing container overflows of current plan, avoiding stacks with stowed containers having many different ports of destination and keeping stack empty if possible (Delgado et al., 2012b). Meanwhile, we also incorporate the objective that the heavier container should be placed into lower tier. The evaluation function for

the solution of subproblem p is expressed as follows:

$$\begin{aligned}
F(\mathbf{x}^p) = & 100 \sum_{i \in G(p)} S_i + 100 \sum_{i \in H(p)} O_i + 20 \sum_{j \in J} \sum_{d > p} P_{jd} + 10 \sum_{j \in J} E_j + \\
& 50 \sum_{k=1}^{\bar{k}} \sum_{d > p} (W_{dk}^{\text{tier}} - \sum_{j \in \bar{J}(k)} \sum_{i \in D(d) \cap NS(p)} W_i^c x_{jki}^p) / \sum_{i \in NS(p)} W_i^c
\end{aligned} \tag{4.26}$$

Evaluation function (4.26) is a weighted sum of five objectives: the first part calculates the cost due to the number of container shifts at port p with the solution $\mathbf{x}^p$, the second part determines the cost because of the number of container over-stowage, the third part computes the cost caused by the number of port of destination within a stack, the fourth part describes the cost of used stack, and the last part evaluates the cost incurred by the deviation that heavier containers are not placed into lower tier, $W_{dk}^{\text{tier}} = \sum_{m=a}^b W_{dm}^{\text{sort}}$ where $a = \sum_{j \in J} \sum_{n=1}^{k-1} \sum_{i \in NS(p)} x_{jni}^p$ and $b = \sum_{j \in J} \sum_{m=1}^k \sum_{i \in NS(p)} x_{jmi}^p$. We adopt the cost coefficient from [Delgado et al. \(2012b\)](#) which was suggested by their industry cooperator. For the cost coefficient of shifted container S_i and last objective, we test through several computational experiments and set them to be 100 and 50 respectively. Evaluation function can be determined by the following equation when given the solution $\mathbf{x}^p$,

$$x_{jki}^{p-1} = 1 \text{ and } \sum_{k' < k} \sum_{i' \in D(p)} x_{jk'i'}^{p-1} \geq 1, \quad \forall j \in J, \forall k \in K(j) \Rightarrow S_i = 1, \quad \forall i \in G(p) \tag{4.27a}$$

$$x_{jki}^{p-1} \neq x_{jki}^p, \quad \forall j \in J, \forall k \in K(j) \Rightarrow S_i = 1, \quad \forall i \in G(p) \tag{4.27b}$$

$$x_{jki}^{p-1} = 1 \text{ and } x_{jk'i'}^{p-1} \neq x_{jk'i'}^p, \forall j \in J, \forall k' < k \in K(j), \forall i' \in G(p) \Rightarrow S_i = 1, \quad \forall i \in G(p) \tag{4.27c}$$

$$A_{id} x_{jki}^p = 1 \text{ and } \sum_{k' < k} \sum_{d' < d} \sum_{i' \in H(p)} A_{i'd'} x_{jk'i'}^p \geq 1 \Rightarrow O_i = 1, \quad \forall i \in H(p) \tag{4.28}$$

$$\sum_{k \in K(j)} \sum_{i \in D(d)} x_{jki}^p \geq 1 \Rightarrow P_{jd} = 1, \quad \forall j \in J, \forall d \in P \text{ and } d > p \tag{4.29}$$

$$\sum_{k \in K(j)} \sum_{i \in H(p)} x_{jki}^{p-1} \geq 1 \Rightarrow E_j = 1, \quad \forall j \in J \tag{4.30}$$

Condition (4.27a), (4.27b) and (4.27c) are three sufficient conditions to determine

whether container $i \in G(p)$ is shifted or not. Condition (4.28) specifies that, if one container with port of destination d is stowed on a container with port of destination d' ($d' < d$), then it is overstowed. Condition (4.29) describes whether there are containers with port of destination d that are stowed in stack j or not. A stack is used or not is determined by Condition (4.30).

Define $N(\mathbf{x}^p)$ as the neighborhood of solution $\mathbf{x}^p$. It is the set of solution satisfying constraints (4.5)-(4.9) and obtained by changing the position of container $i \in NS(p)$ to an eligible active empty slot or swapping pairs of containers between pairs of compatible slots. As was done in Monaco et al. (2014), the attributes identifying a solution $\mathbf{x}^p$ are added to the set $A(x) = \{(j, k, i) \mid \text{container } i \text{ is allocated to slot position } (j, k)\}$. The set $A(x)$ is used to record the moves between solutions and perform as tabu during iterative process.

Starting with the solution $\mathbf{x}^p$ obtained by SAM as the initial solution, we explore the neighborhood of $\mathbf{x}^p$ within $N(\mathbf{x}^p)$. Each neighboring solution is evaluated through $F(\mathbf{x}^p)$ defined by Equation (4.26). At a given iteration, an attribute (j, k, i) is forbidden to add to $A(x)$ for next $\bar{t}$ iterations (tabu tenure) if it is removed from $A(x)$ at that iteration. We denote a solution as tabu if the attributes describing the solution contains at least one tabu attribute. However, a tabu solution can be accepted when it represents an improving solution which is the aspiration criterion. Meanwhile, we diversify the search through penalizing a non-improving solution by the frequencies of the attributes added to identify that solution. The frequency of an attribute is defined as the number of the times the attribute has been used to identify a solution. For each of these possible moves, we compute the evaluation function defined in Equation (4.26) and select the one yielding the minimum value.

4.4 Numerical experiments

In this section, we test the performance of our model formulation and solution algorithms presented in this paper by implementing three numerical experiments with different problem settings. The experiments are set up to examine the following aspects on the complexity of container stowage planning problem and the applicability of the proposed algorithms: the number of port rotations, the space utilization of the block position and the attributes of container weight distribution. The space utilization is the number of TEUs on aboard when departing a port divided by the total TEU capacity of the block position. Without loss of generality, we take the configuration of the block position as rectangular in our experiment. The characteristics of each instance associated with the three experiments are shown in Tables 4.4, 4.5 and 4.6: the configuration of block position within a container ship in terms of number of stack $|J|$ and number of tier for each stack $|K|$, the number of port rotations $|P|$, the minimum and maximum slot utilization within the block position after stowing containers at each port of call, the total number of containers $|I|$, the weight distribution of the containers.

Table 4.4
Characteristic of experiment 1 instances

Instance	$ J \times K $	$ P $	(min,max)	$ I $
E1I1	6×4	4	(0.7,0.8)	39
E1I2	6×4	5	(0.7,0.8)	46
E1I3	6×4	6	(0.7,0.8)	47
E1I4	8×6	4	(0.7,0.8)	79
E1I5	8×6	5	(0.7,0.8)	85
E1I6	8×6	6	(0.7,0.8)	95

Table 4.5
Characteristic of Experiment 2 instances

Instance	$ J \times K $	$ P $	(min,max)	$ I $
E2I1	6×4	4	(0.8,0.9)	47
E2I2	6×4	5	(0.8,0.9)	53
E2I3	6×4	6	(0.8,0.9)	56
E2I4	8×6	4	(0.8,0.9)	91
E2I5	8×6	5	(0.8,0.9)	100
E2I6	8×6	6	(0.8,0.9)	111

Table 4.6
Characteristic of Experiment 3 instances

Instance	$ J \times K $	$ P $	(min,max)	$ I $
E3I1	6×4	4	(0.9,1.0)	53
E3I2	6×4	5	(0.9,1.0)	56
E3I3	6×4	6	(0.9,1.0)	57
E3I4	8×6	4	(0.9,1.0)	105
E3I5	8×6	5	(0.9,1.0)	108
E3I6	8×6	6	(0.9,1.0)	115

We consider three scenarios of container weight distribution: Scenario A, B and C. In scenario A, the weight of each container is treated identically, hence the container weight constraint in the model is relaxed, resulting into a simplified problem. In scenario B, the containers are classified as light containers, medium containers and heavy containers, the weight of the container within each class is assumed to be identical and the proportion of each class container in our experiment is set as 0.1, 0.5 and 0.4 respectively. In scenario C, the container is treated individually, of which the weight is generated following normal distribution in the experiments. The transportation matrix for the instances in each experiment is the type of mixed matrices as described in [Avriel et al. \(1998\)](#), in which the values of the elements are determined in randomly order and satisfy the conditions such that (1) $T_{ij} \leq 0.3 \times Ca$, Ca is the capacity of the block position; (2)

$\min \times Ca \leq \sum_{i=1}^k \sum_{j=k+1}^N T_{ij} \leq \max \times Ca, \forall k \in \{1, 2, \dots, n-1\}$ which guarantees a feasible transportation matrix with mixed matrices and ensures the space utilization fulfill the predetermined utilization range.

The model Γ and solution algorithms have been implemented using a computer with 8 gigabytes of RAM, Windows 10 Professional operating system and Intel Core i5 processors with 2.6 gigahertz cores. We have let CPLEX running for at most two hours adopting default settings in each instance of the three experiments, the parameters for solution algorithm TSA have been chosen through several test, the parameters for TSA is set as follows: maximum number of iteration $h_{max} = 300$; tabu prohibition $h = 10$; diversification penalty $\gamma = 0.005$.

4.4.1 Analysis of results

The computational results obtained for all the instances of the three experiments are summarized in Table 4.7, 4.8 and 4.9. The solution of number of shifts and the computational time are reported for each of the three solution algorithms, i.e., SAM, TSA and the commercial software CPLEX. As CPLEX does not always return an optimal solution within predetermined time limit for all the instances of the experiments, we also report Gap returned by CPLEX.

Table 4.7

Computation results on instances of experiment 1, ``S" represents the container weight scenarios

Instance	S	CPLEX			SAM		TSA	
		Shifts	Time(s)	Gap%	Shifts	Time(s)	Shifts	Time(s)
E1I1	A	0	0.4	0.0	0	0.0	0	5.9
	B	0	1.3	0.0	0	0.0	0	5.8
	C	0	1.4	0.0	0	0.0	0	5.8
E1I2	A	0	1.3	0.0	0	0.1	0	9.4
	B	0	5.0	0.0	0	0.1	0	9.4
	C	0	8.9	0.0	0	0.1	0	9.5
E1I3	A	0	3.5	0.0	0	0.1	0	10.5
	B	0	4.3	0.0	2	0.1	0	10.4
	C	0	3.4	0.0	7	0.1	0	13.4
E1I4	A	0	2.8	0.0	0	0.3	0	48.3
	B	0	9.0	0.0	0	0.3	0	48.2
	C	0	394	0.0	0	0.3	0	48.1
E1I5	A	0	45.6	0.0	0	0.6	0	75.3
	B	0	61.6	0.0	0	0.6	0	75.8
	C	0	6359	0.0	0	0.6	0	74.5
E1I6	A	0	1145	0.0	0	0.9	0	93.0
	B	8	7200	100	1	0.9	0	89.1
	C	65	7200	100	4	0.9	2	114.1

Table 4.8

Computation results on instances of experiment 2, "--" means no corresponding value were obtained, "S" represents the container weight scenarios

Instance	S	CPLEX			SAM		TSA	
		Shifts	Time(s)	Gap%	Shifts	Time(s)	Shifts	Time(s)
E2I1	A	0	0.6	0.0	0	0.0	0	6.1
	B	0	0.8	0.0	0	0.0	0	6.1
	C	0	2.5	0.0	0	0.0	0	6.2
E2I2	A	0	1.3	0.0	0	0.1	0	7.6
	B	0	4.6	0.0	0	0.1	0	7.7
	C	0	9.6	0.0	0	0.1	0	7.8
E2I3	A	0	4.7	0.0	0	0.1	0	8.9
	B	0	560.7	0.0	5	0.1	1	12.2
	C	1	7200	0.0	10	0.1	2	9.7
E2I4	A	0	5.5	0.0	0	0.5	0	54.1
	B	0	33.6	0.0	1	0.5	0	54.8
	C	0	803.3	0.0	4	0.5	0	54.3
E2I5	A	0	393.6	0.0	0	0.7	0	58.1
	B	4	7200	100	0	0.7	0	58.1
	C	--	--	--	3	0.7	1	140.0
E2I6	A	0	2850	0.0	0	1.0	0	76.1
	B	17	7200	100	2	1.0	1	78.5
	C	86	7200	100	4	1.0	3	204.2

Table 4.9

Computation results on instances of experiment 3, "--" means no corresponding value were obtained, "S" represents the container weight scenarios

Instance	S	CPLEX			SAM		TSA	
		Shifts	Time(s)	Gap%	Shifts	Time(s)	Shifts	Time(s)
E3I1	A	0	0.6	0.0	0	0.1	0	7.6
	B	0	1.4	0.0	0	0.1	0	7.7
	C	0	52.3	0.0	3	0.1	0	50.1
E3I2	A	0	6.1	0.0	0	0.1	0	9.4
	B	0	4.4	0.0	1	0.1	0	9.3
	C	0	305.9	0.0	4	0.1	2	74.2
E3I3	A	0	6.8	0.0	1	0.1	0	10.5
	B	0	1306	0.0	6	0.1	1	78.8
	C	7	7200	100	14	0.1	3	94.1
E3I4	A	0	4.7	0.0	0	0.6	0	62.6
	B	0	148.0	0.0	0	0.6	0	62.5
	C	0	2397	0.0	0	0.6	0	62.0
E3I5	A	0	885.9	0.0	0	1.0	0	80.1
	B	19	7200	100	3	1.0	0	561.1
	C	46	7200	100	3	1.0	0	502.4
E3I6	A	2	7200	100	0	1.2	0	113.1
	B	93	7200	100	2	1.2	1	195.6
	C	--	--	--	6	1.2	2	820.2

From the results demonstrated in Table 4.7, 4.8 and 4.9, we have the following observations with respect to the impact on the complexity of the problem: (1) The space utilization at each port of call directly affects the computational complexity of the problem. One can observe that the computation time of CPLEX on the instances among each experiment with same settings (same scenario, port rotations and block position configuration) increases substantially, for example the computation time for instances E1I5-B, E2I5-B and E3I5-B are 61.6s, 7200s and 7200s respectively; (2) Higher number of port rotations also increase the computational complexity of the problem greatly. In each experiment for the instances with same scenario and block position configuration like E3I1-A, E3I2-A and E3I3-A, CPLEX also returns monotonically increasing computation time

with more number of port rotations. This observation becomes more prominent when the problem size is large like E3I4-A, E3I5-A and E3I6-A; (3) If we strictly consider container weight requirements when stowing containers at each port of call, the computation time of CPLEX will be rather long to find optimal solution especially for large-size instance, which can be observed through the computation time of CPLEX for different scenarios of each instance in each experiment. Indeed, CPLEX solves most of the instances in each experiment at optimality but two of the 18 instances in experiment 1, four of the 18 instances in experiment 2 and six of the 18 instances in experiment 3; one instances in experiment 1 and one instances in experiment 2 CPLEX have failed to find a feasible solution within the predetermined time limit.

As a comparison, the overall performance of SAM and TSA algorithm are satisfactory both in terms of solution quality and computation time. Figure 4.3, 4.4 and 4.5 show the comparison of the performance between SAM and TSA in our experiments in terms of number of shifts. Hereby, x-axis of the figure labels the instance index wherein A, B and C represent respective container weight scenario in the experiment, I1, I2 and I3 represent the configuration of block position of 6×4 and I4, I5 and I6 of 8×6 . Actually, SAM works very efficiently with small computational time. The highest running time of SAM is less than two seconds for the instances in our experiment. However, in some conditions it returns poor solution. We can observe that SAM exhibits best performance for those instances with scenario A, and good performance for instances with scenario B but poor performance for scenario C. On the contrary, TSA shows better performance in generating good solution compared with SAM. Indeed, for most of the cases, TSA returns optimal solution, even for large and complicated instances in which CPLEX performs poorly like E3I5-C and E3I6-C. However, TSA requires higher computational time as compared with SAM but still is efficient compared with CPLEX.

Figure 4.3: Comparison of the performance of SAM and TSA for experiment 1

Figure 4.4: Comparison of the performance of SAM and TSA for experiment 2

Figure 4.5: Comparison of the performance of SAM and TSA for experiment 3

4.5 Conclusion

In this chapter, we investigate the multi-port slot planning problem, which focuses on assigning specific slots to containers under a set of placement restrictions. An optimal slot plan considering the interactions among all the ports of calls would minimize the costs induced by the shifts of stowage. To describe the MP-SPP problem, we have proposed an integer nonlinear programming model aiming to minimize the total container shifts during port rotations. Our model is flexible to incorporate more practical constraints. Generally, MP-SPP problem is NP-hard. We developed a dynamic slot attributes heuristic method and a tabu search method for solving the problem. The model and solution methods have been tested through a set of computation experiments, which suggests that SAM is very efficient ensuring high solution quality and less computational time for simple case with no strict container weight constraints. But for complicated case in which more number of port rotations are involved, the space utilization of the block position is higher and container weight distribution is more diverse, the solution obtained by CPLEX and SAM method is poor within an acceptable

computation time, but tabu search algorithm can obtain near-optimal solution in the acceptable computation time.

Chapter 5

MODELING AND SOLVING QUAY CRANE SCHEDULING PROBLEM CONSIDERING DOUBLE CYCLING

5.1 Introduction

Quay side operations, as the core problems in a container terminal, can be roughly classified into three subproblems: the berth allocation problem (BAP), the quay crane assignment problem (QCAP) and the quay crane scheduling problem (QCSP). The BAP aims to determine the berthing time and berth position in container terminal for container ships within the planning horizon. The QCAP assigns quay cranes to container ships considering the availability and accessibility of the quay cranes. Given the berth and quay cranes allocated to a container ship, the QCSP seeks to assign the loading and unloading tasks to the particular quay crane and determine the schedule for the quay crane to process the loading and unloading tasks. The three interrelated problems comprise of a complex operation planning problem at quay side. One can refer to [Vis & De Koster \(2003\)](#) and [Stahlbock & Voß \(2007\)](#) for a comprehensive overview of quay side operations and sea side operations.

In this paper, we focus on tackling QCSP with incorporation of double cycling strategies. QCSP is to find a complete schedule for the QCs working on a container ship, aiming to minimize the container ship make-span. The technique of performing loading and unloading simultaneously in a quay crane cycle is called double cycling. The double cycling technique, enabling Quay cranes (QCs) to carry out unloading after loading activity or loading after unloading activity in each cycle, indeed converts the idle moves of QCs with only single cycling into productive moves. Normally, quay cranes will be firstly deployed to unload the containers from container ship. After the unloading operations, the quay cranes will then proceed to load containers onto container ship. This conventional operation can avoid unloading activity blocking loading activity or vice versa. However, scheduling quay cranes in this approach is less effective than that in the approach of quay crane (QC) double cycling which can reduce the ineffective idle spreader moves substantially ([Goodchild & Daganzo, 2006, 2007](#)). We refer to loading a container or unloading a container as a task in our problem. Performing loading task after unloading task or unloading task after loading task is more efficient than performing loading task after loading task or unloading task after unloading task. In this problem, each task must be assigned to a quay crane and the tasks processing sequence by quay cranes should be determined so as to minimize the make-span of container ship at container terminals. QCs are not allowed to cross each other and required to maintain a safety margin among each other. The processing precedence among tasks, derived by the relative task location on container ship and operation type, needs to be guaranteed in the problem. Meanwhile, tasks located in the bay within the safety margin of a quay crane cannot be handled by another quay crane simultaneously.

The main contributions of this chapter are three folds. First, a mixed integer programming (MIP) model for the QCSP considering double cycling strategies is developed. QC double cycling problem was firstly introduced by [Goodchild](#)

& Daganzo (2006), and thereafter studied by many researchers (Zhang & Kim, 2009; Lee et al., 2015; Liu et al., 2015; Ku & Arthanari, 2016). However, most of existing studies focused on double cycling with single QC, i.e., only one single quay crane performs loading and unloading simultaneously on a given bay of container ship. In this case, the interaction among quay cranes are not considered and the efficiency of double cycling is underestimated. Our model in this study incorporates QCs double cycling within the context of the quay crane scheduling problem, in which crane interference, safety margin between different QCs, precedence relationship among tasks and non-simultaneous handling among tasks all considered in a much more realistic and complicated problem formulation. Second, we propose a logic based decomposition (LBD) method to solve the model. Most of the solution algorithms for QCSP in previous literatures are based on meta-heuristic method, e.g., Sammarra et al. (2007) proposed a tabu search heuristic method, Bierwirth & Meisel (2009) presented unidirectional scheduling (UDS) heuristic method for QCSP, Al-Dhaheri et al. (2016b) applied a simulation-based Genetic Algorithm to construct QC schedule. The LBD method in our paper can guarantee global optimal solution, meanwhile reduce computation time substantially. Indeed, the proposed LBD method is also applicable for general QCSP. Third, a task insertion heuristic (TIH) method is developed to obtain a sufficiently good solution with even less computational time, in which a disjunctive graph model is adopted to build the algorithm.

The remainder of this chapter is organized as follows: In section 5.2, a MIP model is formulated for the QCSP considering double cycling. Section 5.3 presents a logic based decomposition method to solve the MIP model. An insertion heuristic based solution method is also proposed, and presented in section 5.4. In section 5.5, we implement the numerical studies to test the performance of the proposed decomposition method and the insertion heuristic method in comparison with MIP model solved by CPLEX. Lastly in section 5.6, we make a conclusion for the research.

5.2 Notation and model formulation

In this section, we provide a description of the problem and present a MIP model formulation. For ease of reference, the notations of the sets, parameters and the decision variables used in the formulation are listed in Table 5.1.

Table 5.1
Sets, parameters and variables in the MIP model

Parameters	
Γ	Set of quay crane deployed to unload and load a container ship
Ω	Set of tasks involved in loading and unloading a container ship
Ω^0	Set of tasks involved in loading and unloading a container ship with a virtual task 0 which is the start task of each quay crane
Ω^T	Set of tasks involved in loading and unloading a container ship with a virtual task T which is the end task of each quay crane
Φ	Set of task pairs that are forbidden to be processed simultaneously
Υ	Set of combinations of jobs and quay cranes that may potentially cause quay crane interference
τ	The processing time of quay crane to handle a task
$\bar{t}$	The quay crane travel time between two adjacent bay
a	The spreader moving time from sea side to the quay side or vice versa
l_i	Bay location index of task i
ξ_i	Equals 1 if task i is a unloading task, 0 otherwise
r_q	Ready time for quay crane q to start processing tasks
s_{iq}^0	Initial set up time for quay crane q to handle task i

$\bar{L}_q$	The lower bay number that can be processed by quay crane q
$\bar{U}_q$	The upper bay number that can be processed by quay crane q
M	A large positive number
π_{ij}	Equals 1 if task i must be completed before the start of task j , 0 otherwise
$\Delta_{jj'}^{qq'}$	Minimum time-span between the processing of jobs j and j' on quay cranes if they were performed on quay crane q and q' , respectively
Variables	
C	Make-span of a vessel
C_i	Completion time of task i by quay crane
Y_{ij}	Equals 1 if task i is completed before the start of task j , otherwise 0
X_{ijq}	Equals 1 if quay crane q process task j after the completion of task i , 0 otherwise

5.2.1 Model formulation

It is assumed the spreader of quay crane remains intact when quay crane moves along the rail. Hence, the set-up time of a quay crane to process two consecutive tasks i and j , denoted by s_{ij} , depends on the bay location and the operation type of the tasks i and j . It can be expressed as

$$s_{ij} = a|\xi_i + \xi_j - 1| + |l_i - l_j|\bar{t} \quad (5.1)$$

Equation (5.1) indicates that, when double cycling is applied between two consecutive tasks, the idle move of quay crane is converted to productive move.

We assume that the assigned quay cranes to a container ship are not allowed to leave the ship bay area before all the tasks are finished. Because of the non-cross constraint and the safety margin between quay cranes, the feasible working areas for a quay crane is limited. For example, in Figure 5.1, tasks in bay 1 and bay 2 cannot be processed by crane q2, and tasks in bay 7 and bay 8 can only be processed by crane q3, tasks in bay 3 and bay 4 can be processed by either crane q1 or q2, tasks in bay 5 and bay 6 can be processed by either crane q2 or q3. To enforce the working areas of the quay crane, we have constraints (5.2) to (5.3) to impose that tasks can be handled by proper quay cranes,

$$\sum_{j \in \Omega^T} \sum_{q \in \Gamma} X_{ijq} \bar{L}_q \leq l_i, \quad \forall i \in \Omega \quad (5.2)$$

$$\sum_{j \in \Omega^T} \sum_{q \in \Gamma} X_{ijq} \bar{U}_q \geq l_i, \quad \forall i \in \Omega \quad (5.3)$$

Figure 5.1: Feasible work areas for quay cranes with safety margin set to 1 bay

For the movement of quay cranes, the collision must be avoided and the safety margin between quay cranes should be ensured. We refer to the formulation of Bierwirth & Meisel (2009) to model the safety margin and non-cross constraint. To guarantee non-cross between quay cranes and avoid violations of the safety margin, quay cranes must maintain a minimum time span when processing two tasks, which is denoted by $\Delta_{ij}^{qq'}$,

$$\Delta_{ij}^{qq'} = \begin{cases} (l_i - l_j + \delta_{qq'}) \bar{t}, & \text{if } q < q', i \neq j, l_i > l_j - \delta_{qq'} \\ (l_j - l_i + \delta_{qq'}) \bar{t}, & \text{if } q > q', i \neq j, l_i < l_j + \delta_{qq'} \\ 0, & \text{otherwise} \end{cases} \quad (5.4)$$

In the formulation (5.4), $\delta_{qq'} = (\delta + 1) |q - q'|$, wherein δ is the minimum safety margin between adjacent quay cranes. The set of all combinations of task and

quay cranes that potentially lead to crane interference is denoted by Υ , as was defined in Bierwirth & Meisel (2009).

$$\Upsilon = \left\{ (i, j, q, q') \in \Omega^2 \times \Gamma^2 \mid (i < j) \wedge (\Delta_{ij}^{qq'} > 0) \right\} \quad (5.5)$$

The QCSP considering double cycling strategies can be formulated into the following mixed integer programming model :

$$\min z = C \quad (5.6)$$

$$\text{s.t } C \geq C_i, \forall i \in \Omega \quad (5.7)$$

$$C_i + s_{ij} + \tau - C_j \leq M(1 - X_{ijq}), \forall i, j \in \Omega, \forall q \in \Gamma \quad (5.8)$$

$$r_q + s_{jq}^0 + \tau - C_j \leq M(1 - X_{0jq}), \forall j \in \Omega, \forall q \in \Gamma \quad (5.9)$$

$$C_i + \tau - C_j \leq M(1 - Y_{ij}), \forall i, j \in \Omega \quad (5.10)$$

$$C_j - \tau - C_i \leq MY_{ij}, \forall i, j \in \Omega \quad (5.11)$$

$$C_j + \Delta_{jj'}^{qq'} + \tau - C_{j'} \leq M \left(3 - Y_{jj'} - \sum_{i \in \Omega^0} X_{ijq} - \sum_{i \in \Omega^0} X_{ij'q'} \right), \quad (5.12)$$

$$\forall (j, j', q, q') \in \Upsilon$$

$$C_{j'} + \Delta_{jj'}^{qq'} + \tau - C_j \leq M \left(3 - Y_{j'j} - \sum_{i \in \Omega^0} X_{ijq} - \sum_{i \in \Omega^0} X_{ij'q'} \right), \quad (5.13)$$

$$\forall (j, j', q, q') \in \Upsilon$$

$$Y_{ij} \geq \pi_{ij}, \forall i, j \in \Omega \quad (5.14)$$

$$Y_{ij} + Y_{ji} = 1, \forall (i, j) \in \Phi \quad (5.15)$$

$$\sum_{j \in \Omega^T} \sum_{q \in \Gamma} X_{ijq} \bar{L}_q \leq l_i, \forall i \in \Omega \quad (5.16)$$

$$\sum_{j \in \Omega^T} \sum_{q \in \Gamma} X_{ijq} \bar{U}_q \geq l_i, \forall i \in \Omega \quad (5.17)$$

$$\sum_{i \in \Omega^T} X_{0iq} = 1, \forall q \in \Gamma \quad (5.18)$$

$$\sum_{i \in \Omega^0} X_{iTq} = 1, \forall q \in \Gamma \quad (5.19)$$

$$\sum_{q \in \Gamma} \sum_{j \in \Omega^T} X_{ijq} = 1, \forall i \in \Omega \quad (5.20)$$

$$\sum_{q \in \Gamma} \sum_{j \in \Omega^0} X_{jiq} = 1, \forall i \in \Omega \quad (5.21)$$

$$\sum_{j \in \Omega^0} X_{jiq} - \sum_{j \in \Omega^T} X_{ijq} = 0, \forall i \in \Omega, \forall q \in \Gamma \quad (5.22)$$

$$X_{ijq} + X_{jiq} \leq 1, \forall i, j \in \Omega, \forall q \in \Gamma \quad (5.23)$$

$$\sum_{i \in \Omega^0} X_{ijq} + \sum_{i \in \Omega^0} X_{ij'q'} \leq 1 + Y_{jj'} + Y_{j'j}, \forall (j, j', q, q') \in \Gamma \quad (5.24)$$

$$X_{ijq}, Y_{ij} \in \{0, 1\}, \forall i, j \in \Omega, \forall q \in \Gamma \quad (5.25)$$

$$C_i \geq 0, \forall i \in \Omega \quad (5.26)$$

The objective of the problem is to minimize the make-span of discharging and loading a container ship. The make-span is the completion time of the last task, which is determined by constraint (5.7). Constraints (5.8) and (5.9) define the relationship between the completion time of task i and task j if they are handled by crane q . Constraints (5.10) and (5.11) define the relationship between decision variables C_j and Y_{ij} . Constraints (5.12) and (5.13) guarantee the minimum safety distance are satisfied if quay crane q process j and q' process j' when $(j, j', q, q') \in \Upsilon$. Constraint (5.14) ensures task i is processed before task j if i and j are precedence constrained. Constraint (5.15) guarantees task i and task j are not processed simultaneously if task pair (i, j) belongs to set Φ . Constraints (5.16)-(5.17) define the tasks that can be processed by a quay crane. Constraints (5.18)-(5.24) ensure the quay cranes operation sequence. Constraint (5.18) and (5.19) determine the first task and the last task handled by quay crane q , respectively. Constraint (5.20) and (5.21) indicate that the tasks are assigned to one quay crane. Constraint (5.22) imposes task sequence continuity. Constraint (5.23) guarantees the unidirectional of task sequence. Constraint (5.12), (5.13) and (5.24) ensure that the interference between quay cranes are avoided. Constraint (5.24) entails that, for each $(j, j', q, q') \in \Upsilon$, task j and j' are not processed simultaneously if they are assigned to quay crane q and q' , respectively. Constraints (5.25) and (5.26) define the domain of the decision variables.

5.2.2 Graph representation of feasible solution

The feasible solution of the problem can be represented by a directed acyclic graph (DAG) (a cycle in the directed graph means the completion time of the tasks are mutually dependent on each other, hence a feasible solution contains no cycle in the represented directed graph). Take Figure 5.2 as an example, each node in the graph denotes a task (0 and T are two virtual tasks, 0 is the start task of all quay cranes, T is the end task of all quay cranes). In the figure, the task process sequences by a quay crane, the task pairs that are forbidden to process simultaneously and the precedence constrained task pairs are denoted by different types of arcs as shown in the figure. It is observed that the completion time of a task depends on its inflow arcs; for example, the completion time of task 3 depends on the completion time of the tasks corresponding to its inflow arcs, i.e., task 5 and task 1.

Figure 5.2: A graph representation of feasible solution of the model

The considered problem can be viewed as a scheduling problem on parallel uniform machines with precedence constraints and non-preemption processing, which are proven NP-hard in strong sense by [Pinedo \(2016\)](#). Because of the computational intractability, one can hardly obtain the exact optimal solutions in polynomial time using general solution algorithm like branch and bound, simplex algorithm. Thus, in the following section, we present two solution methods

for the problem: one is logic based decomposition method, another is based on the insertion heuristic method.

5.3 An enhanced combinatorial Benders cuts method

[Codato & Fischetti \(2006\)](#) proposed a combinatorial Benders cuts decomposition method for MILP formulation, in which the cuts generated do not depend on the big-M formulation. The method can produce the linear programming relaxation of the master problem considerably tighter than that associated with the origin MILP formulation. The computational results in their paper shows combinatorial Benders cuts decomposition method performs very well. However, our computation shows the cuts generated by the method are still not so efficient in our problem. Hence, in the following we develop a logic based decomposition method, in which the cuts generated in each iteration can efficiently rule out the infeasible solutions and non-optimal solutions.

5.3.1 Idea and outline

In LBD method, the problem is decomposed into two subproblems: lower bound problem and upper bound problem. Lower bound problem is responsible for assigning tasks to quay crane and determining tasks processing sequence $\mathbf{X}$ by each quay crane, but in lower bound problem the crane interference among quay cranes, precedence relation among tasks and non-simultaneous handling among tasks are relaxed. Meanwhile, we also assume that there is no waiting for quay crane to process assigned tasks, hence the formulation for lower bound problem only depend on decision variable $\mathbf{X}$. The upper bound problem is used to incorporate the quay crane interference constraint and precedence constraint among task pairs to determine the minimum waiting time for quay crane to process the task following the solution by lower bound problem, upper bound

problem generates the optimal task process order $\mathbf{Y}$ among the quay cranes with respect to the assigned tasks and tasks process sequence obtained from lower bound problem.

The lower bound problem is used to provided feasible solution and update the lower bound of the problem. When a solution $\mathbf{X}$ is found by the lower bound problem, if $\mathbf{X}$ is feasible, we pass the solution to the upper bound problem, in which an optimal solution of $\mathbf{Y}$ with respect to the feasible solution $\mathbf{X}$ is returned and the upper bound of the problem is obtained. If $\mathbf{X}$ is infeasible, we add infeasible cuts to lower bound problem to rule out infeasible solutions. The solution procedure terminated when the difference of the lower bound computed by the lower bound problem and upper bound by the upper bound problem is within a given gap, or the lower bound problem become infeasible. Figure 5.3 shows the decomposition method solution procedure.

Figure 5.3: procedure for logic based decomposition method

5.3.2 Lower bound problem

As described above, the lower bound problem is based on the relaxation of crane interference constraint, task precedence constraints and non-simultaneous handling constraint among tasks. Therefore, objective function of lower bound problem is to minimize the maximum completion time of the quay cranes. With the relaxation of the problem, there is no waiting time for a quay crane to process the assigned tasks, the completion time of the quay crane can be expressed in terms of the summation of the ready time, process time and set up time to process all the tasks assigned, as shown in Equation (5.27).

$$\min \bar{z} = \max_{q \in \Gamma} \left(\sum_{i \in \Omega^0} (r_q + s_{iq}^0 + \tau) X_{0iq} + \sum_{i \in \Omega} \sum_{j \in \Omega} (s_{ij} + \tau) X_{ijq} \right) \quad (5.27)$$

We out linearize the object function, and the lower bound problem is formulated as follows,

$$[\text{Lower}] \quad \min \bar{z} = \eta \quad (5.28)$$

$$\text{s.t.} \quad \sum_{i \in \Omega} (r_q + s_{iq}^0 + \tau) X_{0iq} + \sum_{i \in \Omega} \sum_{j \in \Omega} (s_{ij} + \tau) X_{ijq} \leq \eta, \forall q \in \Gamma \quad (5.29)$$

$$\sum_{j \in \Omega^T} \sum_{q \in \Gamma} X_{ijq} \bar{L}_q \leq l_i, \forall i \in \Omega \quad (5.30)$$

$$\sum_{j \in \Omega^T} \sum_{q \in \Gamma} X_{ijq} \bar{U}_q \geq l_i, \forall i \in \Omega \quad (5.31)$$

$$\sum_{i \in \Omega^T} X_{0iq} = 1, \forall q \in \Gamma \quad (5.32)$$

$$\sum_{i \in \Omega^0} X_{i T q} = 1, \forall q \in \Gamma \quad (5.33)$$

$$\sum_{q \in \Gamma} \sum_{j \in \Omega^T} X_{ijq} = 1, \forall i \in \Omega \quad (5.34)$$

$$\sum_{q \in \Gamma} \sum_{j \in \Omega^0} X_{jiq} = 1, \forall i \in \Omega \quad (5.35)$$

$$\sum_{j \in \Omega^0} X_{jiq} - \sum_{j \in \Omega^T} X_{ijq} = 0, \forall i \in \Omega, \forall q \in \Gamma \quad (5.36)$$

$$X_{ijq} + X_{jiq} \leq 1, \forall i, j \in \Omega, \forall q \in \Gamma \quad (5.37)$$

Let $\mathbf{X}^0 = \{X_{ijq} | i \in \Omega^0, j \in \Omega^T, q \in \Gamma\}$ denote the solution obtained by the lower bound problem. It is observed that the solution $\mathbf{X}^0$ may not be a feasible solution to the origin model because constraint (5.8) and (5.9) are not presented in lower bound problem, and precedence relationship among task pairs which are guaranteed by constraints (5.10), (5.11) and (5.14) are also not included in lower bound problem. Therefore, the directed graph constructed by solution $\mathbf{X}^0$ may consist sub-cycles. For example in Figure 5.4 (the dashed line in the figure indicates precedence relation between tasks, like Figure 5.4b $\pi_{25} = 1$), infeasible solution $\mathbf{X}^0$ may lead to three types of sub-cycles: (a) sub-cycles in which no precedence violation is involved; (b) sub-cycles within a quay crane task processing sequence caused by precedence violation among tasks; (c) sub-cycles among task processing sequence by different quay cranes caused by precedence violation between task pairs. Therefore, we need to rule out solutions that cause sub-cycles efficiently.

Figure 5.4: type of infeasible solution obtained by lower bound problem: (a) sub-cycle with no precedence violation; (b) sub-cycle with precedence violation within a quay crane task processing sequence; (c) sub-cycle with precedence violation among different quay cranes

5.3.2.1 Infeasible cuts for the Lower bound problem

As infeasible solutions to origin problem may be returned by lower bound problem, we need to cut off these infeasible solutions efficiently. The first step is to identify the sub-cycles, we implement johnson's algorithm to find the elementary cycles resulted from the infeasible solution $\mathbf{X}^0$ (Johnson, 1975). In the following of the paper, following Balas et al. (1995), we define the following notations for simplicity:

- for any task set Q : $\mathbf{X}^q(Q) = \sum_{i \in Q} \sum_{j \in Q \setminus \{i\}} X_{ijq}$
- for any two task sets Q_1 and Q_2 : $\mathbf{X}^q(Q_1, Q_2) = \sum_{i \in Q_1} \sum_{j \in Q_2} X_{ijq}$
- for any task set Q , the link i to Q : $\mathbf{X}^q(i, Q) = \sum_{j \in Q} X_{ijq}$
- for any task set Q , the link Q to j : $\mathbf{X}^q(Q, j) = \sum_{i \in Q} X_{ijq}$

For sub-cycles in which no precedence violation among tasks is involved, it can be cut off from the solution space with the incorporation of valid inequality (5.38) (sub-tour elimination constraints) into the lower bound problem in each infeasible iteration,

$$\mathbf{X}^q(Q_n^k) \leq |Q_n^k| - 1, \quad \forall q \in \Gamma, n = 1, \dots, N \quad (5.38)$$

In inequality (5.38), Q_n^k represents the set of tasks that form the cycle n which is found at the iteration k , N is the number of cycles found at iteration k .

As for sub-cycles with precedence violation within a route, suppose the path of the cycle which results from infeasible solution $\mathbf{X}^q$ is from i_q to j_q , and $\pi_{j_q, i_q} = 1$ (which is the precedence violation of the cycle between i_q and j_q), denote Q_q the set of tasks within the path, task i_q and j_q are not the element of Q_q . After the identification of the sub-cycle, the precedence violation between task pair (j_q, i_q) can be broken down by adopting a valid inequality (5.39) (Balas et al., 1995), which is given as follows,

$$\mathbf{X}^q(i_q, Q_q) + \mathbf{X}^q(Q_q) + \mathbf{X}^q(Q_q, j_q) \leq |Q_q|, \quad \forall q \in \Gamma \quad (5.39)$$

When sub-cycles with precedence violation involved more than two quay cranes, the precedence violation breaking inequality can be generalized to valid inequality (5.40)

$$\begin{aligned} & \mathbf{X}^{q_1}(i_{q_1}, Q_{q_1}) + \mathbf{X}^{q_1}(Q_{q_1}) + \mathbf{X}^{q_1}(Q_{q_1}, j_{q_1}) + \cdots + \mathbf{X}^{q_m}(i_{q_m}, Q_{q_m}) + \mathbf{X}^{q_m}(Q_{q_m}) \\ & + \mathbf{X}^{q_m}(Q_{q_m}, j_{q_m}) \leq (|Q_{q_1}| + 1) + \cdots + (|Q_{q_m}| + 1) - 1, \quad q_1, \dots, q_m \in \Gamma \end{aligned} \quad (5.40)$$

In inequality (5.40), i_{q_m} and j_{q_m} are the first task and last task respectively of the path for quay crane q_m , which are part of the sub-cycle, Q_{q_m} represents the set of tasks within the path from i_{q_m} to j_{q_m} , i_{q_m} and j_{q_m} are not included in Q_{q_m} .

5.3.3 Upper bound problem

When a feasible solution $\mathbf{X}^+$ is obtained from lower bound problem, we incorporate the solution $\mathbf{X}^+$ with the non-interference constraints, precedence violation constraint and non-simultaneous handling constraint by solving the upper bound problem. The solution of the upper bound problem determines the minimum waiting time for the quay cranes to process each task resulting from solution $\mathbf{X}^+$. The upper bound problem is given as follows,

$$[\text{Upper}] \quad \min \hat{z} = C \quad (5.41)$$

$$\text{s.t} \quad C \geq C_i, \forall i \in \Omega \quad (5.42)$$

$$C_i + \tau - C_j \leq M(1 - Y_{ij}), \forall i, j \in \Omega \quad (5.43)$$

$$C_j - \tau - C_i \leq MY_{ij}, \forall i, j \in \Omega \quad (5.44)$$

$$Y_{ij} \geq \pi_{ij}, \forall i, j \in \Omega \quad (5.45)$$

$$Y_{ij} + Y_{ji} = 1, \forall (i, j) \in \Phi \quad (5.46)$$

$$C_i \geq 0, \forall i \in \Omega \quad (5.47)$$

$$C_i + s_{ij} + \tau - C_j \leq M(1 - X_{ijq}^+), \forall i, j \in \Omega, \forall q \in \Gamma \quad (5.48)$$

$$r_q + s_{jq}^0 + \tau - C_j \leq M(1 - X_{0jq}^+), \forall j \in \Omega, \forall q \in \Gamma \quad (5.49)$$

$$C_j + \Delta_{jj'}^{qq'} + \tau - C_{j'} \leq M \left(3 - Y_{jj'} - \sum_{i \in \Omega^0} X_{ijq}^+ - \sum_{i \in \Omega^0} X_{ij'q'}^+ \right), \quad (5.50)$$

$$\forall (j, j', q, q') \in \Upsilon$$

$$C_{j'} + \Delta_{jj'}^{qq'} + \tau - C_j \leq M \left(3 - Y_{j'j} - \sum_{i \in \Omega^0} X_{ijq}^+ - \sum_{i \in \Omega^0} X_{ij'q'}^+ \right), \quad (5.51)$$

$$\forall (j, j', q, q') \in \Upsilon$$

$$\sum_{i \in \Omega^0} X_{ijq}^+ + \sum_{i \in \Omega^0} X_{ij'q'}^+ \leq 1 + Y_{jj'} + Y_{j'j}, \forall (j, j', q, q') \in \Gamma \quad (5.52)$$

In the upper bound problem, $\mathbf{X}^+$ is known parameter. Denote $\mathbf{Y}^+ = \{Y_{ij} | i \in \Omega, j \in \Omega\}$, $\mathbf{C}^+ = \{C_i | i \in \Omega\}$ the solution of the upper bound problem.

Proof. As $\mathbf{X}^+$ is feasible solution, it means that $\mathbf{X}^+$ do not contain cycles. The problem is transformed to prove that there always exist $\mathbf{Y}^+$ so that the graph resulting from $(\mathbf{X}^+, \mathbf{Y}^+)$ does not contain cycles. Given the DAG resulted from $\mathbf{X}^+$, suppose we have only one task pair (i, j) , $\mathbf{Y}_{ij}$, to be determined. Suppose $\mathbf{Y}_{ij} = 1$ and $(\mathbf{X}^+, \mathbf{Y}_{ij})$ resulted in a cycle, then cycle can always be broken down by setting $\mathbf{Y}_{ij} = 0$ and $\mathbf{Y}_{ji} = 1$ (change the direction of arc (j, i)) so that the DAG resulted from $(\mathbf{X}^+, \mathbf{Y}_{ij})$ will not contain cycle. Hence, after $\mathbf{Y}_{ij}$ is determined, a new DAG is constructed. The procedure can be repeated without causing cycles. Hence, the proposition is proved. $\square$

5.3.3.1 Non-optimal cuts for the Lower bound problem

In case that $(\mathbf{X}^+, \mathbf{Y}^+)$ is non-optimal solution of the problem, we cut off the incumbent solution $(\mathbf{X}^+, \mathbf{Y}^+)$ from solution space. One option is to incorporate inequation (5.53) into the lower bound problem at each non-optimal iteration,

$$\sum_{\mathbf{x}^k \in C_k} \mathbf{x}^k + \sum_{\mathbf{x}^k \in C_k^+} (1 - \mathbf{x}^k) \geq 1 \quad (5.53)$$

In inequation (5.53), C_k denote the subset of the variable $\mathbf{X}^k = 0$ and C_k^+ denote the subset of the variable $\mathbf{X}^k = 1$ at iteration k . The inequation (5.53) cut off the incumbent solution from solution space of lower bound problem. It is observed that the inequation (5.53) can only cut off one non-optimal solution from solution space at each iteration and does not utilize too much information from upper bound problem, which is not an efficient cut to the lower bound problem.

From the incumbent solution $(\mathbf{X}^+, \mathbf{Y}^+)$, A longest path (or critical path) of the DAG resulted from $(\mathbf{X}^+, \mathbf{Y}^+)$ can be identified, based on which an enhanced inequality can be generated to cut off more non-optimal solutions from solution space at each non-optimal iteration. Suppose that the path from i_k to j_k by quay crane k are part of the longest path, and Q_k (apart from i_k and j_k) is the set of tasks processed by quay crane k from the path i_k to j_k , then we have the following valid inequality to cut off the non-optimal solution,

$$\begin{aligned} & \mathbf{X}^{q_1}(i_{q_1}, Q_{q_1}) + \mathbf{X}^{q_1}(Q_{q_1}) + \mathbf{X}^{q_1}(Q_{q_1}, j_{q_1}) + \cdots + \mathbf{X}^{q_m}(i_{q_m}, Q_{q_m}) \\ & + \mathbf{X}^{q_m}(Q_{q_m}) + \mathbf{X}^{q_m}(Q_{q_m}, j_{q_m}) \leq (|Q_{q_1}| + 1) + \cdots + (|Q_{q_m}| + 1) - 1 \end{aligned} \quad (5.54)$$

Proof. Let $(\mathbf{X}^*, \mathbf{Y}^*)$ be the global optimal solution of the original model and $\mathbf{C}^*$ be the associated objective function value. For the lower bound problem, according to the construction of infeasible cuts and non-optimal cuts, the optimal objective function value of the lower bound problem η^* will not be excluded and always underestimates the optimal value of $\mathbf{C}^*$, i.e., $\eta^* \leq \mathbf{C}^*$. In each iteration, η^* will be updated, and have $\eta_1^* \leq \eta_2^* \leq \cdots \eta_T^*$. From construction of upper bound problem, the objective value of upper bound problem is not guaranteed to decrease monotonically. However, with the exclusion of non-optimal solution from solution space, and with the iteration number approaches to infinity, the solution space will shrink and eventually become empty in the worst case, hence the convergence of the methods is proved. The efficiency of LBD method lies in the effects of infeasible cuts and non-optimal cuts. $\square$

5.4 Heuristic solution method

Despite that LBD method can reduce the computational time substantially as compared to simply using commercial software like CPLEX, it is still not very efficient for solving large-size practical problem. Therefore, in this section we further propose a task insertion heuristic (TIH) solution algorithm. The method is based on the Push-Forward Insertion Heuristic proposed by [Solomon \(1987\)](#), which is used to insert a new customer into the current route between two adjacent customers without violating the time constraints. In TIH method, task is treated as a customer to be inserted into a quay crane task processing sequence, while ensuring that the insert position cannot violate the precedence relation between tasks and does not form cycles.

5.4.1 Assigning and sequencing tasks to quay crane

The task sequences processed by a quay crane is referred as a route . Initially, each route contains two virtual task: 0 and T. 0 is the starting virtual task and T is the ending virtual task. Denote the route for quay crane q as $R_q = (j_0^q, j_1^q, \dots, j_{p-1}^q, j_p^q, \dots, j_T^q)$, wherein j_0^q is the virtual task 0, and j_T^q is the virtual task T. Denote $U = (u_1, u_2, \dots, u_m)$ as the set of unassigned tasks. Initially, U is the set of all tasks to be assigned and U will be empty set at the end of the solution procedure.

We firstly compute the best feasible insert place for task $u \in U$, which is given by Equation (5.55). Note that feasible insert place means that there is no violation of precedence constrained task pairs and no cycles in the route when a task is inserted in that place.

$$F_1(j(u), u, j'(u)) = \min_{q \in \Gamma, p=1 \dots T} F_1(j_{p-1}^q, u, j_p^q) \quad (5.55)$$

In Equation (5.55), $F_1(j_{p-1}^q, u, j_p^q)$ is the cost of task u if inserted in the two adjacent sequenced tasks (j_{p-1}^q, j_p^q) in the route q , which can be calculated as follows:

$$F_1(j^q, u, j'^q) = \alpha_1 F_{11}(j^q, u, j'^q) + \alpha_2 F_{12}(j^q, u, j'^q), \quad \alpha_1 + \alpha_2 = 1, \alpha_1, \alpha_2 \geq 0 \quad (5.56)$$

$$F_{11}(j^q, u, j'^q) = s_{ju} + s_{uj'} + \tau - s_{jj'} \quad (5.57)$$

$$F_{12}(j^q, u, j'^q) = \Delta C(j^q, u, j'^q) \quad (5.58)$$

Equation (5.57) calculates the incremental of time for the current route given u is inserted in the route. Equation (5.58) measures the incremental of make-span, and the calculation of make-span is described in section 5.4.2. $\Delta C(j^q, u, j'^q)$ denotes the incremental of make-span when u is inserted between j^q and j'^q .

If the best feasible insert place are determined for all the task $u \in U$, the best insert task u^* to be inserted in the current route can be computed as follows:

$$F_2(j(u^*), u^*, j'(u^*)) = \min_{u \in U} F_1(j(u), u, j'(u)) \quad (5.59)$$

The task insertion procedure is repeated until all the tasks have been assigned to quay cranes.

5.4.2 Scheduling of quay crane tasks

Given task sequences for each quay crane, different schedules can be constructed depending on the priority of task pairs that are prohibited to be processed simultaneously.

In the following, we adopt disjunctive graph model to build the schedules for a given task assignment and sequence. In a disjunctive graph $G = (\mathbb{E}, \mathbb{V})$, $\mathbb{E}$ is the arc set and $\mathbb{V}$ is the node set, and each task $i \in \mathbb{V}$ is represented by a node. Arcs $\mathbb{E}$ of G are distinguished by conjunctive arcs (denoted by A) and disjunctive arcs (denoted by D). The task sequences for quay crane $q \in \Gamma$ defines a path from node 0 to node T which is represented by a set of conjunctive arcs denoted by A_q . For task pairs $\{(i, j) \mid \pi_{ij} = 1, (i, j) \notin \cup_{q \in \Gamma} A_q\}$ that are precedence

constrained, they are also represented by a set of conjunctive arcs denoted by $A_{\mathbb{Q}}$. The conjunctive arcs A_q for each quay crane $q \in \Gamma$ and $A_{\mathbb{Q}}$ constitute all the conjunctive arcs in G , i.e., $A = (\cup_{q \in \Gamma} A_q) \cup A_{\mathbb{Q}}$. For the task pairs that are not allowed to be handled simultaneously, they are represented by the set of disjunctive arcs D . It contains arcs for task pairs $(i, j) \in \Phi$ and arcs of task pairs which are not precedence constrained and thus will cause crane interference under the given task assignment. An example of disjunctive graph is shown in Figure 5.5. For Figure 5.5 we have $A_1 = \{(0, 1), (1, 2), (2, 3), (3, 4), (4, T)\}$, $A_2 = \{(0, 5), (5, 6), (6, 7), (7, T)\}$, $A_{\mathbb{Q}} = \{(5, 3)\}$ and $D = \{(1, 5), (2, 5), (4, 7)\}$.

Figure 5.5: Example of disjunctive graph

Given a disjunctive graph G , we further define weight on each arc. Denote w_{ij} as the weight of arc $(i, j) \in \mathbb{E}$. Weights for arcs of $\cup_{q \in \Gamma} A_q$, $A_{\mathbb{Q}}$ and D are defined separately. For conjunctive arcs $(i, j) \in \cup_{q \in \Gamma} A_q$, the weights are given by Equation (5.60),

$$w_{ij} = \begin{cases} r_q + s_{0j}^q + \tau, & \text{if } j \in \Omega \text{ and } i = 0, (i, j) \in A_q, q \in \Gamma \\ 0, & \text{if } i \in \Omega \text{ and } j = T, (i, j) \in A_q, q \in \Gamma \\ s_{ij} + \tau, & \text{if } i \in \Omega \text{ and } j \in \Omega, (i, j) \in A_q, q \in \Gamma \end{cases} \quad (5.60)$$

Weight associated with arc $(i, j) \in \cup_{q \in \Gamma} A_q$ assesses the time needed for a quay crane to process a task or the ready time of quay crane plus the set-up time needed until the beginning of next task. Weights for conjunctive arcs $(i, j) \in A_{\mathbb{Q}}$

and disjunctive arcs $(i, j) \in D$ are given by Equation (5.61),

$$w_{ij} = \begin{cases} \Delta_{ij}^{qq'} + \tau, & \text{if } (i, j, q, q') \in \Upsilon \\ \tau + \tau, & \text{otherwise} \end{cases} \quad (5.61)$$

Weight of arc $(i, j) \in A_{\mathbb{Q}} \cup D$ reflects the minimum temporal distance for a safe crane movement or the minimum time to process task i and j if i is processed before j . In summary, the disjunctive graph obtained can be further denoted as $G = (\mathbb{E}, \mathbb{V}, \mathbb{W})$, where $\mathbb{W} = [w_{ij}]$ is the weight of arc.

From disjunctive graph G , a feasible schedule can be constructed by directing each pair of disjunctive arcs (determining the priority of tasks for task pair in disjunctive arcs), and the resulting graph G' should be acyclic. The scheme to direct disjunctive arcs shall be able to capture optimal schedule. In the approach of Bierwirth & Meisel (2009), disjunctive arcs are directed from nodes of the upper quay crane toward nodes of the lower quay crane, wherein cycle can be effected using their branching criterion. In our approach, we take the following procedure:

Step 1 determine the conjunctive arcs $\cup_{q \in \Gamma} A_q$ and $A_{\mathbb{Q}}$, and disjunctive arcs D ,

$$\text{set } \mathbb{E} = (\cup_{q \in \Gamma} A_q) \cup A_{\mathbb{Q}};$$

Step 2 compute the weight of each arcs in $\mathbb{E}$, and determine C_i for each node in the graph;

Step 3 order the completion time for each node connected by disjunctive arcs in ascending order, choose the node with minimum completion time, direct the disjunctive arcs from the minimum node to another node connected by the disjunctive arc, add the directed disjunctive arcs into $\mathbb{E}$, remove the directed arcs from D ;

Step 4 check if $D = \emptyset$, return the completion time at each node and go to step 5; if $D \neq \emptyset$ repeat step 2 and step 3;

Step 5 the make-span is the maximum completion time of the node.

Therefore, the TIH algorithm can now be described as Figure 5.6.

Figure 5.6: Flow chart for TIH method

5.5 Numerical examples

In this section, we evaluate the performance of LBD and TIH algorithm by solving a set of benchmark instances, and also analyze the operation efficiency of QCSP with double cycling in comparison with the operation method that loading containers after all the container are unloaded. Table 5.2 shows the basic parameter settings in the numerical experiment: the processing time for a quay crane to handle a task; the quay crane travel time between two adjacent bays; the trolley moving time from seaside to quayside or vice versa; the ready time

of each quay crane; the safety margin between two adjacent quay cranes and the parameter settings for TIH method.

Table 5.2
Parameter settings for numerical study

Parameters	Values
The processing time of quay crane to handle a task, τ	1 (unit)
The spreader moving time from quayside to seaside or vice versa, a	2 (unit)
The quay crane travel time between two adjacent bay, $\bar{t}$	2 (unit)
Ready time for quay crane to start handling task, r_q	0 (unit)
Safety margin between two adjacent quay cranes, δ	1 (bay)
Parameter α_1 for TIH method	0.2
Parameter α_2 for TIH method	0.8

All the computational experiments are performed on a personal computer with 8 gigabytes of RAM, Windows 7 Professional Operating System and Intel Core i5 processors with 2.6 gigahertz cores. The MIP model and LBD method are implemented in Python and embedded within the ILOG Concert Technology frame work, based on ILOG CPLEX 12.6. TIS method is coded in Python.

5.5.1 Evaluate the performance of LBD and TIH

The percentage of loading tasks is 50.0% for each benchmark instance in this section. To evaluate the performance of LBD and TIH, we compare the solution results with those solved from using the commercial software of CPLEX. The maximum computation time for each method is set to be two hours. We report the problem size of each instance in terms of the number of constraints and variables, we also report the best-found solution of each instance, the gaps

and computation time it takes for CPLEX, LBD and TIH. The results are shown in Table 5.3.

Table 5.3
Results obtained by CPLEX, LBD and TIH

Instance			Problem size		CPLEX			LBD			TIH		
No.	$ \Gamma $	$ \Omega $	$ E $	$ V $	C	$T(s)$	$g(\%)$	C	$T(s)$	$g(\%)$	C	$T(s)$	$g(\%)$
1	2	10	1054	399	10.0	0.3	0.00	10.0	0.4	0.0	10.0	0.4	0.0
2	2	10	1044	399	9	0.4	0.00	9	0.4	0.00	10	0.1	10.0
3	2	10	1039	399	8	0.5	0.00	8	0.5	0.00	10	0.1	20.0
4	2	16	2722	921	12	3.4	0.00	12	2.3	0.00	13	0.5	7.7
5	2	16	2707	921	12	12.4	0.00	12	2.9	0.00	12	0.6	0.00
6	2	16	2697	921	12	14	0.00	12	8.9	0.00	12	0.7	0.00
7	2	20	4234	1389	14	118.9	0.00	14	4.6	0.00	14	1.2	0.00
8	2	20	4209	1389	14	7200	7.10	14	13.9	0.00	16	1.8	12.5
9	2	20	4204	1389	14	7200	14.3	14	4.6	0.00	16	1.9	12.5
10	2	24	6182	1953	18	7200	33.3	16	29.4	0.00	18	5.3	12.5
11	3	30	18694	4003	n.a	7200	n.a	18	464.1	0.00	18	12.7	0.00
12	3	30	18680	4003	n.a	7200	n.a	16	145.5	0.00	18	11.3	11.1
13	3	30	18037	4003	n.a	7200	n.a	16	57.6	0.00	16	25.7	0.00
14	3	36	25621	5665	n.a	7200	n.a	16	12.6	0.00	18	50.8	11.1
15	3	36	25691	5665	n.a	7200	n.a	18	23.7	0.00	21	47.7	14.3

Note: $|\Gamma|$ and $|\Omega|$ are the number of quay cranes and tasks. $|E|$, $|V|$ denote the number of constraints and variables of the problem. $T(s)$ is the running time of solution algorithm, the time unit is seconds. $g(\%)$ is the gap of Lower bound and upper bound.

It is observed from Table 5.3 that optimal solution of small instance can be efficiently obtained by CPLEX, e.g., instance 1 to 10 are solved to optimal quiet

efficient. However, when problem size increases, the computation time of CPLEX increase rapidly. Indeed, instance 10 to 15 do not return a solution within two hours. The solution quality by LBD is rather good: the gaps of lower bound solution and upper bound solution is 0 which means optimal solution is obtained for all the benchmark instances. Besides, the computation time of LBD algorithm is also competitive in comparison with CPLEX. As for TIH algorithm, the solution quality is not as good as the other two solution methods. One can observe that the gaps for some instances are a bit big. However, the computation time of TIH is rather competitive, as the maximum computation time of all the instances is only 50.8 seconds.

5.5.2 Evaluate the efficiency of QCSP considering double cycling strategies

In this section, we compare the efficiency of three operation methods of quay cranes to load and unload container ship. Method 1: loading containers after all the unloading containers of container ship are discharged. For method 1, we have the following two inequality added to the model in Section 5.2: (1) $X_{ijq} = 0, \forall i \in \bar{\Omega}, \forall j \in \underline{\Omega}$; (2) $Y_{ij} = 1, \forall i \in \bar{\Omega}, \forall j \in \underline{\Omega}$. $\bar{\Omega}$, wherein $\underline{\Omega}$ are the set of loading containers and unloading containers respectively. Method 2: the containers are loaded in each bay only after all the unloading containers in that bay are discharged. Similarly, for method 2, we add the following two inequality to the origin model: (1) $X_{ijq} = 0$, if $i \in \bar{\Omega}, j \in \underline{\Omega}$ and $l_i = l_j$; (2) $Y_{ij} = 1$, if $i \in \bar{\Omega}, j \in \underline{\Omega}$ and $l_i = l_j$. Method 3: loading and unloading containers simultaneously, the model of which is given in Section 5.2.

Each numerical experiment is solved by LBD method. The computational results are reported, such as the optimal solution for each instance and the computation time that LBD method takes, the time saving of method 2 compared with method1 and the time saving of method 3 compared with method 1 and

method 2. For the comparison of optimal solution, the computation time is not limited.

Table 5.4

The comparison of efficiency for each quay crane handling method, where $\tau = 1$, $a = 2\tau$ and $\bar{t} = 2\tau$, r =number of loading tasks/all the tasks

Ratio	Instance		Method 1		Method 2			Method 3			
r	$ \Gamma $	$ \Omega $	$\tilde{C}$	$T(s)$	$\hat{C}$	$T(s)$	$s_1(\%)$	$\bar{C}$	$T(s)$	$s_1(\%)$	$s_2(\%)$
0	1	30	105	6.8	105	8.4	0.00	105	7.1	0.00	0.00
	2	30	54	43.2	54	106.9	0.00	54	44	0.00	0.00
	3	30	36	164.4	36	210.4	0.00	36	170.5	0.00	0.00
0.2	1	30	118	4.8	83	1.7	29.66	81	11.4	31.36	2.41
	2	30	61	2288	44	1124	27.87	42	357.7	31.15	4.55
	3	30	37	98.5	30	83.7	18.92	28	112.7	24.32	6.67
0.4	1	30	118	4.1	77	278920	34.74	57	6.3	51.69	25.97
	2	30	61	898.8	45	311698	26.23	30	337.1	50.82	33.33
	3	30	37	203.3	27	163255	27.03	20	156.2	45.95	25.93
0.6	1	30	118	1.5	77	476320	34.74	55	1.6	53.39	25.97
	2	30	61	2499	45	167233	26.23	28	287.1	54.10	37.78
	3	30	37	99.8	30	13346	18.92	18	137.1	51.35	40.0
0.8	1	30	118	2.6	83	2.5	29.66	79	1.6	33.05	4.82
	2	30	61	424.3	44	233.1	27.87	40	706	34.43	9.09
	3	30	37	120.8	30	1176	18.92	26	269	29.73	13.33
1.0	1	30	103	7.9	103	13.6	0.00	103	7.8	0.00	0.00
	2	30	52	102.1	52	253.6	0.00	52	104	0.00	0.00
	3	30	34	586.9	34	679.0	0.00	34	600.2	0.00	0.00

Note: $|\Gamma|$ and $|\Omega|$ are the number of quay cranes and tasks. T is the running time of solution algorithm, the time unit is seconds. $\tilde{C}$, $\hat{C}$ and $\bar{C}$ are the optimal make-span by method 1, method 2 and method 3 respectively.

Table 5.4 shows the results of the comparison among each operation method of the quay crane. A notable observation is that the computation time of method 2 is greater than that of method 1 and method 3. That is because that the inner formulation ($Y_{ij} = 1$, if $i \in \bar{\Omega}, j \in \underline{\Omega}$ and $l_i = l_j$) generates a large amount of precedence relation among task pairs, which makes the search of feasible solution time consuming.

When the ratio of loading tasks is 0% or 100%, QCSP considering double cycling are not applicable, hence no improvement of operation efficiency. As the ratio r approaches 50%, QCSP considering double cycling improve the operation efficiency most significantly. It is also observed that the operation of method 3 performs better than the operation of method 2, and operation of method 2 is better than the operation of method 1.

5.6 Conclusion

In this chapter, we propose a MIP model for the QCSP considering double cycling. A general practice in container terminals is to carry out loading operation after the discharging operations, which is less effective compared with the double cycling strategies. Scheduling quay crane in this way can reduce the ineffective empty spreader moves substantially. Our model incorporates double cycling strategies into QCSP which results in a much more complex model formulation. Computational experiments show that the increase of problem size leads to an exponentially increasing CPU time to solve the model.

To resolve the computation difficulties, we propose a logic based decomposition method for solving the proposed model formulation. A lower bound problem is formulated to generate task assigning and sequencing, and an upper bound problem is formulated to determine the minimum waiting time for the quay cranes to process the tasks sequence obtained by lower bound problem. We

also present a task insertion scheduling heuristic method for the problem. In this method, we first calculate the best insert place for each task, and then determine the best insert task to be inserted into the route. Out of the two solution methods proposed in this study, the logic based decomposition approach ensures the global optimal solution quality, while the task insertion heuristic reduces the computational time significantly by compromising the solution quality.

As the main objective of this study is to load and unload a single container, if it is used to a large number of containers, the model might be intractable. Besides, we do not distinguish the tasks loading or discharging from upper deck and lower deck of a container ship, which will affect the model formulation and the optimal solutions. In the future study, we are going to develop a tractable model formulation and explicitly consider the upper deck tasks and lower deck tasks in the model formulation.

Chapter 6

CONCLUSIONS

This chapter summarizes the research questions and major contributions of this thesis. Future researches are also discussed.

6.1 Thesis summary

The work in this thesis concerns with developing more realistic models and solution algorithm. A brief summary of the thesis is given as follows.

Chapter 1 makes a brief introduction to the background and motivations of this study. The research objective are also given in this chapter.

Chapter 2 reviews previous work on container stowage planning problem and quay crane scheduling problem, the solution methods for integer programming problems are also briefly reviewed, and make a summarization about the development of corresponding problems.

In Chapter 3, we present a Benders decomposition method for a generic integer linear programming problem. Through introducing auxiliary variables into master problem, we formulate the subproblem as feasibility problem. Based on this, a relaxed cut generation problem is given by which we can obtain Benders cut. However, the Benders cut may not cut off infeasible solution, in this situation we introduced two fixed integer problems to add solution elimination

constraint when the Benders cut is not valid. This guaranteed the convergence of the proposed solution approach.

Chapter 4 studies the container stowage slot planning problem from a multi-port view. The main contributions of this paper are three folds. First, the slot planning problem is modeled from a multi-port view with incorporation of more realistic constraints. A multi-port view of slot planning is essential in reducing the negative effect of stowing containers in the subsequent port of call. Second, we extend the slot planning problem by explicitly including the constraint that the weight of containers stowed in the same ship-stack does not increase from the bottom to the top. Indeed, this constraint is practically required in real loading operations but was not fulfilled in the existing model formulations. Third, as the slot planning problem considered in this study is NP-hard, we propose two specific solution algorithms to solve it, i.e., a slot attributes based heuristic method and tabu search method.

Chapter 5 investigates the efficiency of quay crane scheduling problem considering double cycling. The main contributions of this chapter are three folds. First, a mixed integer programming (MIP) model for the QCSP considering double cycling strategies is developed. Most of existing studies focused on double cycling with single QC, i.e., only one single quay crane performs loading and unloading simultaneously on a given bay of container ship. In this case, the interaction among quay cranes is not considered and the efficiency of double cycling is underestimated. Our model in this study incorporates QCs double cycling within the context of the quay crane scheduling problem, in which crane interference, safety margin between different QCs, precedence relationship among tasks and non-simultaneous handling among tasks all considered in a much more realistic and complicated problem formulation. Second, we propose a logic based decomposition (LBD) method to solve the model. The LBD method in our paper can guarantee global optimal solution, meanwhile reduce computation time substantially. Indeed, the proposed LBD method is also applicable

for general QCSP. Third, a task insertion heuristic (TIH) method is developed to obtain a sufficiently good solution with even less computational time, in which a disjunctive graph model is adopted to build the algorithm.

In Chapter 6, a conclusion and discussion of the thesis are given. The findings and contributions of the thesis are concluded in this chapter, and a number of recommendations for future work are also provided.

6.2 Future research

In the future, further work will be conducted on the following part.

First, as the proposed model for multiport container stowage slot planning is integer programming model, we will apply the proposed Benders decomposition solution method to solve the multiport container slot planning model.

Second, since the study of quay crane scheduling problem considering double cycling refers a task as loading or unloading a single container, the model might be intractable for a large number of containers. Besides, we do not distinguish the upper deck tasks from lower deck tasks of a container ship, which will affect the model formulation and the optimal solution. In the future study, we are going to develop a more tractable model formulation and explicitly consider the upper deck tasks and lower deck tasks in the model formulation.

References

- Al-Dhaheri, Noura, & Ali Diabat. (2015), ``The quay crane scheduling problem ", Journal of Manufacturing Systems, 36:87 -- 94.
- Al-Dhaheri, Noura, Aida Jebali, & Ali Diabat. (2016a), ``The quay crane scheduling problem with nonzero crane repositioning time and vessel stability constraints ", Computers & Industrial Engineering, 94:230 -- 244.
- Al-Dhaheri, Noura, Aida Jebali, & Ali Diabat. (2016b), ``A simulation-based genetic algorithm approach for the quay crane scheduling under uncertainty ", Simulation Modelling Practice and Theory, 66:122 -- 138.
- Ambrosino, Daniela, Davide Anghinolfi, Massimo Paolucci, & Anna Sciomachen. (2010). An Experimental Comparison of Different Heuristics for the Master Bay Plan Problem. In Festa, Paola (ed), *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pages 314--325. Springer Berlin Heidelberg.
- Ambrosino, Daniela, Anna Sciomachen, & Elena Tanfani. (2004), ``Stowing a containership: the master bay plan problem ", Transportation Research Part A: Policy and Practice, 38(2):81 -- 99.
- Applegate, David, Robert Bixby, Vašek Chvátal, & William Cook. (2003), ``Implementing the dantzig-fulkerson-johnson algorithm for large traveling salesman problems ", Mathematical programming, 97(1-2):91--153.
- Applegate, David, Robert Bixby, William Cook, & Vasek Chvátal. (1998), ``On the solution of traveling salesman problems ".

- Araújo, Eliseu Junio, Antonio Augusto Chaves, Luiz Leduino de Salles Neto, & Anibal Tavares de Azevedo. (2016), ``Pareto clustering search applied for 3d container ship loading plan problem ", Expert Systems with Applications, 44:50--57.
- Avriel, Mordecai, & Michal Penn. (1993), ``Exact and approximate solutions of the container ship stowage problem ", Computers & Industrial Engineering, 25(1):271 -- 274.
- Avriel, Mordecai, Michal Penn, & Naomi Shpirer. (2000), ``Container ship stowage problem: complexity and connection to the coloring of circle graphs ", Discrete Applied Mathematics, 103(1):271 -- 279.
- Avriel, Mordecai, Michal Penn, Naomi Shpirer, & Smadar Witteboon. (1998), ``Stowage planning for container ships to reduce the number of shifts ", Annals of Operations Research, 76(0):55--71.
- Balas, Egon, Matteo Fischetti, & William R. Pulleyblank. (1995), ``The precedence-constrained asymmetric traveling salesman polytope ", Mathematical Programming, 68(1):241--265.
- Bazaraa, Mokhtar S, & Hanif D Sherali. (1981), ``On the choice of step size in subgradient optimization ", European Journal of Operational Research, 7(4):380--388.
- Beens, Marie-Anne, & Evrim Ursavas. (2016), ``Scheduling cranes at an indented berth ", European Journal of Operational Research, 253(2):298 -- 313.
- Belgacem, Rachid, & Abdessamad Amir. (2017), ``A new modified deflected subgradient method ", Journal of King Saud University - Science.
- Benders, Jacques F. (1962), ``Partitioning procedures for solving mixed-variables programming problems ", Numerische mathematik, 4(1):238--252.

- Benoist, Thierry, Etienne Gaudin, & Benoît Rottembourg. (2002). Constraint programming contribution to benders decomposition: A case study. In International Conference on Principles and Practice of Constraint Programming, pages 603--617. Springer.
- Bierwirth, Christian, & Frank Meisel. (2009), ``A fast heuristic for quay crane scheduling with interference constraints ", Journal of Scheduling, 12(4):345--360.
- Bierwirth, Christian, & Frank Meisel. (2010), ``A survey of berth allocation and quay crane scheduling problems in container terminals ", European Journal of Operational Research, 202(3):615 -- 627.
- Bierwirth, Christian, & Frank Meisel. (2015), ``A follow-up survey of berth allocation and quay crane scheduling problems in container terminals ", European Journal of Operational Research, 244(3):675 -- 689.
- Bockmayr, Alexander, & Nicolai Pizaruk. (2006), ``Detecting infeasibility and generating cuts for mixed integer programming using constraint programming ", Computers & Operations Research, 33(10):2777--2786.
- Bragin, Mikhail A, Peter B Luh, Joseph H Yan, Nanpeng Yu, & Gary A Stern. (2015), ``Convergence of the surrogate lagrangian relaxation method ", Journal of Optimization Theory and applications, 164(1):173--201.
- Christensen, Jonas, & Dario Pacino. (2017), ``A matheuristic for the cargo mix problem with block stowage ", Transportation Research Part E: Logistics and Transportation Review, 97:151--171.
- Chu, Yingyi, & Quanshi Xia. (2004). Generating Benders Cuts for a General Class of Integer Programming Problems, pages 127--141. Springer Berlin Heidelberg, Berlin, Heidelberg.

- Chvátal, Vašek, William Cook, George B Dantzig, Delbert R Fulkerson, & Selmer M Johnson. (2010). Solution of a large-scale traveling-salesman problem. In 50 Years of Integer Programming 1958-2008, pages 7--28. Springer.
- Codato, Gianni, & Matteo Fischetti. (2006), ``Combinatorial benders' cuts for mixed-integer linear programming ", Operations Research, 54(4):756--766.
- Daganzo, Carlos F. (1989), ``The crane scheduling problem ", Transportation Research Part B: Methodological, 23(3):159--175.
- de Azevedo, Anibal Tavares, Ribeiro Cassilda Maria, Galeno Jose de Sena, Antonio Augusto Chaves, Luis Leduino Salles Neto, & Antonio Carlos Moretti. (2014), ``Solving the 3d container ship loading planning problem by representation by rules and meta-heuristics ", International Journal of Data Analysis Techniques and Strategies, 6(3):228--260. PMID: 63060.
- Delgado, Alberto. (2013). Models and Algorithms for Container Vessel Stowage Optimization. PhD thesis.
- Delgado, Alberto, Rune Møller Jensen, & Nicolas Guilbert. (2012a). A placement heuristic for a commercial decision support system for container vessel stowage. In Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En, pages 1--9. IEEE.
- Delgado, Alberto, Rune Moller Jensen, Kira Janstrup, Trine Hoyer Rose, & Kent Hoj Andersen. (2012b), ``A constraint programming model for fast optimal stowage of container vessel bays ", European Journal of Operational Research, 220(1):251 -- 261.
- Ding, Ding, & Mabel C Chou. (2015), ``Stowage planning for container ships: a heuristic algorithm to reduce the number of shifts ", European Journal of Operational Research, 246(1):242--249.

- Dubrovsky, Opher, Gregory Levitin, & Michal Penn. (2002), ``A genetic algorithm with a compact solution encoding for the container ship stowage problem ", *Journal of Heuristics*, 8(6):585--599.
- Fakhri, Ashkan, Mehdi Ghatee, Antonios Fragkogios, & Georgios KD Saharidis. (2017), ``Benders decomposition with integer subproblem ", *Expert Systems with Applications*, 89:20--30.
- Fischetti, Matteo, Ivana Ljubić, & Markus Sinnl. (2017), ``Redesigning benders decomposition for large-scale facility location ", *Management Science*, 63(7):2146--2162.
- Fischetti, Matteo, Domenico Salvagnin, & Arrigo Zanette. (2010), ``A note on the selection of benders' cuts ", *Mathematical Programming*, 124(1):175--182.
- Fisher, Marshall L. (1981), ``The lagrangian relaxation method for solving integer programming problems ", *Management science*, 27(1):1--18.
- Floudas, Christodoulos A, & Panos M Pardalos. (2001a). *Encyclopedia of optimization*, volume 1. Springer Science & Business Media.
- Floudas, Christodoulos A, & Panos M Pardalos. (2001b). *Encyclopedia of optimization*, volume 1. Springer Science & Business Media.
- Francisco, Dario Pacino, & Ramon Alvarez-Valdes. (2016), ``A {GRASP} algorithm for the container stowage slot planning problem ", *Transportation Research Part E: Logistics and Transportation Review*, 94:141 -- 157.
- Geoffrion, A, & R Me Bride. (1978), ``Lagrangean relaxation applied to capacitated facility location problems ", *AIIE transactions*, 10(1):40--47.
- Geoffrion, Arthur M. (1972), ``Generalized benders decomposition ", *Journal of optimization theory and applications*, 10(4):237--260.
- Glover, Fred, Manuel Laguna, & Rafael Marti. (2007), ``Principles of tabu search ", *Approximation Algorithms and Metaheuristics*, 23:1--12.

- Glover, Fred, Eric Taillard, & Eric Taillard. (1993), ``A user guide to tabu search ", *Annals of Operations Research*, 41(1):1--28.
- Goodchild, Anne V, & CF Daganzo. (2007), ``Crane double cycling in container ports: Planning methods and evaluation ", *Transportation Research Part B: Methodological*, 41(8):875--891.
- Goodchild, Anne V, & Carlos F Daganzo. (2006), ``Double-cycling strategies for container ships and their effect on ship loading and unloading operations ", *Transportation Science*, 40(4):473--483.
- He, Junliang, Youfang Huang, Wei Yan, & Shuaian Wang. (2015), ``Integrated internal truck, yard crane and quay crane scheduling in a container terminal considering energy consumption ", *Expert Systems with Applications*, 42(5):2464 -- 2487.
- Hernández, Paula Hernández, Laura Cruz-Reyes, Patricia Melin, Julio Mar-Ortiz, Héctor Joaquín Fraire Huacuja, Héctor José Puga Soberanes, & Juan Javier González Barbosa. (2013). An ant colony algorithm for improving ship stability in the containership stowage problem. In *Mexican International Conference on Artificial Intelligence*, pages 93--104. Springer.
- Hoffman, Alan J, & Joseph B Kruskal. (2010). Integral boundary points of convex polyhedra. In *50 Years of Integer Programming 1958-2008*, pages 49--76. Springer.
- Holmberg, Kaj, Mikael Rönnqvist, & Di Yuan. (1999), ``An exact algorithm for the capacitated facility location problems with single sourcing ", *European Journal of Operational Research*, 113(3):544--559.
- Hooker, John N, & Greger Ottosson. (2003), ``Logic-based benders decomposition ", *Mathematical Programming*, 96(1):33--60.
- Johnson, Donald B. (1975), ``Finding all the elementary circuits of a directed graph ", *SIAM Journal on Computing*, 4(1):77--84.

- Kang, Jun-Gyu, & Yeong-Dae Kim. (2002), ``Stowage planning in maritime container transportation ", Journal of the Operational Research Society, 53(4):415-426.
- Kelley, James E, Jr. (1960), ``The cutting-plane method for solving convex programs ", Journal of the society for Industrial and Applied Mathematics, 8(4):703--712.
- Kim, Hak-Jin, & John N Hooker. (2002), ``Solving fixed-charge network flow problems with a hybrid optimization and constraint programming approach ", Annals of Operations Research, 115(1-4):95--124.
- Kim, Kap Hwan, & Young-Man Park. (2004), ``A crane scheduling method for port container terminals ", European Journal of Operational Research, 156(3):752 -- 768.
- Ku, Dusan, & Tiru S Arthanari. (2016), ``On double cycling for container port productivity improvement ", Annals of Operations Research, 243(1-2):55--70.
- Lasdon, Leon S. (2002). Optimization theory for large systems. Courier Corporation.
- Lee, Chung-Yee, Ming Liu, & Chengbin Chu. (2015), ``Optimal algorithm for the general quay crane double-cycling problem ", Transportation Science, 49(4):957--967.
- Lee, Der-Horng, Hui Qiu Wang, & Lixin Miao. (2008), ``Quay crane scheduling with non-interference constraints in port container terminals ", Transportation Research Part E: Logistics and Transportation Review, 44(1):124 -- 135.
- Li, Feng, Chunhua Tian, Rongzeng Cao, & Wei Ding. (2008). An Integer Linear Programming for Container Stowage Problem. In Bubak, Marian, Geert Dick van Albada, Jack Dongarra, & Peter M.A. Sloot (eds), Computational Science - ICCS 2008, volume 5101 of *Lecture Notes in Computer Science*, pages 853--862. Springer Berlin Heidelberg.

- Liu, Ming, Feng Chu, Zizhen Zhang, & Chengbin Chu. (2015), ``A polynomial-time heuristic for the quay crane double-cycling problem with internal-reshuffling operations ", *Transportation Research Part E: Logistics and Transportation Review*, 81:52 -- 74.
- Moccia, Luigi, Jean-Francois Cordeau, Manlio Gaudioso, & Gilbert Laporte. (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, Maria Flavia, Marcello Sammarra, & Gregorio Sorrentino. (2014), ``The terminal-oriented ship stowage planning problem ", *European Journal of Operational Research*, 239(1):256 -- 265.
- Ng, WC, & KL Mak. (2006), ``Quay crane scheduling in container terminals ", *Engineering Optimization*, 38(6):723--737.
- Pacino, Dario, Alberto Delgado, Rune Moller Jensen, & Tom Bebbington. (2011). Fast Generation of Near-Optimal Plans for Eco-Efficient Stowage of Large Container Vessels. In Bose, Jurgen W., Hao Hu, Carlos Jahn, Xiaoning Shi, Robert Stahlbock, & Stefan Vos (eds), *Computational Logistics*, volume 6971 of *Lecture Notes in Computer Science*, pages 286--301. Springer Berlin Heidelberg.
- Peterkofsky, Roy I, & Carlos F Daganzo. (1990), ``A branch and bound solution method for the crane scheduling problem ", *Transportation Research Part B: Methodological*, 24(3):159--172.
- Pferschy, Ulrich, & Rostislav Staněk. (2017), ``Generating subtour elimination constraints for the tsp from pure integer solutions ", *Central European journal of operations research*, 25(1):231--260.
- Pinedo, Michael L. (2016). *Scheduling: theory, algorithms, and systems*. Springer.

- Rahmaniani, Ragheb, Teodor Gabriel Crainic, Michel Gendreau, & Walter Rei. (2017), ``The benders decomposition algorithm: A literature review ", European Journal of Operational Research, 259(3):801--817.
- Rei, Walter, Jean-François Cordeau, Michel Gendreau, & Patrick Soriano. (2009), ``Accelerating benders decomposition by local branching ", INFORMS Journal on Computing, 21(2):333--345.
- Sadykov, Ruslan, & Laurence A Wolsey. (2006), ``Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates ", INFORMS Journal on Computing, 18(2):209--217.
- Sammarrà, Marcello, Jean-François Cordeau, Gilbert Laporte, & M Flavia Monaco. (2007), ``A tabu search heuristic for the quay crane scheduling problem ", Journal of Scheduling, 10(4-5):327--336.
- Sciomachen, Anna, & Elena Tanfani. (2003), ``The master bay plan problem: a solution method based on its connection to the three-dimensional bin packing problem ", IMA Journal of Management Mathematics, 14(3):251--269.
- Secretariat, Development. (2015). Review of maritime transport. United Nations.
- Solomon, Marius M. (1987), ``Algorithms for the vehicle routing and scheduling problems with time window constraints ", Operations research, 35(2):254--265.
- Stahlbock, Robert, & Stefan Voß. (2007), ``Operations research at container terminals: a literature update ", OR Spectrum, 30(1):1--52.
- Tavakkoli-Moghaddam, Reza, Ahmad Makui, S Salahi, M Bazzazi, & F Taheri. (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.

- Tierney, Kevin, Dario Pacino, & Rune Moller Jensen. (2014), ``On the complexity of container stowage planning problems ", Discrete Applied Mathematics, 169:225 -- 230.
- Tlili, Takwa, Sami Faiz, & Saoussen Krichen. (2013). A particle swarm optimization for solving the one dimensional container loading problem. In Modeling, Simulation and Applied Optimization (ICMSAO), 2013 5th International Conference on, pages 1--4. IEEE.
- Vis, Iris FA, & Rene De Koster. (2003), ``Transshipment of containers at a container terminal: An overview ", European journal of operational research, 147(1):1--16.
- Wang, Shuaian, Qiang Meng, & Zhiyuan Liu. (2013), ``Bunker consumption optimization methods in shipping: A critical review and extensions ", Transportation Research Part E: Logistics and Transportation Review, 53:49 -- 62.
- Wilson, ID, & PA Roach. (2000), ``Container stowage planning: a methodology for generating computerised solutions ", Journal of the Operational Research Society, 51(11):1248--1255.
- Wilson, Ian D, Paul A Roach, & J Andrew Ware. (2001), ``Container stowage pre-planning: using search to generate solutions, a case study ", pages 349--362.
- Xue, Zhaojie, Canrong Zhang, Lixin Miao, & Wei-Hua Lin. (2013), ``An ant colony algorithm for yard truck scheduling and yard location assignment problems with precedence constraints ", Journal of systems science and systems engineering, 22(1):21--37.
- Zhang, Haipeng, & Kap Hwan Kim. (2009), ``Maximizing the number of dual-cycle operations of quay cranes in container terminals ", Computers & Industrial Engineering, 56(3):979 -- 992. Intelligent Manufacturing and Logistics.

Zhao, Xing, Peter B Luh, & Jihua Wang. (1999), ``Surrogate gradient algorithm for lagrangian relaxation ", Journal of optimization Theory and Applications, 100(3):699--712.