The Crane Split and Sequencing Problem with Clearance and Yard Congestion Constraints in Container Terminal Ports

by

Shawn Choo

B. Eng (Electrical Engineering)
National University of Singapore, 2005

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

Master of Science in Computation for Design and Optimization

at the

Massachusetts Institute of Technology

September 2006

Signature of author.....
Computation for Design and Optimization Program
August 11, 2006

Certified by.....
David Simchi-Levi
Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by.....
Jaime Peraire
Professor of Aeronautics and Astronautics
Co-director, Computation for Design and Optimization Program

THE CRANE SPLIT AND SEQUENCING PROBLEM WITH CLEARANCE AND YARD CONGESTION CONSTRAINTS IN CONTAINER TERMINAL PORTS

by

SHAWN CHOO

Submitted to the School of Engineering
on August 11, 2006 in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computation for Design and Optimization

Abstract

One of the steps in stowage planning is crane split and sequencing, which determines the order of container discharging and loading jobs *quay cranes* (QCs) perform so that the completion time (or *makespan*) of ship operation is minimized. The vessel's load profile, number of bays and number of allocated QCs are known to port-planners hours before its arrival, and these are input parameters to the problem. The problem is modeled as a large-scale linear IP where the planning horizon is discretized into time intervals and at most one QC can be assigned to a bay at any period. We introduce clearance constraints, which prevent adjacent QCs from being positioned too close to one another, and yard congestion constraints, which prevent yard storage locations from being overly accessed at any time. This makes the model relevant in an industrial setting. We examine the case only a single ship arrives at port, and the case where multiple ships berth at different times in the planning horizon. The berth time of each ship and number of ships arriving is known. The problem is difficult to solve without any special technique applied.

For the single-ship problem, a heuristic approach, which produces high-quality solutions, is developed. A branch-and-price method re-formulates the problem into a set-covering form with huge number of variables; standard variable branching provides optimal solutions very efficiently. For the multiple-ship problem, a solution strategy is developed combining Lagrangian relaxation, branch-and-price and heuristics. After relaxing the yard congestion constraints, the problem decomposes into smaller sub-problems, each involving one ship; the sub-problems are then re-formulated into a column generation form and solved using branch-and-price to obtain Lagrangian solutions and lower-bound values. Lagrangian multipliers are iteratively updated using the sub-gradient method. A primal heuristic detects and eliminates infeasibilities in the Lagrangian solutions which then become an upper bound to the optimal objective. Once the duality gap is sufficiently reduced, the sub-gradient routine is terminated. The availability of efficient commercial modeling software such as OPL Studio and CPLEX allows for larger instances of the problem to be tackled than previously possible.

Thesis Supervisor:
David Simchi-Levi,
Professor of Civil Engineering and Engineered Systems

Acknowledgements

Where is the wise man?
Where is the scholar?
Where is the philosopher of this age?
Has not God made foolish the wisdom of the world?

-- 1 Corinthians 1:20 (NIV)

I could never fathom myself one day graduating from a university as prestigious and well-regarded as MIT. I claim no credit for myself. Instead, I am fortunate to have the support and guidance of the following people, to whom I am ebulliently grateful--

Professor David Simchi-Levi, my advisor, for his guidance, direction, patience and understanding, and whose forthcoming words of encouragement kept me plodding on;

Dr Diego Klabjan, University of Illinois at Urbana-Champaign, for the many hours afforded sharing his academic knowledge and practical experience;

Liang Ping Ku and **Hein Thuan Loy**, PSA Corporation Operations Planning Department, for providing an excellent research topic and insight into real-life industry practices;

Laura Rose and **Jocelyn Sales**, course administrators, who were most committed to ensuring that our student experience in MIT was smooth and problem-free;

Singapore-MIT Alliance and the **Singaporean government**, for sponsoring the SMA Graduate Fellowship;

Yimin, my wife and best friend, for your love so unconditionally given in the last 9 years I've known you, and whom I consider the greatest blessing in my life;

And finally, my parents, **Lye Heng** and **Susan Choo**, sister, **Sabrina**, and brother, **Samuel**, who have given me everything.

Contents

List of Figures

Figures

Tables

Chapter 1

Introduction

1.1 Overview of Container Terminal Operations

Today, over 60% of the world's deep-sea cargo is transported in containers onboard ocean-going vessels, and some routes between economically strong and stable countries are containerized up to 100% [1]. The global growth rate for container port throughput in 2002 was reported to be 9.2%, with port traffic reaching a total of 266.3 million TEUs (twenty-foot equivalent units) [2]. Due to this increasing global demand for containerized marine shipping, container terminals have become important components in global logistics and transportation networks.

A container terminal serves as an interface between land and sea transportation. Its main functions are to receive outbound (export) containers from shippers for loading onto vessels, and to discharge (unload) inbound (import) containers from vessels for picking up by consignees [3]. These terminals also have storage yards for the temporary storage of containers. Container terminals are considered essential infrastructure because they are highly capital-intensive, and specialized equipment are needed to handle and transport containers within the port system; for example, a quay crane can cost upwards of US$10 million.

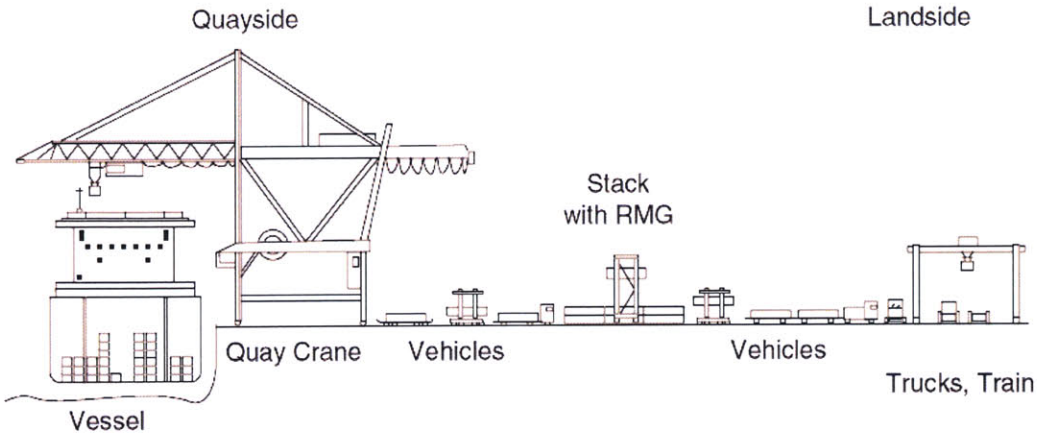


Figure 1-1. The two operational interfaces of a container terminal system

Container terminals can generally be described as having 2 main operational interfaces, *quayside* and *landside*. Quayside activities deal with the loading and unloading of ships, while landside involves loading and unloading of containers on or off external trucks, trains or yard storage locations. Some of the equipment and resources put to use in both interfaces are shown in Figure 1-1. The transportation of containers between the yard storage locations and the quayside is done primarily by trucks (sometimes known as *prime movers*) or automated guided vehicles.

The container terminal also has a storage yard which is usually divided into rectangular regions, known as *container blocks*. Each container block has about six rows for storing containers in *stacks*, with an additional lane for truck passing. A row may

Figure 1-2. A RTG yard crane placing a container on a truck for transport to the quayside

have up to 20 stacks placed end-to-end, each of which can be up to 6 levels high. These blocks are served by yard cranes of which there are 3 types, either rail mounted gantry cranes (RMG), rubber tired gantry cranes (RTG) or more recently, overhead bridge cranes (OHBG). Each type of yard crane confers different advantages to port operators; for example, the RTG cranes can be moved from block to block while the RMG and OHBG cranes are fixed, and the use of OHBG cranes allow for increased stack heights. Yard cranes remove and place containers from the stacks directly onto trucks which park in the passing lane while the transfer occurs. Traffic congestion caused by high rates of loading and unloading containers from a particular block is a concern explored in the chapter 3.

Upon a vessel's arrival at the terminal, a container vessel is assigned to a berth for the loading and discharging of containers. Discharged containers are placed onto trucks by quay cranes for transportation to pre-determined storage locations in the yard, awaiting pickup from the local consignee or trans-shipment onto another vessel. Yard cranes lift containers from the trucks onto their assigned stacks; the trucks are then recycled back into usage for other jobs. Similarly for vessel loading, external customers bring their outbound containers into the port using their own trucks and are instructed to which yard storage location their containers have been assigned. Yard cranes remove the container from the customer's truck onto the stack for storage. When the designated vessel arrives, the yard cranes place the export containers onto trucks for transportation to the vessel area, where they are loaded onto the ship by quay cranes.

The operations of a major Asian container terminal are broadly described in [4,5], while the interested reader is directed to [6] for a general overview of various attributes of container terminal operations. This thesis focuses on quayside operations; relevant terminologies and resources are elaborated upon in the next few sub-sections.

1.1.1 Containers

Containers are steel boxes of dimensions 20×8×9.5 or 40×8×9.5 feet. The unit of measurement for the throughput of 20-ft containers is commonly known as a TEU (twenty foot equivalent unit). Each 40-ft container is counted as 2 TEUs.

The advantage of containerized cargo is that they can be loaded and discharged with fewer crane moves and in a shorter time compared to bulk shipment cargo. Because of the uniformity in sizing, the time needed to handle a container is approximately constant if the equipment used and all other factors are equal. Furthermore, the use of standard-sized cargo streamlines the scheduling and controlling of the flow of goods. Other advantages include the protection of cargo against weather and accidental damage. Also, cargo contents do not have to be unpacked and repacked at each point in transfer resulting in a more rapid handling of freight.

1.1.2 Vessels Structure and Ship bays

A vessel is divided along its length into segments known as *bays*. Within each bay, the vessel is split vertically into 2 parts, the *deck* and the *hatch*, separated by a hatch cover. A drawing of the structure of a generic vessel is shown in Figure 1-3. Each bay can accommodate several rows of containers across and several tiers deep. It is not uncommon for some larger-capacity vessels today to have up to 25 bays, carrying a total of over 9,000 TEUs. These deep-sea vessels have on deck, containers stowed 8 tiers high and 17 rows wide, and in the hatch, 9 deep and 15 rows wide.

A bay is 20-ft wide and can be loaded by 40-ft containers or 20-ft containers. 20-ft container bays are usually odd-numbered, while 40-ft container bays, which occupy twice the length of the former, are evenly-numbered. For example, bays numbered 5 and 7

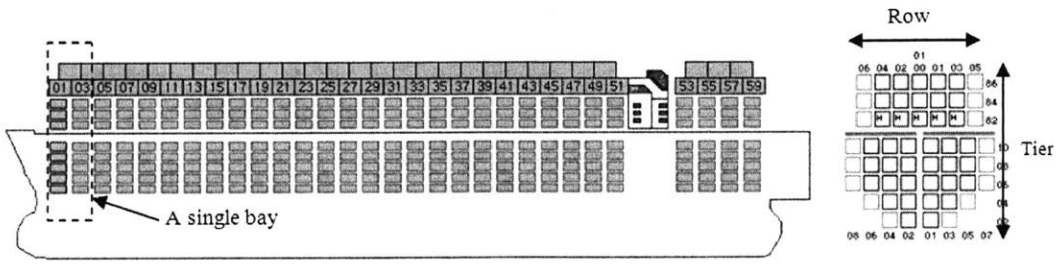


Figure 1-3. Horizontal and vertical cross-section of a typical container vessel

maybe joined together for loading of 40-ft containers and renamed bay 6.

1.1.3 Quay Cranes

Quay cranes (QC) are, in general, relatively immobile compared to yard cranes which can serve a large region in the storage area. QCs have 4 legs each and the space in between each of the 2 rows of legs is divided into traffic lanes for trucks to pass under. A trolley moves along the arm of the crane and is equipped with a *spreader*, which is used to pick up containers from the vessel or truck.

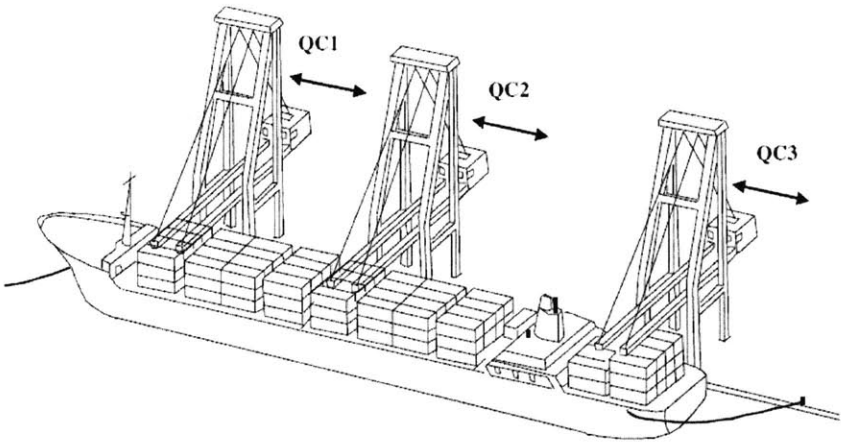


Figure 1-4. Drawing of quay cranes serving a vessel, with a clearance separating each adjacent crane

The width of a typical QC is 25.8 meters, and its practical performance is in the range of 22-30 boxes/hour. It is usual practice for up to 5 QCs to be allocated to large vessels, and up to 2,000 TEUs to be loaded and discharged per vessel in large ports. Port operators usually enforce a work rule that a certain distance, known as *clearance,* between two adjacent working QCs must be observed for safety reasons and unobstructed operation. QCs run on tracks parallel to the berth line; this horizontal movement is known as *gantrying*. Gantrying into position is usually completed in under a minute, lesser time than needed to handle a container.

Trucks queue up under their allocated QCs, and wait for their container to be picked up. For most efficient operation, trucks should always be available to serve the QCs with their next loading job such that a QC is never idle with jobs remaining. This predicates

that there are sufficient number of trucks serving the QCs, no traffic congestion in the port road network and no overloading of the yard cranes at the storage blocks which may delay the arrival of trucks at the QCs. This assumption is made in chapter 2 during problem modeling.

1.2 Problem Motivation and Description

In an increasingly competitive industry, ports have to ensure efficiency in their management of yard resources. Table 1-1 shows the throughput of the top five container ports in 2001-2003 [2]. Efficient port management involves making a variety of inter-related operational decisions to achieve a range of goals, some of which include the minimization of berthing time of vessels, resources needed for handling the work-load, congestion on the roads, and the maximization the use of limited yard storage space. Objective methods involving optimization techniques are necessary to support these decisions to allow for further improvements in efficiency, which benefits the terminals by allowing them to handle a higher volume of containers a day.

Table 1-1. Top 5 container terminals globally and their throughput (in millions), 2001-2003

Port	TEUs 2003	TEUs 2002	TEUs 2001
Hong Kong	20.82	19.14	17.80
Singapore	18.41	16.94	15.57
Shanghai	11.37	8.81	6.33
Shenzhen	10.70	7.61	5.08
Busan	10.37	9.45	8.07

One of the main goals is to minimize the length of time the ship is berthed in port, or *makespan*. An industry estimate puts the cost of a ship being berthed at port to US$1,000 an hour [7]. An important quayside factor which directly impacts the vessel makespan is the way cranes are scheduled to load and discharge containers from the vessels, which is a step in *stowage planning*.

The objective of stowage planning is to achieve an efficient and smooth discharge, re-stowage and loading of containers on vessels to obtain an expeditious on-time turnaround of vessels. It is carried out hours or days before the vessel's arrival and is a fundamental part of terminal management. The steps in stowage planning differ from port to port, but

for most it covers import and export planning, input of stowage instructions, crane split and sequencing, yard slotting and vessel stability checks (See [4]).

The main objective of the crane split and sequencing problem is to partition all the loading and discharging jobs among the allocated QCs, and decide the order at which the jobs are to be executed. Before the berthing of the vessel, the shipping company usually provides a work instruction, called the *load profile*, which details the precise location on the ship and exact identity of the containers which are to be loaded or discharged. Crane split and sequencing usually occurs immediately after a ship is assigned a berthing space, a fixed number of QCs are allocated to work on the vessel and the load profile and storage location of each import or export container in the yard is known.

1.3 OPL and OPL*Script*

OPL, or Optimization Programming Language, is a high-level modeling language for combinatorial optimization that simplifies these optimization problems substantially. OPL is part of a larger system that also includes OPL*Script*, the OPL component library and a development environment known as *OPL Studio*. It provides support for modeling linear and integer programs and provides access to state-of-the-art linear programming algorithms [8]. In addition, OPL supports the powerful CPLEX algorithm for mixed integer programming in combinatorial optimization problems, which are known to be *np*-hard in general.

OPL*Script* is a script language for composing and controlling optimization models which combines high-level data modeling facilities of modeling languages with novel abstractions to simplify the implementation of complex optimization applications [9]. OPL*Script* treats models like first-class objects, which allows modelers to exercise a degree of control over a model and state concisely many applications that require the solving several instances of the same model.

The version number of OPL Studio used in the implementation of the crane split and sequencing problem is 3.7; it has an built-in CPLEX version of 9.1. The author has found

useful the close similarity of OPL*Script* syntax with C/C++, the user-friendly graphical interface of OPL Studio for creating and modifying model and script files, the use of the *open array* whose size can be dynamically increased or decreased at runtime, and the ability to solve several repetitive, interacting instances of an optimization model in a particular sequence.

1.4 Literature Review

The QC scheduling program was first highlighted by Daganzo [10], who proposed an exact linear IP formulation for loading a few ships, and a principle-based heuristic approach for loading a larger number of ships. Available QCs are assigned to ship bays at discretized time periods. The problem, with minimization of makespan as its objective, is solved using enumerative techniques for up to 3 ships. Both the static case, where no other ships arrive in the planning horizon, and dynamic case are considered. Peterkofsky and Daganzo [11] discuss a branch-and-bound algorithm based on the same formulation in [10] to give exact solutions in reduced time. Both papers assume that only a single QC can work at a bay at any time.

Kim and Park [12] similarly discuss the QC scheduling problem, but include crane interference and precedence constraints in the study and a fixed departure time for the vessel. Their study further assumes that there may be multiple tasks or container clusters within a single bay, as opposed to [10] where a bay is considered the smallest positional unit. A branch-and-bound and heuristic search algorithm is proposed to obtain the optimal solution to the problem. The same authors discuss in another study [13] an IP model for scheduling berths and QCs, considering both problems to be dependant of each other. A two-phase solution procedure is suggested for solving the model in which the 1st phase determines the berthing position/time and number of QCs assigned to each vessel, while the 2nd phase produces a schedule for each QC. The detailed assignment of allocated QCs to individual ship bays, however, is not covered in the paper.

Bish [14] develops a heuristic algorithm based on formulating the problem as a three-fold trans-shipment problem – determining a storage location for each unloaded

container, dispatching vehicles to containers and scheduling the loading and unloading operation in cranes.

1.5 Thesis Objectives and Organization

This thesis extends the work done by Daganzo [10] incorporating industry-practice constraints into the model. This is critical for practical application in an industrial setting. The additional constraints were formulated based on the operational practices of a mega-container terminal and they include QC clearance, ordering (QCs cannot 'cross' one another) and yard congestion constraints. Details of these constraints will be discussed in later chapters. Both the single-ship case and the multiple-ship case, where up to 10 ships can berth at different times throughout the planning horizon, are tackled.

The main objective of the thesis is to utilize today's optimization applications to push the boundaries of problem size and account for modern vessel capacities. It has been reported that there was a 2,360 fold speed-up of CPLEX LP code from 1988 to 2002, and within the same period, an additional 800 fold speed-up is obtained due to advances in hardware [15]. Problem-specific methods such as column generation and Lagrangian relaxation, applied to large-scale IPs, should improve on computational performance delivered by CPLEX's default enumerative- and/or polyhedral-based algorithms.

In chapter 2, we discuss the mathematical formulation for the single-ship model, and present a branch-and-price algorithm to solve the problem to optimality. A heuristic is also proposed and is shown to produce effective solutions. Computational results of various techniques are shown and compared against CPLEX's performance. In chapter 3, the multiple-ship model is presented incorporating yard congestion constraints. These constraints are relaxed using Lagrangian multipliers and the problem decomposes by vessel into smaller and easier sub-problems. The sub-gradient optimization technique is used to obtain optimal multipliers. A branch-and-price method is proposed to resolve the Lagrangian sub-problem, while a primal heuristic is developed to obtain feasible upper bound solutions. Lastly, computational results of the Lagrangian relaxation approach are

presented. In chapter 4, we outline possible future work directions and summarize the findings of this thesis.

Chapter 2

Single-ship Model

2.1 Exact Mathematical Formulation

In this chapter, we propose a mathematical formulation of the crane split and sequencing problem for a single vessel that has a specific number of bays and container work-load. No other vessels are assumed to berth during the planning horizon. The number of QCs allocated to work on the vessel has been pre-determined in the planning process.

2.1.1 Problem Characteristics and Modeling Requirements

To model the problem, the entire *planning horizon* (i.e. maximum time in which all crane work has to be completed) is divided into small time units such that all time-related variables have integer time units. The length of each time period is the amount of time needed for a QC to load or discharge a standard 20-ft container. QCs can only move and be assigned to another (or the same) bay at discrete intervals.

The input data to the problem are – (1) number of QCs allocated to work on the vessel, (2) the number of bays in the ship and (3) the detailed distribution of work in all ship bays. The scheduling constraints for the single-ship model are described below:

1. A QC can only be positioned and/or work at a single bay at any time.
2. A QC must be positioned at least r bays away from any adjacent QCs on the left and right.
3. QCs cannot 'cross' each other and must be positionally ordered at all times.

The following assumptions are made:

(a) QC gantrying time is negligible, compared to the time it takes to move a container, and it can be ignored in the calculation of makespan.
(b) QCs cannot be added to or removed from vessel operation during the entire planning horizon
(c) All QCs are identical and have similar work rates
(d) There is no delay in trucks delivering containers to the QCs at the allocated time period.
(e) There is no distinction between the time required to move a 20-ft and 40-ft container.

2.1.2 Notation

The following notations are used for the mathematical formulation.

Indices:

j Bay number, in increasing order of their relative location on the vessel (i.e. left to right)
k Crane number, in increasing order of their relative position (i.e. left to right)
t Time period index, denoting the interval of time from t-1 to t

Parameters:

C Number of QCs allocated
H Number of bays in the vessel
f_j Number of containers to be loaded or discharged in bay j
T Total number of time periods in the planning horizon: set at $\sum_{j=1}^{H} f_j$, or the makespan of the vessel if only one QC was allocated
r QC clearance value, in terms of number of bays

Decision variables:

$x_{jk}(t)$ 1, if QC k is positioned at bay j at time period t; 0, otherwise

$\delta_{jk}(t)$ 1, if QC k is loading or discharging a container at bay j at time period t; 0, otherwise

$\gamma(t)$ Completion flag: 0, if all container jobs are completed after time period t; 1, otherwise

2.1.3 The Model

The single-ship model for the crane split and sequencing problem is as follow:

$$\text{Minimize}_{x,\delta,\gamma} -\sum_{t=1}^{T}\gamma(t) \tag{2.1}$$

Subject to

$$\sum_{j=1}^{H} x_{jk}(t) = 1, \qquad \forall k,t = 1...T \tag{2.2}$$

$$\sum_{k=1}^{C} x_{jk}(t) \le 1, \qquad \forall j,t = 1...T \tag{2.3}$$

$$M\left(1-\sum_{k=1}^{C} x_{jk}(t)\right) \ge \sum_{l=\max\{1,j-r\}}^{j-1}\sum_{m=1}^{C} x_{lm}(t), \qquad j = 2..H, t = 1..T \tag{2.4}$$

$$x_{jk}(t) \le \sum_{l=j+1}^{H} x_{l,k+1}(t), \qquad j = 1..H-1, k = 1..C-1, t = 1..T \tag{2.5}$$

$$\sum_{k=1}^{C-1} x_{Hk}(t) = 0, \qquad t = 1..T \tag{2.6}$$

$$\sum_{k=1}^{C}\sum_{t=1}^{T}\delta_{jk}(t) = f_j, \qquad \forall j \tag{2.7}$$

$$\delta_{jk}(t) \le x_{jk}(t), \qquad \forall j,k,t = 1...T \tag{2.8}$$

$$\gamma(t) \le \frac{\sum_{k=1}^{C}\sum_{l=1}^{t}\delta_{jk}(l)}{f_j}, \qquad \forall j,t = 1...T \tag{2.9}$$

$$\mathbf{x},\boldsymbol{\delta},\boldsymbol{\gamma} \in \{0,1\} \tag{2.10}$$

Constraint (2.2) ensures that each QC must be positioned at a bay at any time, while (2.3) denotes that only one QC can be positioned at any bay at any time. Constraint (2.4) enforces the QC clearance condition, stating that if any QC is positioned at a particular bay, all other QCs are restricted from being positioned r bays to the left, unless the bay in question is less that r bays from 1st bay. Constraints (2.5) and (2.6) describe the QC

'ordering' condition. It states all 'higher-numbered' QCs must be positioned to the right of a respective QC, and that only QC C is allowed to work on the right-most bay. Constraint (2.7) ensures that all required container jobs are completed within the planning horizon. Constraint (2.8) states that a QC cannot be working at a bay if it is not positioned there. Constraint (2.9) defines the work completion flag for bay j. The objective function ensures that when the value on the right-hand-side of (2.9) sums to '1' for all H bays at time period t, $\gamma(t)$ will take the value of '1' as well. (2.10) states that all the decision variables are binary. For δ's, it means that each QC can only work on 1 container in each bay at any time.

The objective function evaluates the value of the makespan (i.e. $T - \sum_{t=1}^{T} \gamma^*(t)$). The feasible region is defined by all possible ways of assigning each QC's position (x's) and work-status (δ's) to a particular bay for all periods in the planning horizon.

2.1.4 Strengthening of the Model

It is well-known that disaggregating or introducing additional constraints may help to speed up the solving of the problem [16]. These additional relations improve the optimal solution of the LP relaxation by excluding some fractional solutions.

The QC clearance constraint (2.4) is reformulated as follows:

$$1 - x_{jk}(t) \geq \sum_{l=\max\{1,\, j-r\}}^{j-1} x_{lm}(t), \qquad j = 2..H, \forall k, m = 1..C, t = 1..T \tag{2.4a}$$

$$1 - x_{jk}(t) \geq \sum_{l=j+1}^{\min\{j+r,\, H\}} x_{lm}(t), \qquad j = 1..H-1, \forall k, m = 1..C, t = 1..T \tag{2.4b}$$

The disaggregation of (2.4) in the k index on the right-hand-side removes large-integer M from the constraint, which is known to cause problems in CPLEX. (2.4b) has the 'mirror' effect of (2.4a) – if any QC is positioned at a particular bay, no other QCs can be positioned r bays on the *right*.

The QC ordering constraints (2.5) and (2.6) are also reformulated as follow:

$$x_{jk}(t) \le \sum_{l=j+1}^{H} x_{l,k+1}(t), \qquad j = 1..H-1, k = 1..C-1, t = 1..T \tag{2.5a}$$

$$x_{jk}(t) \le \sum_{l=1}^{j-1} x_{l,k-1}(t), \qquad j = 2..H, k = 2..C, t = 1..T \tag{2.5b}$$

$$x_{jk}(t) = 0, \qquad \{\forall k \mid k \neq 1\}, j = 1..k-1,, j = 1..k-1, t = 1..T \tag{2.6a}$$

$$x_{jk}(t) = 0, \qquad \{\forall k \mid k \neq C\}, j = H-(C-k)+1..H, t = 1..T \tag{2.6b}$$

Constraint (2.5b) has the 'mirror' effect of (2.5b) - all 'lower-numbered' QCs must be positioned to the left of a given QC. Constraints (2.6a) and (2.6b) further restrict positioning some QCs at bays at the extreme ends of the vessels so that clearance is maintained.

While these additional constraints are indeed redundant in terms of IP feasibility, they significantly prune the LP feasible space. We observe in §2.4 that the enlarged model, while having about 1.5 times the number of constraints, has shorter computational times.

2.1.5 Difficulty of the Problem

Solving combinatorial optimization problem to optimality can be a difficult task and many have been proven to be *np*-hard [17]. The difficulty arises from the fact that the feasible region of the IP may not be a convex set and one must search a lattice of feasible points to find an optimal solution. Most commercial software, including CPLEX, employ enumerative approaches such as branch-and-bound to intelligently search the feasible space around the relaxed-problem solution. The running times of these algorithms, however, are not bounded by a polynomial in the size of the input.

We expect the single-ship model to be applied to vessel sizes in the order of 30 to 40 bays and a total container work-load of several hundred to a thousand TEUs. The number of decision variables x, δ and γ are $H \times C \times T$, $H \times C \times T$ and T respectively, and they may be in the range of hundred thousands, hence the (1) number of bays, (2) number of QCs allocated and (3) total workload determine the size of the problem and can affect its computational speed. We refer the reader to §2.4 for the tabulation of computation results, which show that attempting to solve the exact model in CPLEX leads to excessive

solution times (even with strengthening). This leads us to the next two sections, which propose a heuristic and branch-and-price method to solve the large-scale problem in economically-feasible computational times.

2.2 Heuristic Solution Approach

Heuristic are employed to obtain good but not necessarily optimal solutions in IPs. Typically, they come with no solution-bounds guarantee, but are justified by their performance in practice, i.e. they may be the only way to quickly provide a usable solution to very difficult optimization problems.

2.2.1 Scheduling Principles to Achieve Optimality

A logical approach for the heuristic would be to decide on 'good' bay assignments one period at a time, instead of optimizing across the entire planning horizon. It is observed that the minimum time needed to clear the remaining work load at each period, the *remaining makespan*, is independent of QC assignment decisions made in earlier periods.

It is clear that if the clearance constraints and the constraint that each bay cannot be worked on by more than 1 QC are ignored, the remaining makespan at period t, $RM(t)$, would be simply be $\left\lceil \frac{\sum_j l_j(t)}{C} \right\rceil$, where $l_j(t)$ is the work load remaining in bay j at period t. However since only 1 QC can work on a bay at any time, a possible value for $RM(t)$ would be $\max\{l_j(t) \mid \forall j\}$. The bay with the maximum remaining work-load is labeled the *maximal bay.*

Therefore, a lower bound can be established for *RM(t)*:

$$\max\left\{\max\{l_j(t) \mid \forall j\}, \left\lceil \frac{\sum_j l_j(t)}{C} \right\rceil\right\} \le RM(t) \tag{2.11}$$

(2.11) states that either the remaining workload in the maximal bay or the average QC work rate will be the minimum possible $RM(t)$, whichever is greater. To illustrate each case, consider the following example. A ship has 5 bays and is handled by 2 QCs. If bays 1 to 4 each have 1 container left while bay 5 has 4 containers left, the remaining makespan would be at least 4 periods (i.e. $\max\{l_j(t) \mid \forall j\}$). Alternatively if bays 1 to 5 all have 1 container left each, the remaining makespan will instead be 2 (i.e. $\left\lceil \frac{\sum_j l_j(t)}{C} \right\rceil$).

In each period, we want to make assignment decisions so that the lower bound on $RM(t+1)$ is reduced. To do this, we attempt to reduce the values of both $\max\left\{l_j(t+1) \mid \forall j\right\}$ and $\left\lceil \frac{\sum_j l_j(t+1)}{C} \right\rceil$ by observing 2 broad scheduling principles in each time period:

1. The maximal bay for each time period must always be handled.
2. Remaining QCs should not be idle. They are assigned to work on other bays with heavy work-loads.

However, the lower bound on $RM(t)$ may ultimately not be reached because clearance constraints can force some QCs to remain idle while there is still work to be done. This may occur especially in the last few periods of work and will result in sub-optimal makespan results. However, computational tests demonstrate that the degradation from the optimal solution is minimal.

2.2.2 Description of the Algorithm

The following describes the overall heuristic procedure:

Step 1: Initialize the value for $l_j(t)$ for the 1st time period, i.e. Let $l_j(1) = f_j$ for all j. Set t=1.

Step 2: Rank all bays in terms of remaining work-load for current period, t.

Step 3: Assign QCs to bays for current period t based on the scheduling principles described in §2.2.1. (The exact QC assignment model executed for each period is described in §2.2.3.)

Step 4: Obtain the value of $l_j(t+1)$ by subtracting away the work done in each bay from $l_j(t+1)$.

Step 5: If $\sum_j l_j(t+1) = 0$, there is no remaining work and algorithm terminates with t as the makespan. Otherwise, increment t by 1 and repeat from Step 2.

2.2.3 The Model for assigning QCs in each period

This section describes the OPL model that is repeatedly called in each period until there are no more jobs left. The remaining work-load in each bay, $l_j(t)$, and the mapping of the work-load rankings to bay numbers, $r(j')$, are imported into the model. The model assigns QCs to bays based on the two scheduling principles described in §2.2.1, and is represented as follows:

$$\text{Maximize}_{x,\delta} \sum_{j'=2}^{H} \sum_{k=1}^{C} (H - j')\delta_{r(j'),k} \tag{2.12}$$

Subject to

$$\sum_{j=1}^{H} x_{jk} = 1, \qquad \forall k \tag{2.13}$$

$$\sum_{k=1}^{C} x_{jk} \le 1, \qquad \forall j \tag{2.14}$$

$$1 - x_{jk} \ge \sum_{l=\max\{1,j-r\}}^{j-1} x_{lm}, \qquad j = 2..H, \forall k, m = 1..C \tag{2.15a}$$

$$1 - x_{jk} \ge \sum_{l=j+1}^{\min\{j+r,H\}} x_{lm}, \qquad j = 1..H-1, \forall k, m = 1..C \tag{2.15b}$$

$$x_{jk} \le \sum_{l=j+1}^{H} x_{l,k+1}, \qquad j = 1..H-1, k = 1..C-1 \tag{2.16a}$$

$$x_{jk} \le \sum_{l=1}^{j-1} x_{l,k-1}, \qquad j = 2..H, k = 2..C \tag{2.16b}$$

$$x_{jk} = 0, \qquad \{\forall k \mid k \ne 1\}, j = 1..k-1,, j = 1..k-1 \tag{2.16c}$$

$$x_{jk} = 0, \qquad \forall k \mid k \ne C\}, j = H-(C-k)+1..H \tag{2.16d}$$

$$\sum_{k=1}^{C} x_{jk} = 1, \qquad j \in \max\{l_j \mid \forall j\} \tag{2.17}$$

$$\delta_{jk} \le x_{jk}, \qquad \forall j, k \tag{2.18}$$

$$\delta_{jk} \le l_j, \qquad \forall j, k \tag{2.19}$$

$$\sum_{k=1}^{C} \delta_{jk} = 1, \qquad j \in \max\{l_j \mid \forall j\} \tag{2.20}$$

$$\mathbf{x}, \boldsymbol{\delta} \in \{0,1\} \tag{2.21}$$

The decision variables are x and δ, and they have the same definition as in the single-ship model. (2.13) to (2.16) and (2.18) are identical to QC position constraints in the single-ship model. Constraints (2.17) and (2.20) ensure that a QC is positioned at and working on the maximal bay. Constraint (2.19) states that no work can be done in a bay if there are no remaining containers to handle.

The objective function adds a 'weight' to all δ's relative to their work-load rankings. Since bays with higher rankings are given heavier weightage, the maximization function

will attempt to allocate remaining QCs to work on bays with heavier work-loads, ensuring that no QC remains idle while there is work remaining.

This model is smaller that the exact formulation because decision variables are not indexed by t. Hence, we can expect that overall solution time will be faster despite it being repeatedly run.

2.3 Branch-and-price Solution Approach

2.3.1 General Framework

The main motivation behind the use of the method is the possibility of tightening the existing LP relaxation by reformulating it into a problem which involves a huge number of variables. Columns are left out of the LP relaxation because there are too many to handle efficiently and most of them will have their associated variable equal to zero in the optimal solution anyway. The hope is that such a scheme will converge faster than CPLEX's LP-based branch-and-bound algorithm. Barnhart *et. al.* [18] provide a good exposition of the branch-and-price methodology.

The existing, compact single-ship model is re-written into a '0-1' set-covering problem. This is motivated by the pioneering work of Gilmore and Gomory [19] on the cutting stock problem. Each row represents a ship bay and each column, QC position-bay assignments for a single time period. A column has H (number of ship bays) rows, each having a value of '1' if a QC is positioned at that particular bay.

This synthesis of column generation and branch-and-bound is known as *branch-and-price*. The proposed branch-and-price procedure is an adaptation of the basic branch-and-bound procedure based on LP relaxation, which is reviewed in [20]; the branch-and-bound methodology *per se* is well-established in literature and will not be discussed in this thesis. Branch-and-price solves to optimality the problem involving QC position-assignments (x's); later obtaining the QC work-assignments (δ's) from x^*'s is trivial (§2.3.4).

The following details the procedure to solve the single-ship model:

Step 1: Initialize a problem set pool, $\{IP\_Pset\}$. Place the root problem node into $\{IP\_Pset\}$ and initialize an empty constraint pool for it. Set $bestUB = \infty$ and $z_{best} = 0$.

Step 2: Generate an initial column pool for the restricted master problem (RMP) that can produce a feasible solution.

Step 3: If $\{IP\_Pset\} = \varnothing$, terminate the algorithm with z_{best} as the optimal column-variable integer solution with y_{best} as the optimal makespan.

Step 4: Choose and delete the latest problem node, $IP\_Pset(k)$, away from $\{IP\_Pset\}$.

Step 5: Solve the RMP using the column and constraint (branching rules) pools associated with the current problem node, $IP\_Pset(k)$.

Step 6: If RMP is infeasible, go to step 3. Otherwise, extract the optimal dual variables from the RMP, insert into the PP and solve.

Step 7: If the column with the minimum reduced cost is less than 0, insert that into the RMP column pool and repeat from Step 5. Otherwise, let y^k be the RMP objective value and z^k be the column-variable solution.

Step 8: If $y^k \geq y_{best} - 1$, go to Step 3. If z^k's are all integer, set $y_{best} = y^k$ and $z_{best} = z^k$ and go to Step 3.

Step 9: Find the 1st fractional variable in z_i^k. Insert 2 new problem nodes to $\{IP\_Pset\}$ and add $z_i^k = \lfloor z_i^k \rfloor$ and $z_i^k = \lceil z_i^k \rceil$ respectively to the constraint pools of each. Go to Step 3.

Step 10: Take the non-zero column-variables in z_{best} and set x's. Reset some δ's to zero so that work completion constraints are met.

The constraint pool of each problem node consists of the accumulation of all branching constraints from the root to the current node of the branch-and-bound tree. At step 4, we must consider which problem node to explore in the next branch-and-bound iteration. Because feasible solutions are found deep inside the search tree, a depth-first search is adopted where the most recently created problem node is chosen. Steps 2-9

describe a standard branch-and-bound algorithm. Steps 5-7 explain the column generation and pricing route that is executed at every node of the tree (§2.3.2). Step 8 describes a pruning method, while step 9 refers to standard variable branching of fractional column-variables (§2.3.3).

2.3.2 Column Generation and Pricing Problem

There is an exponential number of feasible QC position-assignment 'patterns', therefore the column generation form of the problem (the *master problem*) contains a huge number of columns. Hence, it is necessary to work with an LP relaxation involving only a small subset of the columns, known as the *restricted master problem* (RMP). New columns are generated as needed when their reduced costs are negative and are candidates to improve the objective function. The *pricing problem* (PP) uses dual variables from the RMP to look for non-basic columns, with feasible QC position-assignments, which have the minimum reduced cost. If the minimum reduced cost is less than or equal to 0, we have proven that the current solution to the RMP is also optimal for the master problem.

The algorithm moves repeatedly between solving the RMP and the PP until no more columns with negative reduced costs can be found. RMP and PP are solved with CPLEX. The solution of the final run of the RMP becomes the lower bound value of a node in the branch-and-bound tree.

An initial column pool which gives a feasible LP solution

To start the column generation scheme, an initial RMP has to be provided which has a feasible LP to ensure that proper dual multipliers are passed into the PP. A pool of columns is generated for the initial RMP such that it has a feasible (but most likely non-optimal) LP solution.

Any set of columns which 'covers' all bays with non-zero work-loads will suffice. Figure 2-1 shows columns of the initial basis used for a ship with 2 bays, 2 QCs allocated

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} \geq \underbrace{\begin{bmatrix} 3 \\ 2 \\ 2 \\ 4 \\ 6 \\ 8 \end{bmatrix}}_{f_j}$$

Figure 2-1. Example of an initial feasible column pool for a ship with parameters $H = 6$, $C = 2$ and $r = 1$

and clearance of 1. Note the feasibility of the columns in terms of the clearance constraints, i.e. at least 1 'empty' row always separates 2 active rows.

The pseudo-code of the procedure for generating the initial column pool is as follows:

---

set *craneposition* = 1
set *numCols* = -1
set *config*[*numCols*,1..*H*] = 0
repeat
    numCols = *numCols* + 1
    Increase size of *config* by 1 column
    Re-set rows of new column to '0'

    for *k* = 1..*C-1* **do**
        config[*numCols*,*craneposition*+*k*(*r*+1)] = 1
    end for

    if *craneposition* + *k* (*r* + 1) = *H* **then**
        break out of repeat loop
    end if
end repeat

---

config is an open array which can be dynamically re-sized at runtime. It stores the '0-1' set-covering matrix of the RMP which has *numCols* number of columns. The index *craneposition* marks the bay position of the first QC in the current column, while the expression *craneposition* + *k* (*r* + 1) indicates the bay position of the last QC.

The pseudo-code for the column generation and pricing routine is as follows:

```
repeat
    Solve the RMP model

    if RMP is feasible then
            bc = Basis information from RMP
            dual[j] = Dual variable associated with row j of set-covering constraint in RMP
            Import dual[j] into PP
            Solve PP model
            if objective value of PP < 10^-5 then
                    x[j] = Feasible QC position 'pattern' with minimum reduced cost
                    numCol = numCols + 1
                    Increase size of config by 1 column
                    config[numCol,j] = x[j]

                    Reset RMP model.
                    Set initial basis of RMP with bc
                    Import new column pool, config
            else
                    break out of repeat loop
            end if
    else
            break out of repeat loop
    end if
end repeat
```

A simplex-based algorithm is used to solve the RMP LP. To reduce the computational work of repeatedly solving the RMP, the basis from the optimal set of columns from previous iterations of the RMP is used as the initial basis for solving the RMP in the next iteration. The basis information is stored as *bc* in the pseudo-code.

Restricted Master Problem (RMP)

The purpose of the RMP is to (1) provide dual variables for the PP, and to (2) determine if the current set of QC position-assignment 'patterns' provide an optimal LP solution or not.

(*RMP*):

$$\text{Minimize}_z \sum_{i \in I} z_i \tag{2.22}$$

Subject to

$$\sum_{i \in I} \sum_{j \in i} z_i \geq f_j, \qquad \forall j \tag{2.23}$$

$$z_i \geq 0, \qquad \forall i \tag{2.24}$$

$$z_{i'} \geq U, \qquad \forall i' \in \{P \mid \text{Ceiling constraints}\} \tag{2.25}$$

$$z_{i'} \leq L, \qquad \forall i' \in \{P \mid \text{Floor constraints}\} \tag{2.26}$$

The following is a description of notations in the RMP:

I	Set of feasible QC position-assignment 'patterns' in the column pool
P	Set of branching constraints in the constraint pool of this problem node
z_i	Number of times QC position-assignment 'pattern' i is used
f_j	Work-load in bay j

The cost of including each column is '1' since the makespan is incremented by 1 when a column is used. The objective function (2.22) minimizes the total number of columns needed to meet work-load requirements. (2.23) depicts the set-covering constraints. Constraint (2.24) enforces only the non-negativity of z_i, but allows columns to be used multiple times. (2.25) and (2.26) are all the branching constraints associated with the current problem node, and its effect is to discard regions of the feasible space that included fractional variables obtained in previous nodes of the search tree.

Pricing Problem (PP)

The role of PP is to provide a column that prices out profitably or to prove that none exist. The optimal dual variables associated with constraint (2.23), p, from the RMP are imported into PP.

(*PP*):

$$\text{Minimize}_x\, 1 - \sum_{j=1}^{H} p_j x_j \tag{2.27}$$

Subject to

$$\sum_{j=1}^{H} x_j = C \tag{2.28}$$

$$C\left(1 - x_j\right) \le \sum_{l=j+1}^{\min\{j+r,H\}} x_l, \qquad j = 1..H-1 \tag{2.29a}$$

$$C\left(1 - x_j\right) \le \sum_{l=\max\{1,j-r\}}^{j-1} x_l, \qquad j = 2..H \tag{2.29b}$$

$$\mathbf{x} \in \{0,1\} \tag{2.30}$$

The decision variable, x_j, is '1' if a QC is positioned at bay j and '0' otherwise. All other notations have similar meanings as the single-ship model. The objective function (2.27) arises from the calculation of the reduced cost of a non-basic column, indexed by i, in an LP: $c_i - \left(c_B^T B^{-1}\right) A_i$, where c_i is the cost of a non-basic column i ('1' in this case), $\left(c_B^T B^{-1}\right)_j$ is equivalent to p_j (the optimal dual variables associated with constraint (2.23) in the RMP), and A_i is the non-basic column vector in i represented by x_j's in the PP.

Constraint (2.28) states that all QCs must be positioned at a particular bay, therefore there are a total of C 'active' rows. (2.29) enforces QC clearance constraints similar to that of the single-ship model. QC ordering constraints (such as in (2.5) and (2.6)) are not necessary since QCs will be assigned in order of their numbering (in the k index) from the top to bottom of the column.

Column Generation at every node of branch-and-bound-tree

The LP relaxation solved by the column generation technique does not necessarily produce integer solutions, and applying a standard branch-and-bound procedure to the RMP with its existing columns does not guarantee optimality. There may be columns outside the current pool which may have negative reduced cost after branching in subsequent nodes occurs. Therefore, the branch-and-bound procedure must be modified

so that the column generation procedure is used at every node of the search tree to generate a lower bound.

2.3.3 Branching and Pruning

Branching

A valid branching scheme partitions the solution space in such a way that the current fractional optimal solution of a node is excluded, integer solutions remain intact, and finiteness of the algorithm is ensured [21]. The use of standard variable branching often creates problems along a branch where a particular variable has been set to '0'; in future iterations of PP, columns that were previously excluded may be re-generated causing cycling in the branch-and-bound algorithm. However in this case it would occur infrequently because the decision variables of the RMP, *z's*, are not binary. The standard variable branching rules will not force the fractional column out of the basis, hence the branched variables will have a reduced cost of zero (being basic in the LP) and automatically be excluded from being inserted into future column pools (refer to Step 7 in §2.3.1).

It is common to make important branching decisions at the top of the branch-and-bound tree. Because all columns have equal cost, in deciding on which fractional column-variable in the LP solution to branch on for every node in the tree, it suffices to select the first fractional column found.

Pruning

Since the costs of all the columns are integral, it is possible to discard nodes which have optimal LP values that are greater than an integer away from the upper bound, i.e. $y^k > y_{best} - 1$ instead of $y^k > y_{best}$. This allows us to more easily disregard previously partitioned regions of feasible space in searching for the optimal solution, thus significantly speeding up the branch-and-bound procedure.

2.3.4 Calculating x's and δ's

The output of a successful run of the branch-and-price algorithm is z_{best}, the number of times each feasible QC position-assignment 'pattern' is used. From z_{best}, we can use RMP column pool data to form up values of x's, a decision variable of the original single-ship model, for each time period. Next, QCs are assumed to be working where they are positioned at all time periods. This is allowed through (2.8). We then arbitrarily re-set some δ's back to '0' so that the work requirement constraint (2.7) is met exactly (as opposed to being exceeded).

The following is the pseudo-code for the re-arrangement of data from z_{best} to x's:

```
for i = 1..numCols do
    set c = 1
    if z_best^i = 1 then
            for j = 1..H
                    if config[numCols,j] = 1 then
                            x_{j,c}(i) = 1
                            c = c + 1
                    end if
            end for
    end if
end for
```

The following is the pseudo-code for the extraction of δ's from x's:

```
forall j, k, t do
    δ_jk(t) = x_jk(t)
end for

Initialize wastage[1..H]
for j = 1..H do
    Calculate amount that exceeded work requirement constraints for bay j
    Store amount in wastage[j]

    for p = 1..wastage[j], k = 1..C do
            if δ_jk(p) = 1 then δ_jk(p) = 0 end if
    end for
end for
```

2.4 Computational Results

2.4.1 Test Problems

A data set with 12 different problem instances was created. The problem instances are arranged in order of increasing ship bay numbers, each with varying work-loads, number of QCs allocated and clearance requirements. The main objective of using this data set is to validate the accuracy of the exact model and the proposed algorithms in producing QC position- and work-assignments that adhere to all the constraints. The secondary purpose is to compare the computational performance and quality of solution of the heuristic and branch-and-price methods against that of the exact model solved with CPLEX. Table 2-1 summarizes the various parameters of each problem instance, and provides the total number of constraints and decisions variables of the aggregated and 'disaggregated' forms of the exact model.

Table 2-1. Description of the first data set of test problems

Problem No.	*H*	*C*	*r*	*T*	No. of constraints*	No. of variables*	Aggregated form - No of const.	Aggregated form - No. of var.
SS1-1	10	2	1	61	8184	2501	5378	2501
SS1-2	10	3	1	61	15687	3721	9343	3721
SS1-3	10	4	1	61	25508	4941	14406	4941
SS1-4	10	5	1	61	37647	6161	20567	6161
SS1-5	10	2	3	61	8184	2501	5378	2501
SS2-1	20	3	3	106	55882	12826	33198	12826
SS2-2	20	4	4	106	90968	17066	51536	17066
SS2-3	20	3	5	106	55882	12826	33198	12826
SS3-1	30	3	8	143	114001	25883	67669	25883
SS4-1	40	2	8	198	109732	31878	◊	◊
SS4-2	40	3	8	198	211306	47718	◊	◊
SS5-1	50	3	8	249	◊	◊	◊	◊

* Disaggregated form of the single-ship model
◊ Unknown – takes too long to execute model

A second data set is also created which contains problem instances with vessel sizes, work-loads and other parameters that are typically encountered in the operations of any major container terminal. The main purpose is of using this data set is to ensure that the proposed solution approaches can handle realistic industry conditions. Table 2-2 describes the parameters of this practical data set.

Table 2-2. Description of a second data set of practical test problems

Problem No.	*H*	*C*	*r*	*T*	No. of constraints*	No. of variables*
SSP-1	25	2	8	519	178561	52419
SSP-2	30	3	8	839	◊	◊
SSP-3	40	4	8	1385	◊	◊

* Disaggregated form of the single-ship model
◊ Unknown – takes too long to execute model

2.4.2 Results and Analysis

First data set

All computational tests performed for this thesis were carried out using a Pentium-IV 1.6 GHz PC running OPL Studio in the UNIX environment. 4 different experiments were carried out on all problem instances in the first data set; the aggregated and disaggregated form of the exact model, the heuristic and branch-and-price solution approaches were run. The optimized makespan and computation time results of tests are tabulated in Table 2-3.

Table 2-3. Output and computational performance of the exact model and proposed solution approaches on first data set

Problem No.	Exact makespan	Heuristic makespan	Br-n-Pr. makespan	Exact runtime (s)	Exact runtime (s)	Heuristic runtime (s)	Br-n-Pr. runtime (s) / No. BnB iterations
SS1-1	31	31	31	0.8769	1.118	0.091	0.062 / 13
SS1-2	21	21	21	0.9829	1.251	0.083	0.111 / 9
SS1-3	16	16	16	1.103	1.416	0.084	0.062 / 7
SS1-4	14	14	14	1.029	1.516	0.096	0.000 / 1
SS1-5	31	31	31	1.134	1.157	0.011	0.016 / 3
SS2-1	36	36	36	0.843	21.21	0.318	0.780 / 15
SS2-2	38	40	38	11.58	16.47	0.504	0.000 / 1
SS2-3	36	36	36	9.677	19.23	0.367	0.281 / 4
SS3-1	56	56	56	6.542	~ 120	1.047	0.000 / 1
SS4-1	99	99	99	25.89	◊	1.838	2.706 / 9
SS4-2	66	70	66	368.7	◊	1.826	18.84 / 43
SS5-1	◊	86	83	◊	◊	2.998	102.3 / 99

◊ Unknown – takes too long to execute model

CPLEX is able to solve the exact model within a reasonable time for problem instances up to 40 bays which has over 200,000 constraints and 50,000 decision variables. In contrast, for the aggregated form of the exact model, a solution was obtained in reasonable time only for problem instances of up to 30 bays. This validates the 'LP-tightness' of the strengthened model. Tests with the heuristic approach show that computational time does not exceed 3 seconds for all problem instances, a drastic

improvement over CPLEX's performance on the exact model. The makespan solution differs from the optimal makespan of the exact model and branch-and-price approach in several problem instances (i.e. sub-optimal solutions in SS2-2, SS4-2). The overall quality of the heuristic solutions, however, remains high. The branch-and-price algorithm runs the most rapidly on average among the 4 experiments conducted. In problem instances SS1-4, SS2-2 and SS3-1, integer solutions were obtained at the root node and no branching was necessary; computation time is almost negligible in these cases. Performance, in the branch-and-price approach, is also affected by the number of branch-and-bound iterations required.

Output from OPL*Script* is exported to Matlab, where the data is re-organized and a graphical, intuitive form of the solution is presented. Figures 2-2 to 2-4 shows solutions of the exact method, heuristic and branch-and-price approaches applied to several problem instances. The lines in the figures indicate the position of the QCs (x-axis) at each time period (y-axis), while the crosses show when they are at work; crosses at bay number 0 indicate that the QC is not working at that time period. Different colored lines represent the different QCs.

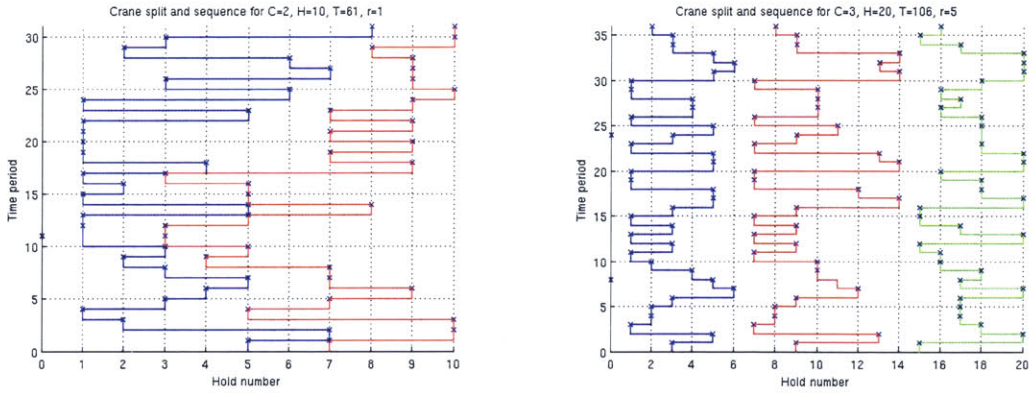


Figure 2-2. Solutions from the exact method applied to problem instances SS1-1 and SS2-3

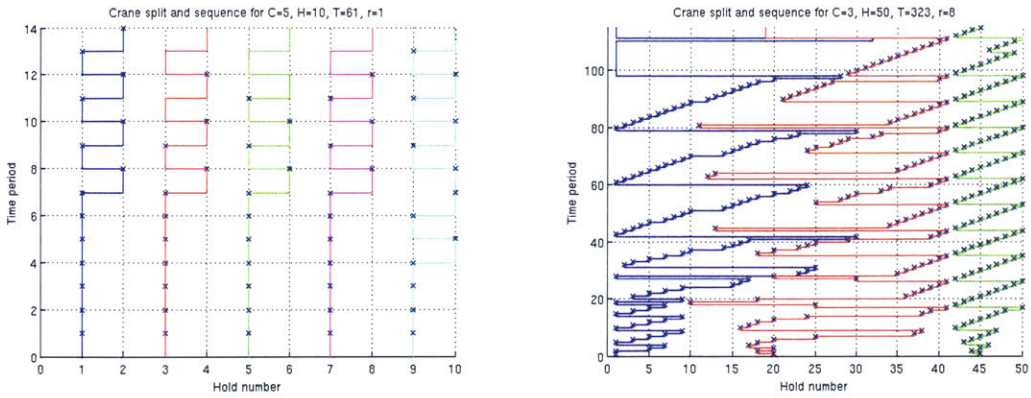


Figure 2-3. Solutions from the heuristic approach applied to problem instances SS1-4 and SS5-1

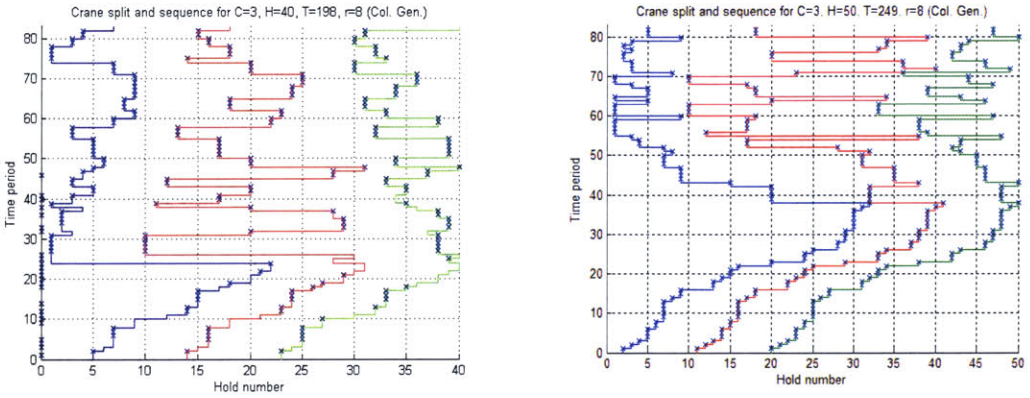


Figure 2-4. Solutions from the branch-and-price approach applied to problem instances SS4-2 and SS5-1

Second data set

The abovementioned methods are applied to realistic problems in the second data set. Their output makespan and computational performance are shown in Table 2-4.

Table 2-4. Output and computational performance of the various methods on second data set

Problem No.	Exact makespan	Heuristic makespan	Br-n-Pr. makespan	Exact runtime (s)	Heuristic runtime (s)	Br-n-Pr. runtime (s) / No. BnB iterations
SSP1	260	260	260	808.09	2.877	0.967 / 20
SSP2	◊	317	317	◊	6.997	1.039 / 12
SSP3	◊	412	412	◊	18.42	5.693 / 6

◊ Unknown – takes too long to execute model

Figures 2-5 and 2-6 shows the graphical solutions. The exact method gave 'out of memory' error for SSP2 and SSP3. The heuristic approach provided optimal makespan solutions. Branch-and-price algorithm performance was again shown to be the best

among the 3 approaches. For SSP3, the problem instance with the largest size, computation time did not exceed beyond 20 seconds. We can conclude that the heuristic and branch-and-price approaches are effective in producing high-quality solutions on realistic data very efficiently.

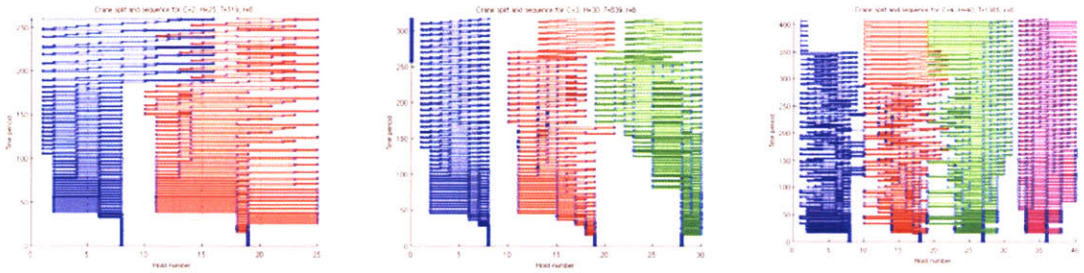


Figure 2-5. Solutions from the heuristic approach applied to problem instances SSP1, SSP2, SSP3

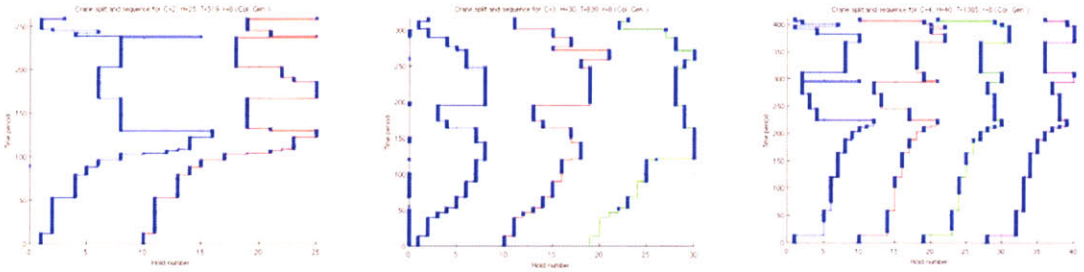


Figure 2-6. Solutions from the branch-and-price approach applied to problem instances SSP1, SSP2, SSP3

Effect of input parameters on runtime

The objective this sub-section is to analyze how values of H, C and T affect the computational time of the abovementioned methods. Figures 2-7, 2-8 and 2-9 respectively show the effect of increasing H, T and C on computational time, while keeping other parameters constant.

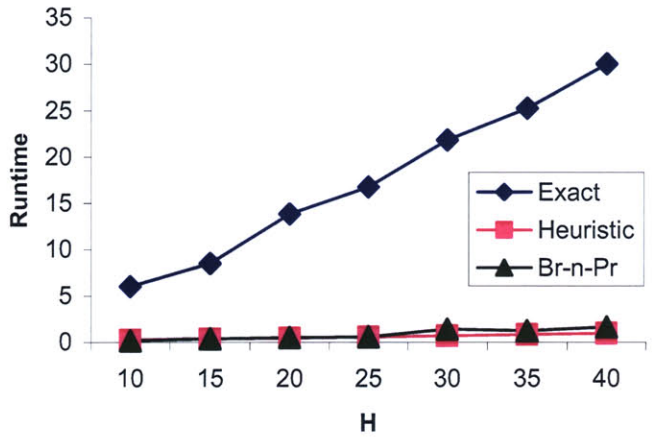


Figure 2-7. Computational time for various values of H, fix $T = 125$, $C = 2$, $r = 2$

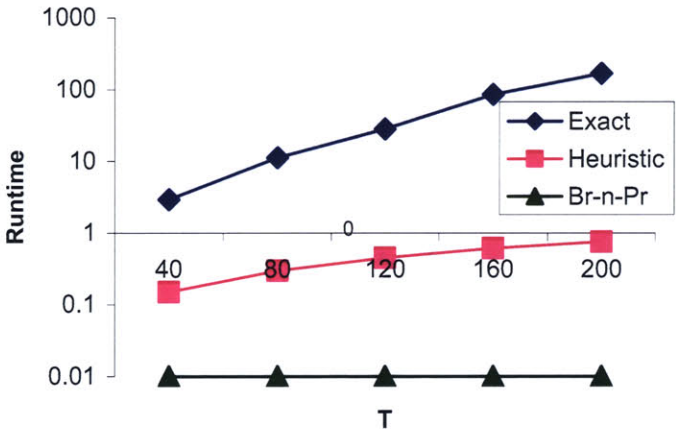


Figure 2-8. Computational time for various values of T, fix $H = 20$, $C = 2$, $r = 2$

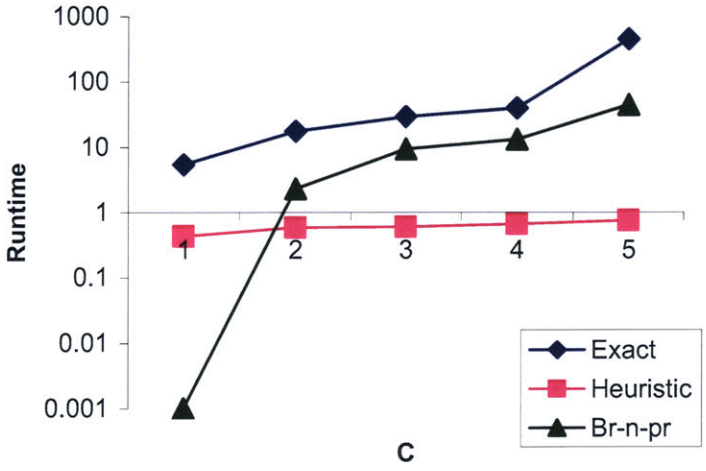


Figure 2-9. Computational time for various values of C, fix $T = 125$, $H = 25$, $r = 2$

For the exact model, there is an approximate linear increase in runtime with a corresponding increase in H, and an exponential increase in runtime with an increase in C and T. Figure 2-9 shows a jump in runtime when $C = 5$ because of the difficulty of 'squeezing' 5 QCs into a space of 25 bays with a clearance of 2; it becomes harder to find feasible integer solutions. This shows that computation time is affected by the difficulty of the problem and not just problem size alone. The branch-and-price runtime shows no definite trend. When $C = 1$, only 1 branch-and-bound iteration and required and runtime is negligible. In Figure 2-8, we observe that for certain combinations of H and T, computation time is almost zero even when C is greater than 1. The heuristic approach shows a relative linear relationship between runtime and H, C and T.

Chapter 3

Multiple-ship Model

3.1 Exact Mathematical Formulation

In this chapter, we develop a mathematical formulation which models ships berthing at specific, pre-planned times throughout the master planning horizon for loading and discharging operations. The objective remains to minimize the makespan of each vessel, while ensuring that the necessary clearance constraints for each vessel's QCs and a new set of yard congestion constraints are adhered to.

3.1.1 Problem Characteristics and Modeling Requirements

The same modeling assumptions and scheduling constraints made in the single-ship model, described in §2.1.1, are also applied to QC bay assignments for each vessel in the multiple-ship model. Hence, the constraint structure of the multiple-ship model is largely similar to that of the single-ship model.

In the multiple-ship model, QC scheduling of each vessel would be independent of other vessels if not for the additional yard congestion constraints imposed. The constraints prevent the number of QCs (from any vessel) handling containers that are

slated for a particular yard storage location at any time from exceeding a given quantity, known as the *yard activity threshold.* The aim is to keep to a minimum the level of yard crane and truck activity in the container storage blocks and hence prevent congestion and other operational inefficiencies. Thus, QC activity in each vessel interacts with activity in other vessels in the yard; if the yard activity thresholds are breached, QC jobs in the various vessels have to be re-scheduled.

Since berth planning has been completed in the pre-planning stages, we are given as an additional input parameter the *berth time* of each vessel, or the time at which QC operations for each vessel may commence. Instead of knowing only the number of containers to be discharged in each bay, we are now also provided with the yard storage destination for each container. Other new input parameters are the yard activity threshold for each storage location and the total number of ships berthing in the master planning horizon.

For simplicity, it is assumed that the time expended in moving containers from any storage block to any vessel at the quayside is the same. Realistically, the amount of time needed for transportation is affected by the distance between the vessel and storage locations, and also if there is congestion in the port road network. A second simplifying assumption is made such that only discharging jobs are conducted and storage locations will always be accessed after the QC move, as opposed to loading export containers which have a reverse operational flow.

3.1.2 Notation

The following notations are used for the multiple-ship mathematical formulation.

Indices:

i	Ship number, in no particular berthing order
j	Bay number, in increasing order of their relative location on the vessel (i.e. left to right)
k	Crane number, in increasing order of their relative position (i.e. left to right)
z	Yard storage location number
t	Time period index, denoting the interval of time from t-1 to t

Parameters:

S Number of ships berthing during the master planning horizon

C_i Number of QCs allocated to ship i

H_i Number of bays in ship i

L Number of container storage locations in the yard

f_{ijz} Number of containers to be discharged from bay j of ship i headed for storage location z

d_i Berth time of ship i; the vessel cannot be handled before this time

T_i Number of time periods in the individual planning horizon of ship i; set at $\sum_{j=1}^{H}\sum_{z=1}^{L} f_{ijz}$, or the makespan of the vessel if only one QC was allocated

T_{max} Total number of time periods in the master planning horizon, i.e. $\max_i\{d_i + T_i - 1\}$

r QC clearance value for all vessels, in terms of number of bays

w_z Yard activity threshold, maximum number of QCs allowed to work on containers headed for storage location z at any time

Decision variables:

$x_{ijk}(t)$ 1, if QC k is positioned at bay j of ship i at time period t; 0, otherwise

$\delta_{ijkz}(t)$ 1, if QC k is discharging a container headed for storage location z, at bay j of ship i at time period t; 0, otherwise

$\gamma_i(t)$ Completion flag: 0, if all container jobs in ship i are completed after time period t; 1, otherwise

3.1.3 The Model

The multiple-ship model for the crane split and sequencing problem is described as follow:

(*MSP*):

$$\text{Minimize}_{x,\delta,\gamma} -\sum_{i=1}^{S}\sum_{t=d_i}^{d_i+T_i-1}\gamma_i(t) \tag{3.1}$$

Subject to

$$\sum_{j=1}^{H_i} x_{ijk}(t) = 1, \qquad i=1..S, k=1..C_i, t=d_i...d_i+T_i-1 \tag{3.2}$$

$$\sum_{k=1}^{C_i} x_{ijk}(t) \le 1, \qquad i=1..S, j=1..H_i, t=d_i...d_i+T_i-1 \tag{3.3}$$

$$C_i\left(1-x_{ijk}(t)\right) \ge \sum_{l=\max\{1,j-r\}}^{j-1}\sum_{m=1}^{C_l} x_{ilm}(t), \quad i=1..S, j=2..H_i, k=1..C_i, t=d_i...d_i+T_i-1 \tag{3.4a}$$

$$C_i\left(1-x_{ijk}(t)\right) \ge \sum_{l=j+1}^{\min\{j+r,H_i\}}\sum_{m=1}^{C_l} x_{ilm}(t), \quad i=1..S, j=1..H_i-1, k=1..C_i, t=d_i...d_i+T_i-1 \tag{3.4b}$$

$$x_{ijk}(t) \le \sum_{l=j+1}^{H} x_{il,k+1}(t), \quad i=1..S, j=1..H_i-1, k=1..C_i-1, t=d_i...d_i+T_i-1 \tag{3.5a}$$

$$x_{ijk}(t) \le \sum_{l=1}^{j-1} x_{il,k-1}(t), \quad i=1..S, j=2..H_i, k=2..C_i, t=d_i...d_i+T_i-1 \tag{3.5b}$$

$$x_{ijk}(t) = 0, \qquad i=1..S, \{\forall k \mid k \ne 1\}, j=1..k-1, t=d_i...d_i+T_i-1 \tag{3.5c}$$

$$x_{ijk}(t) = 0, \qquad i=1..S, \{\forall k \mid k \ne C_i\}, j=H_i-(C_i-k)+1..H_i, t=d_i...d_i+T_i-1 \tag{3.5d}$$

$$\sum_{k=1}^{C_i}\sum_{t=d_i}^{d_i+T_i-1}\delta_{ijkz}(t) = f_{ijz}, \qquad i=1..S, j=1..H_i, z=1..L \tag{3.6}$$

$$\delta_{ijkz}(t) \le x_{ijk}(t), \qquad i=1..S, j=1..H_i, k=1..C_i, z=1..L, t=d_i..d_i+T_i-1 \tag{3.7}$$

$$\sum_{z=1}^{L}\delta_{ijkz}(t) \le 1, \qquad i=1..S, j=1..H_i, k=1..C_i, t=d_i..d_i+T_i-1 \tag{3.8}$$

$$\sum_{s}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\delta_{sjkz}(t) \le w_z, \qquad s \in \{\text{Ships berthed at } t\}, z=1..L, t=1..T_{\max} \tag{3.9}$$

$$\gamma_i(t) \le \frac{\sum_{k=1}^{C_i}\sum_{l=d_i}^{t}\delta_{ijkz}(l)}{f_{ijz}}, \qquad i=1..S, j=1..H_i, z=1..L, t=d_i...d_i+T_i-1 \tag{3.10}$$

$$\mathbf{x},\boldsymbol{\delta},\boldsymbol{\gamma} \in \{0,1\} \tag{3.11}$$

The individual planning horizon of a vessel i ranges from d_i to $d_i + T_i - 1$, and for each vessel the QC position and work constraints are applied in every time period within this range. All of the QC position constraints, labeled (3.2)-(3.5), are identical to the single-ship model's position constraints described in §2.1.3, and they are applied to each vessel, indexed by i, in the multiple-ship model. QC work constraints are almost similar except

for the additional index for storage location. The added work constraint is (3.8), which complements (3.11) in ensuring that a QC cannot discharge more than one container from a bay at any time. (3.6) is the work completion constraints; here, we have to ensure containers headed for all storage locations in each bay are handled. A similar observation is made in (3.10) for the definition of the work completion flag; containers in all storage locations in each bay must be handled for it to go to '1'.

Constraint (3.9) sums up the total amount of QC work done on containers headed for a particular storage location, z, for all 'active' vessels at a particular time period, t, and ensures that it does not exceed the yard activity threshold for that location, w_z. The value of w_z may be dissimilar among the different storage locations depending on, for example, the type of yard crane used or the size of the traffic lanes. Each of the yard congestion constraints restricts the yard activity for location z at time t, and is given a coordinate, (z,t), for easy reference.

The objective function evaluates the sum of the makespan for each vessel; the makespan for vessel i is $T_i - \sum_{t=d_i}^{d_i+T_i+1} \gamma_i^*(t)+1$. We consider that each vessel is given equal priority for work completion, however by affixing differentiated cost co-efficient values for each vessel, i.e. $-\sum_{i=1}^{S}\sum_{t=d_i}^{d_i+T_i-1} c_i\gamma_i(t)$, we can easily penalize waiting times for higher-priority ships.

The multiple-ship model is programmed with OPL*Script* and computational experiments performed with CPLEX show that the computational time is excessive even for smaller-sized problems. This leads us to develop an approach which utilizes the efficient solution strategies for minimizing the makespan of a single vessel, presented in chapter 2, to speed up the solution of the multiple-ship problem. The Lagrangian relaxation technique is one such method and it is described in §3.2.

3.2 Lagrangian Relaxation Framework

Lagrangian Relaxation (LR) is a powerful tool for the approximate solving of large-scale IP problems. It was first introduced by Held and Karp [22] and applied to the traveling salesman problem. The main motivation for using LR is that it provides stronger lower bounds compared to LP relaxation for problems that do not possess the Integrality Property. Solutions to LR problems, or *Lagrangian solutions*, can also be used as good starting points for a heuristic search of a primal feasible, upper bound solution. These bounds provide us with brackets around the optimal integer value which are usually tighter than brackets coming from a standard LP-based branch-and-bound algorithm. Geoffrion [23] and Fisher [24] provide a thorough overview of the technique and show proof of the superior bounds of generated by Lagrangian solutions.

In the LR approach, a set of 'complicating' constraints are relaxed and dualized by adding them to the objective function with penalty coefficients, known as *Lagrangian multipliers*. The LR problem must be much simpler to solve compared to the original problem for it to yield any computational advantage. Guignard [25] lists out the many ways in which a given problem can be relaxed in a Lagrangian fashion. In particular, we can select a set of 'coupling' constraints to relax so that the LR problem decomposes into multiple sub-problems. One clear advantage is the reduction in computational complexity for the Lagrangian sub-problems; for instance, it is generally easier to solve 50 problems with 100 binary variables each than a single problem with 5000 (50 × 100) binary variables.

In our case, a logical set of constraints to relax are the yard congestion constraints (3.9). They link the decision variables associated with different vessels together and if relaxed, the problem would decompose by vessel. The corresponding objective function of the relaxed problem is defined as follow:

$$L(\mathbf{x},\boldsymbol{\delta},\boldsymbol{\gamma},\boldsymbol{\lambda}) = -\sum_{i=1}^{S}\sum_{t=d_i}^{d_i+T_i-1}\gamma_i(t) + \sum_{z=1}^{L}\sum_{t=1}^{T_{\max}}\lambda_{zt}\left(\sum_{s}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\delta_{sjkz}(t) - w_z\right) \tag{3.12}$$

A Lagrangian multiplier, λ_{zt}, is attached to each yard congestion constraint with coordinate (z,t). We can further define the minimization objective function with relaxation for a given values of λ_{zt}:

$$L^*(\lambda) = \min_{x,\delta,\gamma} -\sum_{i=1}^{S}\sum_{t=d_i}^{d_i+T_i-1}\gamma_i(t) + \sum_{z=1}^{L}\sum_{t=1}^{T_{\max}}\lambda_{zt}\left(\sum_{s}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\delta_{sjkz}(t) - w_z\right) \tag{3.13}$$

The primal problem is the original multiple-ship model described in §3.1.3, or *MSP*. The LR approach involves solving the (3.13), for given values of λ_{zt}'s, over the primal problem polytope excluding the 'complicating' yard congestion constraints; this problem is labeled as *MSLR* (described in §3.2.1). For any non-negative values of λ_{zt}'s, the following bounds are valid, where δ^* and γ^* are the optimal primal solutions:

$$L^*(\lambda) \le -\sum_{i=1}^{S}\sum_{t=d_i}^{d_i+T_i-1}\gamma_i^*(t) + \sum_{z=1}^{L}\sum_{t=1}^{T_{\max}}\lambda_{zt}\left(\sum_{s}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\delta_{sjkz}^*(t) - w_z\right) \le -\sum_{i=1}^{S}\sum_{t=d_i}^{d_i+T_i-1}\gamma_i^*(t) \tag{3.14}$$

Since the value of $\sum_{s}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\delta_{sjkz}^*(t) - w_z$ is negative by definition, the optimal cost of *MSLR*, for any given value of λ_{zt}, is always less than or equal to the optimal primal cost. The ultimate goal is to find optimal dual variables, λ_{zt}^*, such that complementary slackness conditions occur; the perturbation to the original cost objective, $\sum_{z=1}^{L}\sum_{t=1}^{T_{\max}}\lambda_{zt}^*\left(\sum_{s}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\delta_{sjkz}^*(t) - w_z\right)$, is equal to 0. The following describes the problem which produces the 'tightest' lower bounds for *MSLR* possible:

$$(MSLD): \qquad L^*(\lambda^*) = \text{Maximize}_{\lambda \ge 0} L^*(\lambda) \tag{3.15}$$

MSLD is known as the Lagrangian dual of *MSP* relative to the yard congestion constraints. The difference between *MSLD* and *MSLR* is that *MSLD* is a problem in the dual space of λ_{zt}'s, while *MSLR* is a problem in the primal variables x, δ and γ. A

systemic approach to searching the dual space for λ_{zt}^{*}'s that maximize $L^{*}(\lambda)$ is the sub-gradient method, which is further described in §3.2.3.

In the process of evaluating λ_{zt}^{*} in *MSLD*, Lagrangian solutions are also generated. In the sub-gradient procedure, *MSLR* is repetitively solved and Lagrangian solutions produced for different values of λ_{zt}. The primal-dual optimal saddle-point which invokes complementary slackness is not always found in integer programming. In general, the optimal cost of the dual problem, *MSLD*, is not equal to the optimal cost of the primal problem, *MSP*. This means that the latter is unknown to us; an upper bracket has to be found.

It is usually rare in practice that Lagrangian solutions are primal feasible in the original problem, *MSP*. However, it is not uncommon that Lagrangian solutions are only mildly affected by infeasibility and can be made feasible with minor adjustments. In our case, the Lagrangian solutions, obtained in every iteration of the sub-gradient procedure, may violate yard congestion constraints and a primal heuristic is developed (described in §3.2.4) to remove these violations. The resulting feasible solutions then become upper bounds to the optimal primal solution. The eventual proximity of the primal (or the upper bound to it) and dual optimal objective values would indicate that the polytope formed by the non-relaxed constraints approximates the convex hull of the same region ([26] provides a rigorous proof) and that we have been provided with solutions of high quality.

The main objective of the LR framework is to reduce the difference between the upper bounds and lower bounds to the optimal cost of *MSP* to a satisfactorily small value. The stopping criterion of the sub-gradient routine is discussed in §3.2.5. The following describes the overall flow of the LR framework:

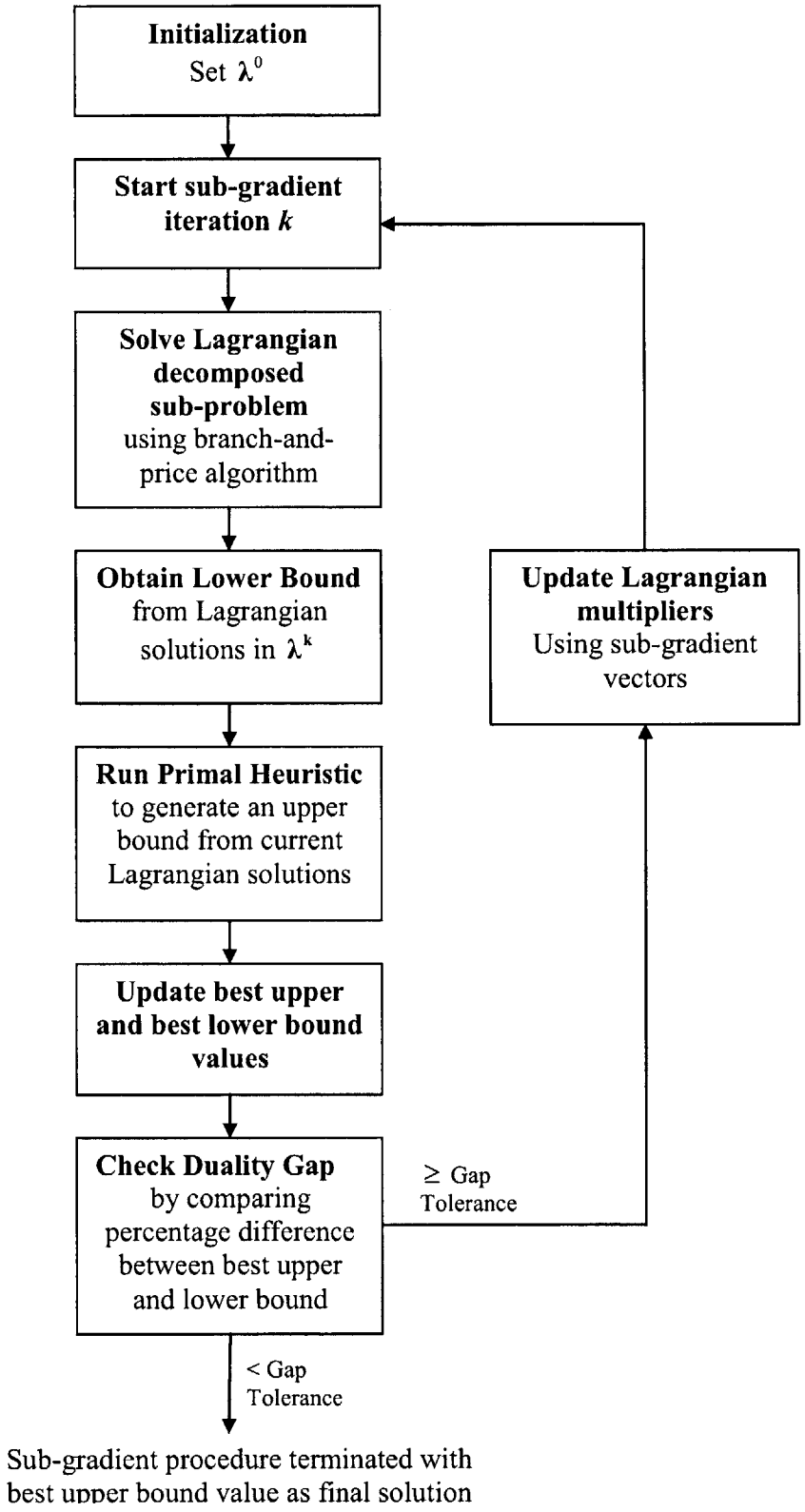


Figure 3-1. Overall procedure of the Lagrangian Relaxation framework

3.2.1 Decomposition of the Lagrangian Relaxation Form into Sub-problems

The objective function of *MSLR*, shown in (3.12), can be re-written as follows:

$$
\begin{aligned}
&L(\mathbf{x},\boldsymbol{\delta},\boldsymbol{\gamma},\boldsymbol{\lambda}) \\
&= -\sum_{i=1}^{S}\sum_{t=d_i}^{d_i+T_i-1}\gamma_i(t) + \sum_{z=1}^{L}\sum_{t=1}^{T_{\max}}\lambda_{zt}\left(\sum_{s}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\delta_{sjkz}(t) - w_z\right) \\
&= -\sum_{i=1}^{S}\sum_{t=d_i}^{d_i+T_i-1}\gamma_i(t) + \sum_{s}\sum_{t=1}^{T_{\max}}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\sum_{z=1}^{L}\lambda_{zt}\delta_{sjkz}(t) - \sum_{z=1}^{L}\sum_{t=1}^{T_{\max}}\lambda_{zt}w_z \\
&= \sum_{i=1}^{S}\sum_{t=d_i}^{d_i+T_i-1}\left(\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\sum_{z=1}^{L}\lambda_{zt}\delta_{ijkz}(t) - \gamma_i(t)\right) - \sum_{z=1}^{L}\sum_{t=1}^{T_{\max}}\lambda_{zt}w_z \\
&= -\sum_{z=1}^{L}\sum_{t=1}^{T_{\max}}\lambda_{zt}w_z + \sum_{i=1}^{S}\underbrace{\left\{\sum_{t=d_i}^{d_i+T_i-1}\left(\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\sum_{z=1}^{L}\lambda_{zt}\delta_{ijkz}(t) - \gamma_i(t)\right)\right\}}_{L^i(\boldsymbol{\lambda})}
\end{aligned}
\tag{3.16}
$$

Without the 'coupling' yard congestion constraints, the objective function and all the non-relaxed constraints of *MSP* can be separated into S independent minimization sub-problems, one for each vessel and with objective function $L^i(\boldsymbol{\lambda})$. To solve $L^*(\boldsymbol{\lambda})$ given values of λ_{zt} 's, the Lagrangian sub-problems for each vessel, $MSLR^i$, are solved individually and the results summed together; finally a constant term (1st term in (3.16)) is added to the summation. The Lagrangian sub-problem for each vessel is defined as follows:

($MSLR^i$):

$$\text{Minimize}_{x,\delta,\gamma} L^i(\lambda) \tag{3.17}$$

Subject to

Constraints (2.2), (2.3), (2.4a), (2.4b), (2.5a), (2.5b), (2.6a), (2.6b), (2.10) and

$$\sum_{t=d_i}^{d_i+T_i-1} \delta_{jz}(t) = f_{ijz}, \qquad j=1..H_i, z=1..L \tag{3.18}$$

$$\delta_{jkz}(t) \le x_{jk}(t), \qquad j=1..H_i, k=1..C_i, z=1..L, t=d_i..d_i+T_i-1 \tag{3.19}$$

$$\sum_{z=1}^{L} \delta_{jkz}(t) \le 1, \qquad j=1..H_i, k=1..C_i, t=d_i..d_i+T_i-1 \tag{3.20}$$

$$\gamma(t) \le \frac{\sum_{k=1}^{C}\sum_{l=d}^{t} \delta_{jkz}(l)}{f_{jz}}, \qquad j=1..H_i, z=1..L, t=d_i...d_i+T_i-1 \tag{3.21}$$

Constraints appearing in $MSLR^i$ are almost identical to the single-ship model (described in §2.1.3), except that the variables δ's and the input parameter f's are additionally indexed by z for yard storage locations. The input parameters for each Lagrangian sub-problem are its associated vessel's load profile, berth time and vessel size. The objective function (3.17) differs from the single-ship objective by a single penalty term, $\sum_{t=d_i}^{d_i+T_i-1}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\sum_{z=1}^{L}\lambda_{zt}\delta_{ijkz}(t)$, which defines the summation of a factor of the total QC usage in each (z,t). This extra term causes an inflation in cost from the single-ship objective.

The basic idea of the $MSLR^i$ is to re-schedule the QC jobs to (z,t) coordinates with small λ_{zt} values without compromising the vessel makespan. Although the $MSLR^i$ is easier to solve compared to *MSP*, it is still very difficult for large-scale problems. We can infer the best-case computational performance of $MSLR^i$ by a comparison with the single-ship model's performance found in Table 2-4; the latter, without the complicating penalty term in the objective, cannot be efficiently solved by CPLEX for realistic-sized problems. However, due to the relative similarity between $MSLR^i$ and the single-ship model, we are able to take advantage of the efficient branch-and-price solution strategy developed for the single-ship model; the branch-and-price method will be adjusted to take into account

the penalty term in the objective. The following sub-sections detail the application of this approach for solving $MSLR^i$.

3.2.2 Solving Lagrangian Sub-problems with Branch-and-price

The general framework for the branch-and-price algorithm applied to the Lagrangian sub-problems is similar to that used to solve the single-ship model to optimality. However, because of the additional index, z, for yard storage locations, there will be L times the number of rows in each column. The presence of a time-indexed term, λ_{zt}, in the cost function for each column, arising from the left term in $L^i(\lambda)$ (3.16), means that the cost scheme differs according to time period and a separate pricing problem has to be solved for each period.

The expected increase in the problem size of the RMP and number of RMP/PP iterations needed for LP optimality suggest that certain problem-specific measures (i.e. column management), not used in solving the single-ship model, have to be developed to reduce the computational complexity of the branch-and-price algorithm. The overall flow of the branch-and-price algorithm, within a single sub-gradient iterate, is described in the following chart:

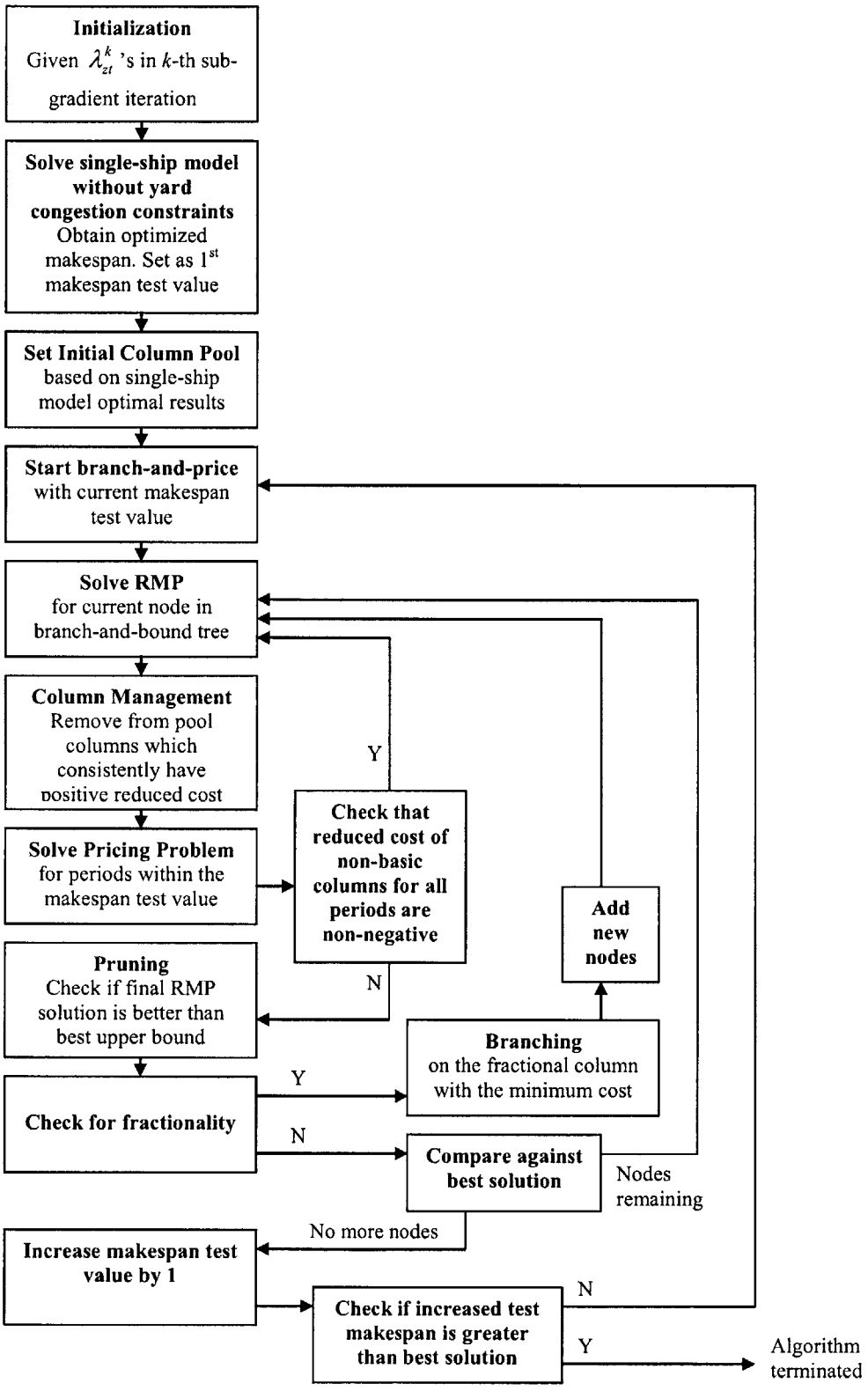


Figure 3-2. Overall flow of the branch-and-price algorithm for solving Lagrangian sub-problems

3.2.2.1 Re-formulation of the Lagrangian Sub-problem into Column Generation Form

The goal of re-formulating the original compact formulation of $MSLR^i$ is to tighten the LP relaxation so that less branching is required to obtain integer solutions during branch-and-bound. In the column generation form of the single-ship model, each column represents a particular QC position-assignment, as described in §2.3.2. Here, because the cost of QC activity in each period is affected by δ's, each column instead represents a particular QC work-assignment designated for a period.

The column generation form of $MSLR^i$ is called the *master problem*. Each column in the master problem consists of $H \times L$ rows, one for each (j,z) coordinate. For example, an 'active' (j,z) row of a column designated to period t means that $\delta_{jz}(t) = 1$. Because the work completion constraints (3.18) have to be satisfied, the master problem has a '0-1' set-partitioning structure with the right-hand vector, f, being the load profile for each (j,z) coordinate. Identical columns may differ in cost if designated to different periods because of the effect of λ_{zt}'s. Figure 3-3 shows a pattern, r, which depicts work-assignments for period 2 specifically:

$$y_{r,2} = \begin{bmatrix} 0 \\ 1 \\ \hline 1 \\ 0 \\ \hline 1 \\ 0 \end{bmatrix} \qquad f = \begin{bmatrix} 3 \\ 2 \\ \hline 5 \\ 1 \\ \hline 3 \\ 0 \end{bmatrix}$$

$$\delta_{1,1,2}(2) = 1, \delta_{2,2,1}(2) = 1, \delta_{3,3,1}(2) = 1, \text{ All other } \delta \text{'s are '0'.}$$

$$f_{1,1} = 3, f_{1,2} = 2, f_{2,1} = 5, f_{2,2} = 1, f_{3,1} = 3, f_{3,2} = 0$$

Figure 3-3. Example of a column for t=2 and right-hand vector of the master problem in set-partitioning form for H=3, L=2, C=3, r=0

The master problem includes an exponential number of columns, designated to different periods, each representing a feasible 'pattern':

($MSMP^i$):

$$\text{Minimize}_y \sum_{t=d_i}^{d_i+T_i-1} \sum_{r\in R_t} \left(1+\sum_{z\in R_t} \lambda_{zt}\right) y_{r,t} \tag{3.22}$$

Subject to

$$\sum_{t=d_i}^{d_i+T_i-1} \sum_{r\in R_t:(j,z)\in R_t} y_{r,t} = f_{jz}, \quad \forall j,z \tag{3.23}$$

$$\sum_{r\in R_t} y_{r,t} \le 1, \qquad t = d_i..d_i+T_i-1 \tag{3.24}$$

$$\sum_{r\in R_t} y_{r,t+1} - \sum_{r\in R_t} y_{r,t} \le 0, \qquad t = d_i..d_i+T_i-2 \tag{3.25}$$

The following is a description of notations in $MSMP^i$:

R_t Set of all feasible QC work-assignment 'patterns' designated to period t (huge set)
$y_{r,t}$ Number of times QC work-assignment 'pattern' r, designated to period t, is used
f_{jz} Number of containers in ship bay j headed for storage location z

The columns in $MSMP^i$ can be designated to any time period within the planning horizon for the vessel. The integrality of column-variables is relaxed in $MSMP^i$ and it is solved as an LP. An explicit search of all possible feasible 'patterns' in $MSMP^i$ is computationally impossible since $|R_t|$ is huge. The LP relaxation of the problem with a reasonably smaller subset of columns is solved in the RMP, as described in §3.2.2.3.

The objective function (3.22) sums up the costs associated with 'active' columns. The cost of each column is split into 2 parts, contribution to the makespan ('1' for all columns) and the *penalty term* associated with QC activity in the designated period (as discussed in §3.2.1). Since δ's are either '1' or '0', for each column designated to t, we sum up only λ_{zt} values which correspond the z-coordinate of an active row . If values for λ_{zt}'s are large, there may be a trade-off between minimizing the cost of the 2 competing components. Take for example a vessel with an optimized makespan of 16. We have values of λ_{zt}'s for $t \le 16$ significantly larger than values of λ_{zt}'s for $t > 16$. When considering both cost components, it may be worthwhile to let the makespan

deteriorate by a small amount so that the smaller λ_{zt}'s for $t > 16$ are added to the overall cost instead of the larger λ_{zt}'s for $t \leq 16$. This is elaborated further in the discussion of solution bounds for $MSMP^i$ in §3.2.2.2.

(3.23) show the work-completion constraints in set-partitioning form. Constraint (3.24) states that for each period in the planning horizon, only one column designated to it from R_t can be included in the solution. All columns designated to periods later than the optimal makespan are set to 0. Constraint (3.25) imposes that columns designated to earlier periods must be in the solution if later periods are, ensuring the vessel makespan is evaluated correctly. Without (3.25), columns will simply be picked from periods which have the smallest λ_{zt}'s, and these periods may be non-consecutive. An example is given for a vessel that requires 5 periods to complete QC work and has a planning horizon of 12 periods. If λ_{zt}'s are smallest (i.e. '0') during $8 \leq t \leq 12$, columns designated to the last 5 periods will be selected with a cost of 5. This masks the fact that the vessel has actually been in port for 12 periods in total, although QCs are only active for the last 5.

The branch-and-price approach is especially useful when the resulting master problem has a structure, such as set-partitioning, that is well known to have a tight LP objective function value and whose polyhedral structure has been well-studied [27]. Constraints (3.25) can cause many fractional solutions to appear. Strengthening of the $MSMP^i$ through disaggregation of (3.25) causes 'messiness' in the algorithm when passing dual variables into the pricing problem. It is worthwhile to examine if an alternative approach, which utilizes a formulation that excludes (3.25), can be developed, and this is discussed in §3.2.2.3.

3.2.2.2 Bounds on the Optimal Solution of the Lagrangian Sub-problem

Before attempting to solve $MSMP^i$ directly, it may be pertinent to examine if we can obtain an upper and lower bound bracket around the optimal integer solution cost. The knowledge of a range of values around the optimal cost will allow us to take certain

computational 'short-cuts' in the branch-and-price procedure because certain column combinations can immediately be disregarded.

A lower bound

The main component that contributes to overall costs is the makespan value. Ignoring the penalty term cost component, $\sum_{z \in R_t} \lambda_{zt}$, for a moment, the problem of minimizing the makespan, while adhering to clearance, work-completion and other constraints, is almost similar the single-ship model described in §2.1.3, with the exception of additional z indices for the δ variables. In chapter 2, we presented two quick solution strategies for the single-ship model. We treat the process of solving the modified single-ship model as a 'black box', and refer to it as the *single-ship oracle*. Since λ_{zt}'s are non-negative by definition, the optimal makespan of the single-ship oracle becomes a lower bound to the actual integer solution cost of $MSMP^i$.

In the example shown in Figure 3-4, if a vessel has an optimized makespan of 6, since the term $\sum_{z \in R_t} \lambda_{zt}$ is strictly non-negative, the actual cost cannot be less than 6 without violating work-completion constraints (3.23). Note that for each time period, only a single column designated to it is selected (3.24), and that 'active' periods are consecutive starting from the berth time (3.25).

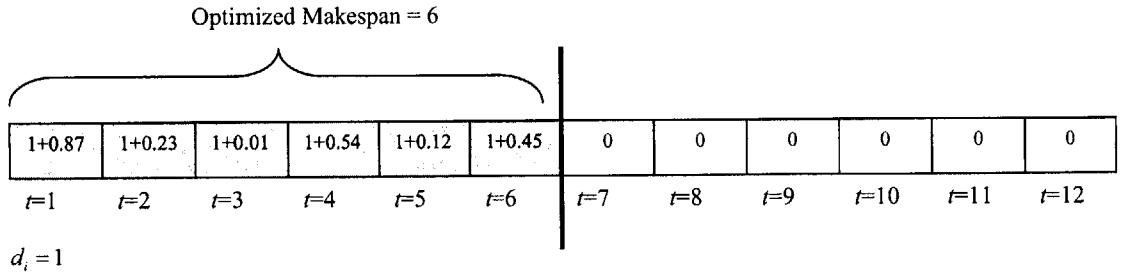


Figure 3-4. Illustration of the breakdown of costs by column in an optimal integer solution of $MSMP^i$

Generating an initial feasible column pool

To start off the RMP, it is necessary to initialize it with a small set of columns which provide a feasible basis. The outputs of the single-ship oracle are designated as δ^{SSO}, which can be mapped into '0-1' columns. The initial column pool will have the same number of columns as the optimal makespan derived from the single-ship oracle; columns designated to periods later than the makespan are excluded.

The pseudo-code of the procedure for generating the initial column pool is as follows:

---

set *makespanLB* = Optimized makespan from single-ship oracle
set *numCols* = -1
initialize *config*[*numCols*,1..$H\times L$]
for $t = d_i .. d_i$ + *makespanLB* - 1
    numCols = *numCols* + 1
    Increase size of *config* by 1 column
    Re-set rows of new column to '0'

    forall (*j*,*z*) **do**
        config[*numCols*,(*j*,*z*) row number] = $\delta_{jz}(t)$
    end for

    config[*numCols*, $H\times L+1$] = t
end for

---

config stores all the feasible columns in the column pool. *makespanLB* holds the optimized makespan value from the single-ship oracle. The $(H\times L+1)$-th row in each column stores the designated period number. The following details the mapping between the j and z index for δ^{SSO}'s, and the row number of columns in *config*:

$$j = \left\lfloor \frac{l-1}{L} \right\rfloor + 1 \tag{3.26}$$

$$z = \left((l-1) \bmod L\right) + 1 \tag{3.27}$$

Generating an upper bound

An available upper bound to the optimal integer cost of $MSMP^i$ will allow us to discard solutions with makespan values that are greater than its rounded-up value. The

problem size is reduced because columns designated to periods above the rounded-up upper bound value can be ignored; in addition these columns need not be priced out (§3.2.2.3) when solving the LP. For example, if a given upper bound is 17.9, the makespan of the vessel cannot exceed 18 because the minimum cost of the columns designated to periods 1 to 18 is '1' for each.

An initial upper bound value can be obtained from δ^{SSO} 's. The penalty cost component, $\sum_{z \in R_t} \lambda_{zt}$, of each column in the initial column pool can be added to the lower bound value obtained earlier; the total value added is $\sum_{t=d_i}^{d_i+T_i-1} \sum_{j=1}^{H_i} \sum_{k=1}^{C_i} \sum_{z=1}^{L} \lambda_{zt} \delta_{ijkz}^{SSO}(t)$ (refer to (3.16)).

In Figure 3-4, the brackets around the optimal cost of $MSMP^i$ in the example presented is $6 \leq \sum_{t=d_i}^{d_i+T_i-1} \sum_{r \in R_t} \left(1 + \sum_{z \in R_t} \lambda_{zt} \right) y_{r,t}^{*} \leq 8.22$. Possible values for vessel makespan in the optimal solution are 6, 7 or 8. In §3.2.2.1, we mentioned that increasing the vessel makespan may allow a lowering of overall costs. This leads us a formulation of the RMP such that each possible value of makespan is tested one at a time; this is discussed in §3.2.2.3.

3.2.2.3 Restricted Master Problem and Pricing Problem

Restricted Master Problem

The RMP is commonly identical in structure to the master problem except that only a small subset of all possible columns/variables is accounted for. In our master problem, $MSMP^i$, we previously discussed problems arising from constraint (3.25); it causes a high degree of fractionality in the LP. However, if the problem involves minimizing costs over a fixed makespan value (derived from solution bounds discussed in §3.2.2.2), (3.25) can be discarded and the problem more closely approximates a 'tight' set-partitioning form.

The modified RMP now specifies the exact number of time periods that are expended to meet work-load requirements. Intuitively, the problem difficulty decreases as we no longer have the dual objectives of minimizing the sum of both the makespan and penalty term costs; instead, only minimization of the latter is required. The modified RMP model reads as follows:

($MSRMP^i$):

$$\text{Minimize}_y \sum_{t=d_i}^{d_i+T_{MS}-1} \sum_{r\in R_t'} \sum_{z\in R_t'} \lambda_{zt} y_{r,t} \tag{3.28}$$

Subject to

$$\sum_{t=d_i}^{d_i+T_{MS}-1} \sum_{r\in R_t':(j,z)\in R_t'} y_{r,t} = f_{jz}, \quad \forall j,z \tag{3.29}$$

$$\sum_{r\in R_t'} y_{r,t} = 1, \qquad t = d_i..d_i+T_{MS}-1 \tag{3.30}$$

$$y_{r,t} = 1, \qquad \forall (r,t) \in \{P \mid \text{Branch-up constraints}\} \tag{3.31}$$

$$y_{r,t} = 0, \qquad \forall (r,t) \in \{P \mid \text{Branch-down constraints}\} \tag{3.32}$$

T_{MS} refers to the *makespan test value*, one of the possible makespan values arising from the solution bounds of $MSMP^i$. R_t' refers to a subset of columns of R_t for each period. The objective function (3.28) only includes penalty term costs since the makespan cost is known (fixed at T_{MS}). (3.30) now becomes an equality constraint because columns designated to all periods up to T_{MS} must be represented in the solution. P denotes the set of all branching rules in the constraint pool; (3.31) and (3.32) are branching constraints discussed in §3.2.2.4.

Pricing Problem

The pricing problem to $MSRMP^i$ has a similar function and structure to that for the single-ship model discussed in §2.3.1. Dual variables associated with new constraints (3.30), p, from the RMP are imported into the PP. In addition, work-assignment decision variables, δ's, are included, which are used to define a new feasible column. The PP is shown as follows:

($MSPP_t^i$):

$$\text{Minimize}_{x,\delta} \sum_{j=1}^{H_i} \sum_{z=1}^{L} \lambda_{zt} \delta_{jz} - \sum_{(j,z)\in\{\delta_{jz}=1\}} u_{jz} - p_t \tag{3.33}$$

Subject to constraint (2.28), (2.29a), (2.29b) and

$$\delta_{jz} \le x_j, \qquad \forall j, z \tag{3.34}$$

$$\sum_z \delta_{jz} \le 1, \qquad \forall j \tag{3.35}$$

$$\mathbf{x}, \boldsymbol{\delta} \in \{0,1\} \tag{3.36}$$

The objective function (3.33) searches for the minimum reduced cost of a non-basic column. (2.28) and (2.29a-b) enforce QC position constraints, while constraints (3.34) and (3.35) dictate where the QCs are allowed to work, or which combinations of rows may be 'active'. (3.36) defines $MSPP^i$ as an IP.

$MSPP_t^i$ has a different cost vector for pricing out columns designated to each t. The RMP and PP are solved repetitively for each time period in the range $d_i \le t \le d_i + T_{MS} - 1$. A full run of RMP and PP in this range is called an *RMP/PP cycle*. At least 1 RMP/PP cycle must be completed, where the PP produces non-negative objective values for all periods in the range, before LP optimality is satisfied. When solving the LP in the example presented in Figure 3-4, in each RMP/PP cycle, the RMP and PP are run repetitively to find good columns for each period in $1 \le t \le 6$. Taken that for the first 10 RMP/PP cycles, columns with negative reduced cost are found for all periods in the stated range. During the 11^{th} cycle, the PP adds columns to the RMP only at t=1 and t=2; when PP is run from t=3 to t=6, a non-negative objective is found. A 12^{th} RMP/PP cycle, where no new columns are added for any period in the range, must be run before the LP algorithm can terminate.

For non-root nodes in the branch-and-bound tree, branching has occurred on some column-variables. When solving the LP, the $MSPP_t^i$ need not be solved for t's where columns designated to them have been branched to '1', since each period can only be represented by at most one 'active' column (i.e. constraint (3.30)). Computational savings

are made especially when solving the LP for nodes deep in the branch-and-bound tree. Branching on column-variables is further discussed in §3.2.2.4.

Equivalence of Set-Covering and Set-Partitioning Forms

Barnhart [18] asserts that when there is a choice, a set-covering formulation is preferred over a set-partitioning formulation because its LP relaxation is numerically far more stable and thus easier to solve. In general, the set-covering and set-partitioning solutions are the same if changing any 1 in a column into a 0 gives a valid and no more expensive column; non-negative λ_{zt}'s and δ's in the objective of the Lagrangian sub-problem ensure this is the case.

Returning to the original multiple-ship model, *MSP*, in §3.1.3, assuming we replace the equality sign in work-completion constraints (3.6) with the greater-than-or-equal sign; in other words, allow more work to be done than necessary. The modified *MSP* is solved with the same optimal makespan, since doing more work with the same work-load can never reduce completion time. Assuming that constraints (3.6) are exceeded by K, we can arbitrarily reset K number of δ's that are '1' back to '0'. QC position constraints (3.1)-(3.5) are not violated by the resetting action because constraint (3.7) allows values of x's to be independent of δ's. Neither are the QC work constraints and yard congestion constraints violated since less work done makes them 'less violated' in general. We can conclude that since constraint (3.6) can be changed as abovementioned in *MSP* without affecting cost or feasibility, constraint (3.29) in $MSRMP^i$ can be converted to the set-covering form:

$$\sum_{t=d_j}^{d_j+T_{MS}-1} \sum_{r\in R_t^i:(j,z)\in R_t^i} y_{r,t} \geq f_{jz}, \quad \forall j,z \tag{3.37}$$

Column Management

After many RMP/PP cycles, the column pool will grow to a large size and cause a reduction in the speed of solving the $MSRMP^i$. Column management involves removing columns that are no longer active in the LP solution from the RMP column pool. Only

non-basic columns which are consistently inactive should be deleted, otherwise it may be re-generated by the PP in future iterations.

The column management algorithm is run immediately after the RMP. It scans the entire pool for columns with positive reduced cost and increments an inactivity counter for each column. Once the inactivity counter value exceeds a threshold, set at 5 planning-horizon lengths, the column is discarded from the pool.

The following pseudo-code describes the column management algorithm:

---

forall k = active columns in column pool
  if k-th column has not been fixed to '0' **then**
    if Reduced cost of $y_k > 10^{-5}$ **then**
      Increment *config*[k, $H \times L+2$] by 1
      if *config*[k, $H \times L+2$] $= 5T_i$ **then**
        config[k, $H \times L+2$] = -1
      end if
    end if
  end if
end for

For non-root nodes in the branch-and-bound tree, some column-variables may have been fixed permanently to '0'. However, we are not free to discard these previously branched columns as they are needed in the RMP (3.32) to fix column values. The column counter is stored in *config*[k, $H \times L+2$], a hidden row within the column.

Testing each possible makespan value

Bounds to the optimal solution of $MSMP^i$ provide us with several possible makespan values, the integer points within the bounds, as discussed in §3.2.2.2. The entire branch-and-price procedure must run several times, once for each makespan test value, T_{MS}, using $MSRMP^i$ and $MSPP_t^i$, to see which produces the smallest cost objective. This would be advantageous only if the combined computational time is smaller compared to running branch-and-price procedure once on $MSMP^i$; computational experiments performed show this is indeed the case.

An initial upper bound value is created from the output of the single-ship oracle. When branch-and-price is carried out on each T_{MS} value, improved upper bounds may be generated which narrow the bounds bracket and eliminate the need to test larger T_{MS} values. Using the example in Figure 3-4 where the initial bounds bracket was 6 and 8.22, the first T_{MS} value tested is 6 and we take that the solution produced by branch-and-price is 7.05. A narrower solution bounds of 6 and 7.05 is created and we can eliminate testing for $T_{MS} = 8$, saving some computational effort. If we get a solution of 7.8 for when $T_{MS} = 7$, then the final minimum cost is 7.05 with a makespan value of 6.

Computational time can be further shaved if we examine the LP solution produced at the root node of the search tree. We are aware that the LP-relaxed cost is a lower bound to the optimal IP cost. Therefore, if the LP objective from the root-node RMP (when testing for any T_{MS}), is larger than the best upper bound known for $MSMP^i$, we can immediately stop the searching for an integer solution and proceed to test for $T_{MS}+1$ or terminate testing.

The following describes the pseudo-code of the procedure used to test different makespan values within the solution bounds bracket:

---

set *makespanLB* = Optimal makespan from single-ship oracle

set $makespanUB = makespanLB + \sum_{t=d_i}^{d_i+T_i-1} \sum_{j=1}^{H_i} \sum_{k=1}^{C_j} \sum_{z=1}^{L} \lambda_{zl} \delta_{ijkz}^{SSO}(t)$

set *Tms* = *makespanLB*

repeat
  Run branch-and-price procedure with current *Tms* value
  if optimal LP objective of RMP in root node + *Tms* > *makespanUB* **then**
    break out of repeat loop
  end if

  BBbestUB = Optimal integer solution from branch-and-price

  if *Tms* + *BBbestUB* < *makespanUB* **then**
    makespanUB = *Tms* + *BBbestUB*

```
            Store x*, δ* and γ* values for this ship
        end if

        if Tms + 1 ≥ makespanUB then
            break out of repeat loop
        else
            Tms = Tms + 1
        end if
    end repeat
```

makespanLB and *makespanUB* store the lower bound and upper bound values to $MSMP^i$. They are initially set to values obtained from the output of the single-ship oracle. *BBbestUB* holds the value of the best integer solution from the search tree; it consists of the penalty term cost component only (refer to §3.2.2.3 for RMP formulation).

3.2.2.4 Branching and Pruning

Branching

In branch-and-bound, if the goal is to find a good feasible solution, it makes sense to choose a branching decision that divides the solution space in such a way that we are more likely to find a good solution in one of the two nodes created and then choose this node for evaluation first [18]. Standard variable branching is used as it provides a deep cut of the LP feasible space in the '1' branch. Together with a depth-first-search strategy of the tree, it ensures that integer solutions are found quickly. This avails us the option of terminating the branch-and-bound early even when the optimal solution has not been found. In each sub-gradient iteration, multiple Lagrangian sub-problems must be solved; hence some loss of accuracy is traded-off for computational efficiency although for most of the time, the optimal solution is found since the set-covering form is known to have a sharp LP relaxation. More discussion on balancing optimality and the need for computational speed is found in §3.2.3.

The '1' branch is always followed first because it is more likely to yield an integer feasible solution. However, the question of which fractional column-variable to branch on still remains. Here, we greedily search all non-zero columns for the one that has the

least cost, regardless of period designation. Thus, there is a higher chance of a search tree upper bound solution close to the optimum being found.

Once a branching decision has been made, constraints which fix the value of the branched column-variable are added to $MSRMP^i$ in subsequent nodes (refer to constraints (3.31)-(3.32)). Constraints which prevent the re-generation of column-variables fixed to '0' are added to the $MSPP^i$. Since column-variables fixed to '0' are not in the basis of the RMP, they could possibly have a negative reduced cost and be priced into RMP column pool. Therefore $MSPP^i$ differs slightly for each node in the search tree through the following constraint:

$$\sum_{(j,z)\in\delta^i_{jz}=1} \delta_{jz} < \sum_{(j,z)\in\delta^b_{jz}=1} \delta^{\mathrm{B}}_{jz}, \qquad \forall \delta^{\mathrm{B}}_{jz} \in \{P \mid \text{Branch-down constraints}\} \tag{3.38}$$

δ^{B}'s refer to the values of δ mapped from columns that have been fixed to '0'.

Pruning

There are instances where there is great differentiation between column costs, for example when λ_{zt}'s are spread out over a wide and distinct range of values. This is common in the early stages of the sub-gradient routine when a randomly generated starting vector is used.

In such cases, exploring every part of the feasible space, just to make mild improvements in the objective, is counter-productive and computationally intensive; we do not expect high-quality integer solutions to appear in the first few sub-gradient iterations anyway. Hence, nodes that have LP solutions that are greater than 95% of the value of the best upper bound solution are discarded. We can be assured that branch-and-price will produce solutions that are at most 5% away from the optimal cost of $MSMP^i$.

3.2.3 Updating Lagrangian Multipliers using Sub-gradient Procedure

The aim of the sub-gradient procedure is to intelligently search the dual space of the LR relaxation for the optimal Lagrangian multiplier values, λ^*_{zt}. The procedure solves the

dual problem, *MSLD* (3.15), which maximizes $L^*(\lambda)$ over all non-negative λ_{zt}'s to obtain an objective cost that is as close to the optimal primal objective as possible.

$L^*(\lambda)$ has been shown to be concave and continuous. However, $L^*(\lambda)$ is non-differentiable at any λ where there are multiple optimal Lagrangian solutions. It is at non-differentiable points that a general hill-climbing, calculus-based method would fail. The function, at these λ values, does not have a gradient, but it always has sub-gradients. [24, 25] describe the nature of the LR function and defines what a sub-gradient vector is.

Sub-gradient optimization is an iterative method was proposed by Held and Karp [22] in 1971. An initial Lagrangian multiplier vector is chosen, λ^0, and a sequence of λ is determined by updating, at each iteration, the current value of λ with a step in the direction of the sub-gradient at the point. If necessary, the resulting point is projected back into the non-negative orthant. In our case, the sub-gradient vector for the (j,z) coordinate and the rule for updating λ in the k-th iteration are respectively defined:

$$G_{zt}(\lambda^k) = \sum_s \sum_{j=1}^{H_j} \sum_{k=1}^{C_j} \delta_{sjkz}^{LS}(t) - w_z \tag{3.39}$$

$$\lambda_{zt}^{k+1} = \max\left\{0, \lambda_{zt}^k + t^k G_{zt}\right\} \tag{3.40}$$

(3.39) utilizes the Lagrangian solutions obtained using current value of λ^k. In (3.40) t^k is a positive step size; the step-size rule is discussed later. The sub-gradient procedure can be run indefinitely because only lower bounds to the primal solution are generated and there is no way to prove optimality. Ideally, the procedure terminates only when $\sum_{z=1}^{L} \sum_{t=1}^{T_{max}} \lambda_{zt}^* \left(\sum_s \sum_{j=1}^{H_j} \sum_{k=1}^{C_j} \delta_{sjkz}^{LS}(t) - w_z \right) = 0$, $\forall z,t$, which indicates that both the primal and dual optimal solutions are found. However, this is unlikely to occur because *MSLR* is sometimes solved approximately (§3.2.2.4) and a duality gap may exist. In §3.2.4, we discuss a heuristic for generating upper bound values which are used in the terminating criteria for the sub-gradient procedure.

3.2.3.1 Interpreting the Values of λ_{zt}'s

From the definition of the sub-gradient (3.39), it is observed that the way λ_{zt} is updated is based on the degree of violation or adherence of the Lagrangian solutions to the relaxed yard congestion constraints at each (j,z) coordinate. If $\sum_{s}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\delta_{sjkz}^{LS}(t) - w_z > 0$, yard congestion constraints are violated for the current iterate and λ_{zt} is increased, and vice versa.

When solving the Lagrangian sub-problems, $MSLR^i$, an increase in λ_{zt} penalizes QC activity in the (z,t) coordinate, and in the next iteration it costs more for a QC to work on containers headed from storage location z at period t. When λ_{zt} decreases, QC activity in the (z,t) coordinate costs less, and QCs are 'encouraged' to work at (z,t) to fulfill work-requirement constraints. If yard congestion constraints for (z,t) are persistently adhered to over many sub-gradient iterations, λ_{zt} will eventually be reduced to '0'; this is in effect discarding that particular yard congestion constraint from the LR problem.

The values of λ_{zt} give an indication of the likelihood of yard congestion constraints being breached at that (z,t) coordinate. Higher λ_{zt} values are expected at time periods where vessel planning horizons overlap and QCs from different vessels are working at the same time. On the other hand, in time periods where the total number of QCs from berthed ships is smaller than the yard activity threshold, it is impossible for the threshold to be breached even if the constraint is omitted. Hence, low or zero values of λ_{zt} are expected.

3.2.3.2 Choice of Lagrangian Multiplier Starting Vector

For most problem instances, the majority of the optimal λ_{zt}^*'s are equal to zero. This occurs when vessel berthing times are well spread out, and a high yard activity threshold,

w_z, is given. Hence, setting $\boldsymbol{\lambda}^0 = \mathbf{0}$ is an obvious choice, so fewer sub-gradient iterations are required to reach the optimal dual point.

However, when the problem involves many vessels docked at overlapping time intervals and small w_z's, setting $\boldsymbol{\lambda}^0$ to a small non-negative value is better as there is more 'space' for updating λ_{zt} 's as the starting vector is not a corner point. $\boldsymbol{\lambda}^0$ is fixed to -0.5 or set to a uniformly random value in the range [0, 0.5]. The perturbation lets the sub-gradient procedure converge to a different Lagrangian multiplier vector each time it is run, and therefore producing different, and hopefully better, primal solutions.

3.2.3.3 Detailed Description of Procedure

The following describes the entire sub-gradient optimization procedure:

Step 1: Initialization:

$$k = 0,\ \lambda_{zt}^0 = 0\ \ \forall z,t,\ k_{\max} = 200,\ Z_{UB} = +\infty,\ Z_{LB} = -\infty,\ N = 5,\ \zeta^0 = 1.5$$

Z_{UB} and Z_{LB} are the best upper and lower bounds to the primal solution, while N and ζ^k are parameters related to the step-size rule.

Step 2: Start k-th sub-gradient iteration. Using current values of λ_{zt}^k 's, solve S Lagrangian sub-problems by branch-and-price (§3.2.2). Calculate the value of:

$$L^*(\boldsymbol{\lambda}^k) = -\sum_{i=1}^{S}\sum_{t=d_i}^{d_i+T_i-1}\gamma_i^{LS}(t) + \sum_{z=1}^{L}\sum_{t=1}^{T_{\max}}\lambda_{zt}^k\left(\sum_{s}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\delta_{sjkz}^{LS}(t) - w_z\right)$$

If $L^*(\boldsymbol{\lambda}^k) > Z_{LB}$, then set $Z_{LB} = L^*(\boldsymbol{\lambda}^k)$

Step 3: Based on the Lagrangian solutions obtained in step 2, construct a feasible solution to the original problem, *MSP*, using primal heuristic presented in §3.2.4. The cost of the primal heuristic solution is Z^k. If $Z^k < Z_{UB}$, then set $Z_{UB} = Z^k$.

Step 4: Update the Lagrangian multiplier values for the (k+1)-th iteration using rule (3.40). The following step-size rule is used [22]:

$$t^k = \frac{\zeta\left(Z_{UB} - L^*(\lambda^k)\right)}{\sum_{(z,t)}\left(\sum_s \sum_{j=1}^{H_t} \sum_{k=1}^{C_t} \delta_{sjkz}^{LS}(t) - w_z\right)^2}$$

Justification of this formula is given in [28]. ζ^k is a parameter used in the calculation of step-size for changing λ_{zt}'s. The sequence ζ^k is determined by setting $\zeta^0 = 1$ and halving ζ^k whenever $L^*(\lambda^k)$ has failed to increase after N consecutive number of iterations.

Step 5: If any one of the following conditions is satisfied, then stop; otherwise set $k = k+1$ and go to step 2.

(1) $k = k_{\max}$

(2) Step-size parameter $\zeta^k \leq 0.005$

(3) No improvement $L^*(\lambda^k)$ after $\frac{N}{4}$ iterations

(4) Duality gap $\frac{Z_{UB} - Z_{LB}}{Z_{UB}} < \eta$, where η is user-specified

The duality gap, η, is the percentage gap between the best lower bound and the best upper bound value to the optimal cost of original problem, *MSP*. Once it is small enough, the sub-gradient routine can stop. The convergence of these bounds is discussed in §3.2.5.

3.2.4 Heuristic for Generating Primal Feasible Solutions

The iterative procedure for solving the Lagrangian sub-problems and updating the Lagrangian multipliers may fail to obtain a primal feasible solution, even when the duality gap becomes very small. Lagrangian solutions are generally infeasible in the original problem, *MSP*, and result from the violation of the yard congestion constraints.

It often happens that Lagrangian solutions are nearly feasible in the primal problem because the LR objective function penalizes large constraint violations; they can be made feasible with some judicious tinkering. Such a method is developed here and known as a *primal heuristic*. The primal heuristic attempts to correct infeasibilities while keeping objective function deterioration small, i.e. avoid increasing vessel makespans excessively.

The primal heuristic provides upper bound values, Z_{UB}, to optimal cost of *MSP* which are used in the sub-gradient procedure terminating conditions and in calculating step-sizes for updating λ_{zt}'s (§3.2.3.3).

This primal heuristic detects (*z,t*) coordinates where infeasibility occurs and attempts to swap QC work with any other QCs working on the same bay at other periods. The makespan is not increased, and no re-positioning of QCs is required. However, if this is not possible and as a last resort, the QC work is postponed to the end and makespan is increased. The heuristic is randomized at various points in the algorithm; for example, at each run of the heuristic, a different search for infeasible (*z,t*) coordinates is conducted, and QCs may swap work with different QCs. For each sub-gradient iterate, the primal heuristic is run five times and the best upper bound output, Z^k, is stored. Once no more infeasibility is detected, then the primal heuristic stops.

The following details the primal heuristic algorithm:

Step 1: Initialization:

$x_{jk}^{\text{BEST}}(t) = 0,\ \forall i,j,k,t$, $\delta_{ijkz}^{\text{BEST}}(t) = 0,\ \forall i,j,k,z,t$, $\gamma_i^{\text{BEST}}(t) = 0,\ \forall i,t$,

$\Gamma_i^{\text{BEST}} = +\infty$, Iteration $p = 1$

The variables with superscript BEST store the best feasible solution generated during multiple runs of the primal heuristic. Γ_i^{BEST} stores the best makespan value.

Step 2: If $p = 5$, then terminate algorithm. Otherwise, start the p-th try of the primal heuristic. Set:

$x_{jk}^{p}(t) = x_{jk}^{LS}(t),\ \forall i,j,k,t$, $\delta_{ijkz}^{p}(t) = \delta_{ijkz}^{LS}(t),\ \forall i,j,k,z,t$, $\gamma_i^{p}(t) = \gamma_i^{LS}(t),\ \forall i,t$,

$\Gamma_i^{p} = T_i - \sum_{t=d_i}^{d_i+T_i-1} \gamma^{LS}{}_i(t)$

The decision variables for the p-th primal heuristic iterate are initialized to the Lagrangian solution values. The relationship $\Gamma_i^{p} \le \Gamma_i^{\text{BEST}},\ \forall i,p$ applies as the makespan may be compromised when correcting infeasibilities.

Step 3: Scan all [*z*,*t*] coordinates for yard congestion constraint violation. If there is infeasibility, randomly pick one infeasible point, [*z'*,*t'*], and go to step 4. Otherwise, if $\Gamma_i^p < \Gamma_i^{\text{BEST}}$ then set:

$x_{jk}^{\text{BEST}}(t) = x_{jk}^p(t),\ \forall i,j,k,t\ ,\ \delta_{ijkz}^{\text{BEST}}(t) = \delta_{ijkz}^p(t),\ \forall i,j,k,z,t\ ,\ \gamma_i^{\text{BEST}}(t) = \gamma_i^p(t),\ \forall i,t\ ,$

$\Gamma_i^{\text{BEST}} = \Gamma_i^p$, Primal heuristic output cost $Z^k = -\sum_{t=d_i}^{d_i+T_i-1} \gamma_i^{\text{BEST}}$

Set $p = p + 1$ and go to step 2.

Step 4: Check the degree of infeasibility in [*z'*,*t'*]. If no more infeasibility detected, go to step 3. Otherwise, go to step 5.

Step 5: Search for ships and bays ([*i*,*j*] coordinates) that have QCs working at [*z'*,*t'*]. Randomly pick a ship and bay, [*i'*,*j'*]. The QC working at [*i'*,*j'*,*z'*,*t'*] will be considered for swapping. If none are found, then go to step 10; otherwise, go to step 6.

Step 6: Search for QCs positioned in ship *i'* bay *j'* at other time periods. If found at time period *t"*, go to step 7; otherwise, go to step 5.

Step 7: Check if yard congestion constraints will be breached at time *t"* if swapping of work is done between QCs positioned in ship *i'* bay *j'* at times *t'* and *t"*. If they are breached, go to step 6. Otherwise, go to step 8.

Step 8: Check if the QC positioned at ship *i'* bay *j'* at time *t"* is working. If working, go to step 9. Otherwise, identify the which *z*-coordinate it is working on, *z"*, and swap the QC work between coordinates [*i'*,*j'*,*z"*,*t"*] and [*i'*,*j'*,*z'*,*t'*]; set:

$$\delta_{i',j',k,z'}(t') = 0,\ \delta_{i',j',k,z''}(t') = 1,\ \delta_{i',j',k,z''}(t'') = 0,\ \delta_{i',j',k,z'}(t'') = 1$$

Then, go to step 4.

Step 9: Check if yard congestion constraints will be breached at time *t'* if swapping of work is done between QCs positioned in ship *i'* bay *j'* at times *t'* and *t"*. If they are breached, go to step 6. Otherwise swap the QC work between coordinates [*i'*,*j'*,*z"*,*t"*] and [*i'*,*j'*,*z'*,*t'*]. Then, go to step 4.

Step 10:Search for any ship and bay with a QC working at [*z'*,*t'*]. The coordinate at this point is $[\hat{i},\hat{j},z',t']$. QC work done here is re-scheduled to the back, causing the

makespan to increase by 1; set $\Gamma_i^p = \Gamma_i^p + 1$. The QC position-assignments at t' are replicated at the new Γ_i^{LR} to maintain clearance. Then set:

$$\delta_{i,j,k,z'}(t') = 0,\ \delta_{i,j,k,z'}(\Gamma_i^{LR}) = 1$$

Go to step 4.

3.2.5 Convergence of Upper and Lower Bounds

The Lagrangian solutions provide a lower bound, Z_{LB}, to the optimal primal cost, while feasible solutions generated by the primal heuristic provides an upper bound, Z_{UB}. Computational experience shows that almost equivalent near-optimal Lagrangian multipliers can result in primal solutions of substantially different quality. Therefore, it is worthwhile applying the primal heuristic to Lagrangian solutions in every sub-gradient iterate to improve chances of obtaining a good Z_{UB} value.

The sub-gradient procedure terminates once the duality gap, η, falls below a user-specified value. In our case, convergence of $L^*(\lambda^k)$ is quick and fairly reliable. At the optimal λ_{zt}^*'s, the lower bound values closely approximate the optimal primal objective, providing bounds tighter than the LP relaxation (see §3.2.6 for comparison between LP-relaxed and LR bounds). Figure 3-5 shows the lower bound value (blue line) compared to the corresponding upper bound value (red line) generated in each sub-gradient iterate, when a starting vector of $\lambda^0 = 0.5$ is used. In this example, the optimal primal solution obtained from CPLEX is -40, while the best upper bound value found after 100 iterations is -39. The lower bound converges at iteration 32 to within 1% of the best solution to *MSLD* (3.15). The best value of $L^*(\lambda)$ found is -40.019, resulting in a final duality gap of 2.61%. Although there is no strict correlation between the value $L^*(\lambda)$ and the quality of the primal solutions found, it is observed that there is a greater tendency for high-quality solutions to be produced by the primal heuristic when the duality gap is small.

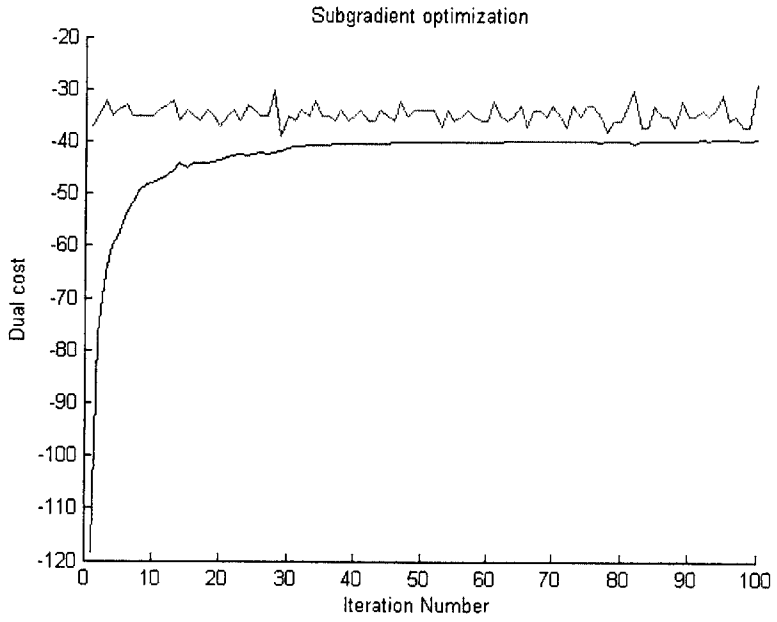


Figure 3-5. Upper and lower bound values vs. sub-gradient iterations for the case of S=2, H_1=10, H_2=10, C_1=2, C_2=2, T_1=32, T_2=44, w_z=1

Because the problem is *np*-hard, some degradation from optimal vessel makespan values is expected and accepted; it is more important to ensure that yard congestion constraints are not violated and a guarantee on the quality of the generated solutions is provided. The interested reader is reader is referred to [29] for a description of the theoretical convergence of the sub-gradient procedure when there is a duality gap.

3.3 Computational Results

3.3.1 Test Problems

A data set with 7 different problem instances was created, arranged in order of increasing number of vessels from 2 to 8. The number of bays, number of QCs allocated, load profile and berth time for each vessel is provided together with the QC clearance requirements, as shown in Table 3-1. Vessels with 20 or more bays are used in some problem instances for simulate realistic conditions. The problem instances are generated such that there is considerable overlap in the berthing periods of the vessels (i.e. d_i's are close to one another), thus we can test fully the effectiveness of the primal heuristic in

removing yard congestion constraint violations. The number of constraints and variables in the original primal problem for each problem instance is also given.

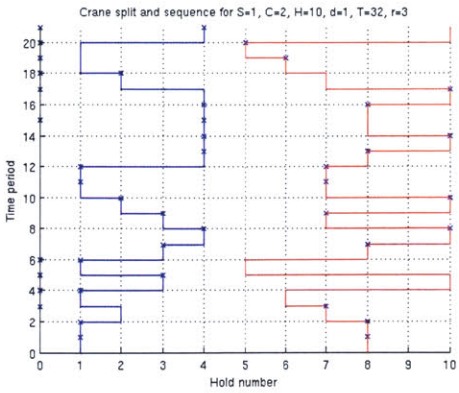
Computational experiments on this data set will validate the accuracy of the exact multiple-ship model formulation. It will also determine the quality of lower bound values produced by the Lagrangian framework and sub-gradient procedure, compared to the LP relaxation, and the computational performance of branch-and-price approach in solving the Lagrangian sub-problems.

Table 3-1. Description of the data set of test problems for the multiple-ship model

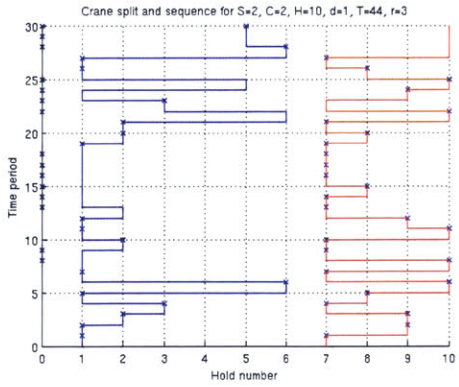
Data code	S	H_i	C_i	r	d_i	T_i	No. of constraints	No. of variables
MSD2	2	10,10	2,2	2	1, 4	32, 44	18423	9196
MSD3	3	10,10,15	2,2,3	3	1, 5, 10	32, 44, 73	76161	43394
MSD4	4	10,10,15,15	2,2,3,3	3	1, 1, 8, 14	32, 44, 73, 50	110715	63694
MSD5	5	10,10,15,15, 20	2,2,3,3, 4	3	1, 8, 20, 40, 65	32, 44, 73, 50, 79	204757	120653
MSD6	6	10,10,15,15, 20,25	1,1,2,2, 2,3	5	1, 8, 20, 40, 65, 65	32, 44, 73, 50, 79, 89	231990	128932
MSD7	7	10,10,15,15, 20,25,25	1,1,2,2, 2,3,3	6	1, 8, 20, 40, 65, 65, 41	32, 44, 73, 50, 79, 89, 111	359729	203968
MSD8	8	10,10,15,15, 20,25,25,25	1,1,2,2, 2,3,3,3	6	1, 8, 20, 40, 65, 65, 41, 10	32, 44, 73, 50, 79, 89, 111, 80	451849	258048

3.3.2 Results and Analysis

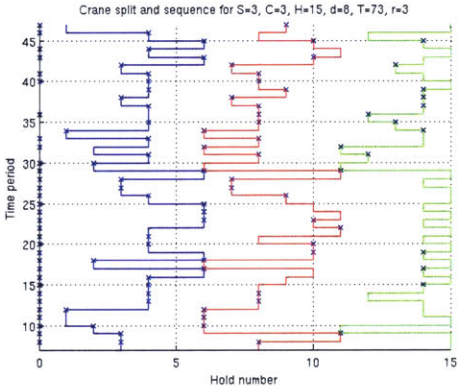
3 sets of computational experiments are performed on the abovementioned data set, labeled as the 'easy', 'moderate' and 'hard' runs. The level of difficulty of each run of the problem set is determined by the value of the yard activity threshold, w_z; for difficult problems, w_z is small and yard congestion constraints are easily violated, and vice versa. An example of the output generated from applying the entire Lagrangian framework and primal heuristics on the 'hard' problem instance with 4 vessels is shown in Figure 3-6. In this case, the Lagrangian sub-problems are solved directly by CPLEX.



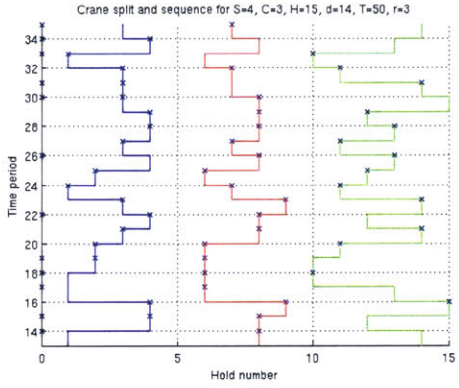
3-6(a) Vessel 1 QC schedule



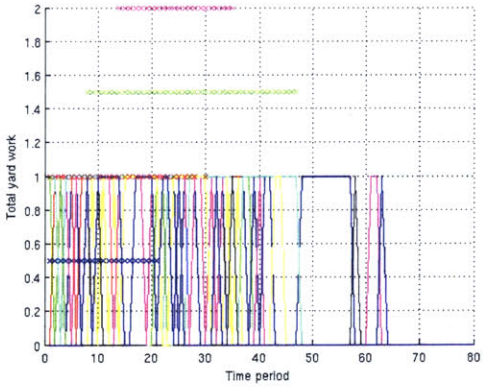
3-6(b) Vessel 2 QC schedule



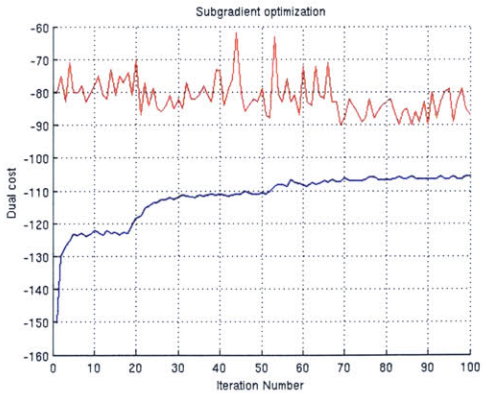
3-6(c) Vessel 3 QC schedule



3-6(d) Vessel 4 QC schedule



3-6(e) Yard activity for 8 storage locations



3-6(f) Upper and lower bound convergence of sub-gradient routine

Figure 3-6. Matlab output of problem instance MSD4 using Lagrangian framework

Figures 3-6(a)-3.6(d) show feasible QC schedules for each of the 4 vessels. We observe that QCs remain idle for relatively more time periods (shown by crosses at H=0 in the x-axis) as compared to QC schedules of the single-ship model, to avoid breaching the yard congestion constraints. For 'hard' problem instances, w_z is set to '1' for all z's. In this example, the amount of yard activity per period for each of the 8 storage locations is shown in Figure 3-6(e), verifying that total yard activity in each period for all z's never exceeds 1. The crosses represent the individual planning horizons (with length T_i) for each vessel, and give an overall view of how compact the berthing schedule is and therefore, the corresponding difficulty of the problem. In this case, the primal heuristic has re-scheduled QC work to beyond the master planning horizon, which ends at period 47. Figure 3-6(f) shows the upper and lower bound values for each iterate in the sub-gradient route. The lower bound values plateau from iterate 5 to 22 because CPLEX was terminated (based on 500-second time limit) before optimal values for $MSLR^i$ are found. Subsequently, optimality for $MSLR^i$ is reached and convergence continues unaffected toward the eventual dual solution of -105.37. It can be observed from Figure 3-6(f) that better upper bound values are provided as the duality gap is reduced. Eventual values of $Z^{UB} = -90$ and duality gap of 17% are produced after 100 iterations.

Computational results for the 'easy', 'moderate' and 'hard' runs are shown in Table 3-2. For the 'easy' runs, w_z is set to a very low value and as a result, the yard congestion constraints do not restrict the feasible space. The primal heuristics perform no work at all as no infeasibilities are detected from the Lagrangian solutions. For 'moderate' and 'hard' runs, much lower values of w_z are used. Furthermore, the exact mathematical model of the multiple-ship formulation and its LP relaxation are solved with CPLEX to compare the quality of the lower bound values.

Table 3-2. Exact solution, LP and Lagrangian cost output and computational performance of the multiple-ship data set

Level of difficulty	Data code	L (w_2)	Makespan (IP)	Makespan (Lag.)	Z_{IP}	Z_{LB}^{*} / Z_{UB}^{**}	Z_{LP}	Duality gap (%)	CPU Time (Z_{IP}) (s)	CPU Time (Lag.) (s) / No. sub-grad. iterations
	MSD2	5 (8)	16, 22	16, 22	-40	-40 / -40	-66.00	0	3.202	2.377 / 1
	MSD3	8 (8)	17, 26, 35	17, 26, 35	-74	-74 / -74	-133.00	0	15.78	11.62 / 1
	MSD4	8 (8)	17, 26, 35, 19	17, 26, 35, 19	-106	-106 / -106	-180.00	0	25.86	17.71 / 1
	MSD5	8 (8)	17, 26, 35, 19, 24	17, 26, 35, 19, 24	-162	-162 / -162	-254.86	0	60.96	39.20 / 1
'Easy'	MSD6	8 (8)	32, 44, 49, 29, 40, 33	32, 44, 49, 29, 40, 33	-146	-146 / -146	-316.72	0	171.28	121.32 / 1
	MSD7	8 (8)	32, 44, 52, 32, 40, 34, 41	32, 44, 52, 32, 40, 34, 41	-210	-210 / -210	-418.22	0	473.90	242.46 / 1
	MSD8	8 (8)	-	32, 44, 52, 32, 40, 34, 41, 29	⌂	-262 / -262	-493.22	0	Out of memory	295.16 / 1
	MSD2	5 (2)	16, 22	16, 22	-40	-40 / -40	-66.00	0	8.460	4.817 / 2
	MSD3	8 (2)	17, 26, 35	17, 26, 35	-74	-74 / -74	-132.68	0	17.41	24.08 / 2
	MSD4	8 (2)	17, 27,6, 35, 19	17, 27, 35, 19	-106	-106 / -105	-179.27	0.97	53.20	71.54 / 4
	MSD5	8 (2)	17, 26, 35, 19, 24	17, 26, 35, 19, 24	-162	-162 / -162	-254.70	0	71.11	39.16 / 2
'Moderate'	MSD6	8 (2)	32, 44, 49, 29, 40, 33	32, 44, 49, 29, 40, 33	-146	-146 / -146	-313.49	0	242.76	121.23 / 2
	MSD7	8 (2)	-	32, 44, 53, 32, 41, 34, 41	⌂	-208 / -210	-414.99	0.96	Out of memory	240.76 / 1
	MSD8	8 (2)	-	32, 44, 52, 32, 41, 34, 41, 30	⌂	-262 / -260	-489.99	0.77	Out of memory	294.20 / 1
	MSD2	5 (1)	16, 22	17, 23	-40	-40.078 / -38	-62.65	5.46	13.63	270.63 / 100
	MSD3	8 (1)	-	18, 28, 35	-73◊	-74.040 / -71	-119.62	4.28	Too long	733.34 / 87
	MSD4	8 (1)	-	21, 30, 40, 22	-105◊	-105.37 / -90	-151.56	17.08	Too long	3043 / 100
	MSD5	8 (1)	-	17, 27, 35, 19, 24	⌂	-161.77 / -161	-242.74	0.47	Out of memory	1271 / 94
'Hard'	MSD6	8 (1)	-	32, 44, 52, 33, 47, 37	⌂	-185.82 / -130	-292.24	42.94	Out of memory	1736 / 2 Δ
	MSD7	8 (1)	-	32, 49, 66, 48, 59, 48, 45	⌂	-268.32 / -150	-414.99	78.89	Out of memory	3965 / 2 Δ
	MSD8	8 (1)	-	38, 53, 63, 39, 57, 45, 49, 30	⌂	-328.60 / -192	≈-440□	71.14	Out of memory	2182 / 2 Δ

*Lagrangian sub-problems solved using CPLEX

**Z^{UB} obtained from applying primal heuristic on Lagrangian solutions

◊ Best sub-optimal primal integer solution

□ Best sub-optimal primal LP solution

⌂Unable to generate any primal feasible solution

Δ Sub-gradient procedure stopped before terminating criteria is met

Z_{IP} and Z_{LP} refer to the optimal primal cost and optimal LP solution respectively, while Z_{LB} and Z_{UB} respectively denote the optimal dual solution and best primal feasible solution generated by the Lagrangian framework. A brief inspection of Z_{LP} and Z_{LB} values shows that the Lagrangian framework produces superior bounds compared to the LP relaxation; in some cases for the 'easy' and 'moderate' runs, the ideal result of $Z_{LB} = Z_{IP}$ is obtained. When tackling the exact model for larger problem instances, they become too large for CPLEX to even initialize the solving process because of the total memory size required to solve the LP relaxation (i.e. 'Easy': MSD8). The poor bounds for LP relaxation also suggest that CPLEX will perform poorly in its LP-based branch-and-bound algorithm even for smaller instances; this is seen when running 'hard' problem instances (i.e. 'Hard': MSD3, MSD4).

The Lagrangian framework is applied on all 3 runs of the data set. CPLEX is used to solve Lagrangian sub-problems, subject to a cut-off time of 500 seconds and using a best-bound search strategy. For the 'easy' run, a starting Lagrangian multiplier vector of all zeros is used. Since yard congestion constraints are extremely lax and are also absent in the LR cost objective (because of $\boldsymbol{\lambda}^0 = 0$), QC activity between vessels do not interact in the yard and it is akin to solving S independent single-ship problems. The sub-gradient procedure halts after a single iteration with a zero duality gap and is more efficient compared solving to the original problem.

For the 'moderate' run, a starting vector of $\boldsymbol{\lambda}^0 = 0$ is used as well, because we do not expect many of the optimal λ_{zt}^*'s to be non-zero. A maximum of 4 sub-gradient iterations among all the problem instances is required to reduce the duality gap to below 1%. Mild infeasibilities in the Lagrangian solutions are easily corrected by the primal heuristic to produce high quality upper bound values. Again, it appears to be advantageous to use the LR framework over a directly application of CPLEX to the original problem.

For the 'hard' run, almost none of the problem sets are solved to optimality. A starting vector of $\boldsymbol{\lambda}^0 = 0.1$ is used and the sub-gradient procedure is terminated once

$\eta < 0.05$ or 100 iterations are reached. A maximum duality gap of 17.08% is found for smaller problems MSD2 to MSD5. As for problems MSD6 to MSD8, the sub-gradient procedure is artificially terminated because of excessive times needed by CPLEX to solve the Lagrangian sub-problems. Hence, low-quality upper bound values are obtained because primal heuristics are applied to Lagrangian solutions derived from non-optimal λ_{zl}'s.

Table 3-3 shows the computational performance and bounds generated when the Lagrangian sub-problem is solved using branch-and-price for the 'hard' runs; the best lower bound value obtained is denoted by Z_{LB-CG}. The final duality gap obtained for the branch-and-price approach is comparable to the CPLEX approach for smaller instances

Table 3-3. Comparison of computational performance of 'hard' problem instances when Lagrangian sub-problems are solved by CPLEX and branch-and-price

Data code	L / w_z	Z^{IP}	Z_{LB}* / Z_{UB} (% Gap)	$Z_{LB\text{-}CG}$** / Z_{UB} (% Gap)	CPU Time (Z_{LB}) (s) / No. subgrad. Iterations	CPU Time ($Z_{LB\text{-}CG}$) (s) / No. subgrad. Iterations
MSD2	5 (1)	-40	-40.08 / -38 (5.46)	-40.019 / -39 (2.68)	270.63 / 100	2743 / 85
MSD3	8 (1)	-73$^{\diamond}$	-74.04 / -71 (4.28)	-74.85 / -71 (5.42)	733.34 / 100	3003 / 100
MSD4	8 (1)	-105$^{\diamond}$	-105.37 / -90 (17.08)	-105.62 / -91 (16.07)	3043 / 100	3543 / 100
MSD5	8 (1)	⌂	-161.77 / -161 (0.47)	-161.20 / -158 (2.02)	1271 / 94	3295 / 69
MSD6	8 (1)	⌂	-185.82 / -130 (42.94)	-146.45 / -141 (3.86)	1736 / 2 $^{\Delta}$	5976 / 97
MSD7	8 (1)	⌂	-268.32 / -150 (78.89)	-215.54 / -192 (12.26)	3965 / 2 $^{\Delta}$	3342 / 20 $^{\Delta}$
MSD8	8 (1)	⌂	-328.60 / -192 (71.14)	-283.93 / -246 (15.41)	2182 / 2 $^{\Delta}$	4332 / 20 $^{\Delta}$

*Lagrangian sub-problems solved using CPLEX
** Lagrangian sub-problems solved using branch-and-price
⌂Unable to generate any primal feasible solution
Δ Sub-gradient procedure stopped before terminating criteria is met

MSD2 to MSD5. For MSD6 to MSD8, significantly smaller duality gaps are obtained because the sub-gradient procedure is allowed to run for an increased number of iterations, resulting in a better convergence of upper and lower bounds. This arises because the specialized branch-and-price algorithm solves larger instances of $MSLR^i$ faster than CPLEX does and more sub-gradient iterations can be carried out in the same time. However, for smaller problem instances it appears that using CPLEX confers a greater computational advantage.

Summarizing the computational results, we find that in cases where the yard congestion constraints are 'lax' and a zero starting vector is used, the LR framework (with sub-problems solved by CPLEX) is the preferred approach and solves 100% of problem instances at computation times up to half that where the original problem is tackled directly. The number of sub-gradient iterations required to obtain duality gaps of less than 1% is extremely small. However, when we 'tighten' the yard congestion constraints, LR appears to be the only practical approach as CPLEX fails to initialize when attempting to solve the original problem even for relatively smaller instances. The use of branch-and-price to solve $MSLR^i$ appears to be less efficient compared to CPLEX's performance for sub-problems of smaller vessels comprising up to 20 bays, but was superior for the larger sub-problems of 20 bays or more.

Chapter 4

Summary and Future Directions

In this thesis, we have incorporated the QC clearance and yard activity restriction requirements, required by a major container terminal port, as new constraints in the original linear IP model proposed by Daganzo in his 1989 paper. The model is further strengthened by disaggregating the new constraints so that sharper LP relaxation bounds are obtained. During computational experiments, the performance of the state-of-the-art CPLEX IP solver was used as a baseline standard for evaluating the computational efficiency of our proposed solution approaches.

We have articulated formal mathematical definitions of the single-ship and multiple-ship model, with the latter taking into account the occurrence of many ships docking at close intervals with each other and the possibility of congestion forming in the yard from high loading and discharging activity. For the single-ship model, a heuristic approach, adapted from Daganzo's scheduling principles, is proposed which has no optimality guarantee but produces high-quality solutions in practice. A novel branch-and-price approach, with an internal column generation routine influenced by Gilmore's cutting stock problem (proposed in 1961), is developed which solves the single-ship model to optimality. Both methods may be used in-lieu of a direct tackling of a problem, whose size may be in the order of 100,000 variables; they can be applied in less than 1% of the time CPLEX requires to solve the original problem.

The solution approach to the multiple-ship model was developed based on the successful methods of efficiently tackling the single-ship problem. We proposed a method based on a combination of Lagrangian relaxation, a sub-gradient iterative procedure and primal heuristics. The basic scheme and various enhancements were discussed. Conveniently, we noted that relaxing the yard congestion constraints would allow for the separation of the original primal problem into independent sub-problems which have the same structure as the single-ship model. This created the opportunity to develop a modified branch-and-price method, originally proposed for the single-ship model, to solve the Lagrangian sub-problem. Good primal solutions were achieved by effectively utilizing information generated by the dual, sub-gradient procedure; namely, applying the primal heuristic to Lagrangian solutions. Computational results were affected considerably by the value of the yard activity threshold. When this value was high, zero duality gap outputs could be obtained, however for lower threshold values, we would have to tolerate varying levels of sub-optimality. Nevertheless, in all cases, our proposed Lagrangian relaxation technique is the recommended approach, in terms of computational efficiently, for solving the multiple-ship problem.

In the future, we can consider altering the model to better represent real-world conditions and to improve on port operational efficiencies. For example, a good objective would be to reduce the amount of gantrying done by the QCs when working. Either a cost term associated with the magnitude of QC movement from period-to-period may be added to the objective, or additional constraints can be added to restrict the total amount of gantrying within a specified period. The solution methodologies would have to be modified accordingly.

Most of the computational effort is spent in solving the Lagrangian sub-problems. As such, an efficient algorithm for tackling the sub-problems is critical to solving large-scale multiple-ship problems. However, for the few problem instances tested, the proposed branch-and-price approach appears to confer little computational advantage over CPLEX. Although branch-and-price performs better for larger problems, the overall algorithm still requires a significant amount of time for convergence, especially when there are more

than 6 vessels. More extensive testing is required to achieve clear conclusions on the applicability of branch-and-price within Lagrangian relaxation. When we attempted solving problem instances with 10 or more vessels, the entire algorithm slowed down considerably even when the each of the vessels had only 10 bays or less. This suggest a possible limitation of the OPL Studio platform in handling large codes with multiple nested loops and numerous optimization models instantiated in memory, rather than reflecting the effectiveness of the method or the level of difficulty of the Lagrangian sub-problems. Migration of the code to C++ and the use of the associated compiler could demonstrate the full potential of the proposed approach and would be the next logical step; nonetheless, this preliminary study suggests that this approach is promising.

Works Cited

[1] Muller, G. (1999), Intermodel Freight Transportation, *Eno Transportation Foundation, Inc.*

[2] United Nations Conference on Trade and Development (2004), Geneva, *Review of Maritime Transport.* Chapter 5, Port Development

[3] K.G. Murty, J.Liu, Y. Wan, R. Linn (2005), A decision support system for operations in a container terminal. *Decision Support Systems*, **39,** No. 3: 309-332

[4] C.Y. Zhang (2002), A Heuristic for Real-time Container Load Sequencing. *Master's Thesis*, HPCES, Singapore-MIT Alliance

[5] H.D. Duong (2002), Automatic Crane Sequencing, *Master's Thesis*, HPCES, Singapore-MIT Alliance

[6] D. Steenken, S.Vob, R. Stahlbock (2004), Container terminal operations and operations research – a classification and literature review. *OR Spectrum*, **26**: 3-49

[7] Peterkofsky, R.I., C.F. Daganzo (1990), A Branch and Bound Solution Method for the Crane Scheduling Problem, *Transportation Research*, Part B, Vol. 3, **24**:159-172

[8] P. van Hentenryck (1999), *The Optimization Programming Language*, MIT Press

[9] P van Hentenryck, L. Michel (2000), OPL Script: Composing and Controlling Models, *New Trends in Constraints*, Lecture Notes in Artificial Intelligence (LNAI 18), Springer Verlag

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

[11] R.I. Peterkofsky, C.F. Daganzo (1990), A branch and bound solution method for the crane scheduling problem, *Transportation Research,* Vol.3, **24B**: 159-172

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

[13] Y.M Park, K.H. Kim (2002), Berth scheduling for container terminals by using a sub-gradient optimization technique, *Journal of Operational Research Society*, **53**: 1054-1062

[14] E.K. Bish (2003), A multiple crane constrained scheduling problem in a container terminal. *European Journal of Operational Research*, **144**: 83-107

[15] Bixby et. al (2002), *Mixed-integer programming: A progress report.*

[16] Wosley, L (1998), *Integer Programming*. John Wiley, New York

[17] C. H. Papadimitriou (1981), On the complexity of Integer Programming. *Journal of the ACM*, **28**, Issue 4: 765-768

[18] C. Barnhart, et al. (1998), Branch-and-price: Column generation for solving huge integer problems, *Oper. Res.*, **46** (3): 316-329

[19] P.C. Gilmore, R.E. Gomory (1961), A Linear Programming Approach to the Cutting Stock Problem. *Operations Research*, **9**: 849-859

[20] A. Atamturk, M. Savelsberg (2005), Integer-Programming Software Systems. *Annals of Operations Research*, **140**: 67-124

[21] M. Lubbecke, J. Descrosiers (2005), Selected Topics in Column Generation, *Oper. Res.*, **53** (6): 1007-1023

[22] M. Held, R. Karp (1970), The Travelling Salesman Problem and Minimum Spanning Trees, *Operations Research*, **18**: 1138-1162

[23] A.M. Geoffrion (1974), Lagrangean Relaxation for Integer Programming, *Mathematical Study*, **2**: 82-114

[24] M.L. Fisher (1981), The Lagrangian Relaxation Method for Solving Integer Programming Problems, *Management Science*, **27** (1): 1-18

[25] M. Guignard (2003), Lagrangean Relaxation. *Top*, **11** (2): 151-228

[26] D. Bertsimas, J. Tsitsiklis (1997), *Introduction to Linear Optimization*. Athena Scientific

[27] K. Hoffman, Combinatorial Optimization: Current Successes and Directions for the Future. *Working Paper*, http://iris.gmu.edu/~khoffman

[28] P. Wolfe, H.D Crowder (1974), Validation of Subgradient Optimization, *Mathematical Programming*, **6**: 62-88

[29] Allen, E, *et. al* (1987), A generalization of Polyak's convergence result for subgradient optimization, *Math. Programming*, **37**: 309-317