A FRAMEWORK FOR GENERATING EFFICIENT YARD PLANS FOR MARINE CONTAINER TERMINALS

KU LIANG PING

(Master of Science, National University of Singapore)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2011

Acknowledgements

I am grateful to have two cheerful children, Jovan and Shannon, to be with me, who continue to inject motivation into my life. I am also thankful to my wife, Jesline, for taking good care of our two children while I am busy with this thesis. We thank our parents, who help us selflessly in times of need.

Great appreciation goes to my supervisors, A/Prof. LEE Loo Hay, A/Prof. CHEW Ek Peng and A/Prof. TAN Kok Choon, for their advice and opinions that steer me through the research, hence making this thesis possible.

Lastly, I would also like to thank my friends Joon Leng, Pang Jin, Rajiv and Tian Heong, for offering their help to improve my writing.

Liang Ping

Contents

Summary

As a result of globalisation, increasingly more cargoes are transported across the globe in marine containers as it the most cost-effective means of transportation. Container terminals must be efficient in order to meet the shipping community's demand. Terminal efficiency depends heavily on the efficiency of storage and retrieval of containers from the yard, and the most important factor in determining yard efficiency is the yard plan.

Yard planning is the decision of where to stack the containers in the yard. Among many strategies, consolidated strategy seems to be the preferred strategy by many terminals in the world, where containers to be loaded into the same vessel are stacked in groups. These locations are optimally chosen to improve yard efficiency, such that no two groups of containers are stacked in close vicinity if they are to be loaded simultaneously. Due to the cyclical behaviour of vessels arrival schedule, a yard plan template - plan that repeats on a weekly basis - can be generated.

We define a generic yard plan template problem specification where a variety of yard plan strategies can be represented. We formulate a mixed integer mathematical programme to model this problem. Two solution approaches are presented, namely solving the mathematical programme using CPLEX, and a heuristic local search algorithm. Experiments with the consolidated strategy show that scenarios where yard ranges are non-dedicated can be solved by CPLEX efficiently, while scenarios with dedicated yard ranges are best solved with the heuristic approach.

Next, we consider uncertainty in the vessels' arrival schedule. For the case of con-

solidated strategy, changes in vessel arrival schedule may cause congestion of trucks at yard locations where groups of containers in the near vicinity are loaded simultaneously. While the community for robust optimisation may suggest having a robust plan that remains feasible when subjected to uncertainty, we want to find a solution that allows us to change easily when the uncertainty is revealed - a nimble yard plan template. The decision process has two stages - stage 1 finds a nimble solution, and stage 2 applies a recovery policy to change the plan after the uncertainty is revealed. We consider three cases of the nimble yard plan template problem by varying the recovery policy. We explore local search heuristics that enable us to find good solutions for the first two cases. Experiments show that nimble plan gives a better yard plan in situations with uncertain vessel arrival schedule. The experiments also show that the problem is harder to solve as the recovery policy increases its flexibility. The third case, being the most generic, could not be solved with the proposed heuristic algorithm, and we provide an intuitive explanation of the complexity.

Motivated by the nimble yard plan template problem, and many other real life problems that require decisions to be nimble, we define a generic formulation for this class of problem, called Nimble Optimisation. We show that our formulation is a generalisation of a few other related works that we have reviewed. We present a distributed solution architecture approach to solve this class of problems.

Keywords: yard plan strategy, yard plan template, optimisation under uncertainty, nimble optimisation, mixed-integer programming, heuristic algorithms.

List of Tables

List of Figures

List of Abbreviations and Notations

AGV	:	Automated Guided Vehicle
ARC	:	Adjustable Robust Counterpart
ASC	:	Automatic Stacking Crane
CARO	:	Complete Adaptable Robust Optimisation
CPLEX	:	A commercial software that solves **MIP**
FARO	:	Finite Adaptable Robust Optimisation
GA	:	Genetic Algorithm
GPS	:	Global Positioning System
ILP	:	Integer Linear programme
IMPR	:	Improved **ORIG** model without using big-M
IMPR-M	:	Improved **ORIG** model using big-M
IMPR-u	:	IMPR with u as an input instead of decision variable
ITT	:	Inter-Terminal Transport
LOSM	:	Loading Optimised Surrogate Model
LRR	:	Linear Recovery Robust problem
LSA-HC	:	Loading Separation Assignment and Hill Climbing local search
MIP	:	Mixed Integer Programme
Nimo	:	Nimble Optimisation
ORIG	:	Original yard plan template model
ORIG-Relax	:	Relaxed **ORIG** model with s_r^{20} and s_r^{40} set as real

RA	:	Recovery Algorithm with **strict assignment policy**
RA-1	:	Recovery Algorithm with **one change policy**
RegO	:	Regret Optimisation
RMG	:	Rail Mounted Gantry cranes
SOY	:	Soyster's Model
SGLS	:	Simulation Guided Local Search Algorithm
SGLS-R	:	Simulation Guided Local Search with recovery **RA**
SGLS-R1	:	Restricted Simulation Guided Local Search with recovery **RA-1**
TEU	:	Twenty-foot Equivalent Unit
TOS	:	Terminal Operating System

List of Terminology

20-footers	:	Containers of 20 feet length
40-footers	:	Containers of 40 feet length
20-footer slot	:	A slot that can **stack 20-footers**
40-footer slot	:	A pair of slots that can **stack 40-footers**
Activities	:	**Storage activities** and **retrieval activities** collectively
Activity concentration	:	Number of **activities** within a defined yard location violating the pre-defined threshold
Cluster	:	A set of consecutive **ranges** that will be assigned as a group
Discharging	:	Moving containers out of the vessel
Loading	:	Moving containers into a vessel
Mount	:	Put a container on a **truck**
Nimble optimisation	:	A problem that is subject to uncertainty, such that the solution can be changed easily to an optimal solution when uncertainty is revealed
Nimble yard plan template	:	A **yard plan template** that can be changed easily to an optimal solution when uncertainty is revealed
Offload	:	Remove a container from a truck
One change policy	:	A recovery policy that allows only one of the assignment of services to ranges to change
Port of rotation	:	A cyclical list of marine container ports that a vessel will visit in sequence
Quay crane	:	A lifting crane that works at the quay

Range	:	A set of consecutive slots
Retrieval activity	:	Activity of taking a container from the yard
RMG contention	:	Number of **activities** within a yard block violating the pre-defined handling capacity of the **RMGs** in the block
Services	:	A set of vessels that have the same **port of rotation**
Stack	:	Put a container on top of another
Static model	:	**Yard plan template** model that has deterministic inputs
Strict assignment policy	:	A recovery policy that does not allow changes to the assignment of services to ranges
Stubborn policy	:	A recovery policy that does not allow any changes to the plan at all
Storage activity	:	Activity of putting a container into the yard
Truck	:	A vehicle that transport containers
Violations	:	Any violations to the **activity concentration** and **RMG contention** collectively
Workload	:	The ratio of total number of **TEU**s to be loaded in a week to the maximum stacking capacity of the entire yard in **TEU**s
Yard crane	:	A lifting crane that works at the yard
Yard planning	:	The decision of where to **stack** the containers in the yard when they arrive, while considering both the efficiency of **storage** and **retrieval activities**
Yard plan strategy	:	A set of rules that constrains the choice of yard locations that each container may be stacked
Yard plan template	:	A yard plan that repeats on a weekly basis

List of Symbols

Ω = Uncertainty set

ω = Uncertainty vector

δ = Maximum perturbation of ϵ_i

ϵ_i = Number of time periods where service i's arrival time deviates from original schedule

$\mathcal{A}_i$ = Derived from $\mathcal{L}_i$, the set of time periods that are after the loading for service i,

$\mathcal{B}$ = Set of blocks where each element b is a set of ranges that are in the block

$\mathcal{C}$ = Set of clusters where each element c is a set of ranges that can be combined to form a cluster

$\mathcal{D}_\mathcal{T}$ = $(\mathcal{V}, DA_{it}^{20}, DA_{it}^{40}, \mathcal{L}_i)$

$\mathcal{G}$ = Set of groups where each element g is a set of ranges that belong to the group

$\mathcal{L}_i$ = Set of time periods where each element t is the time when service i is loading its containers

$\mathcal{R}$ = Set of ranges

$\mathcal{S}_\mathcal{T}$ = $(\mathcal{R}, \mathcal{G}, \mathcal{B}, \mathcal{C}, CS_r, SR_r, MA_g, RY, MY_b, W^k, W^p, SH, OS)$

$\mathcal{T}$ = Set of time periods

$\mathcal{V}$ = Set of services

$\mathbb{A}$ = Abstraction of constraint matrix of static model

$\mathbb{B}$ = Abstraction of right hand side of static model

$\mathbb{C}$ = Abstraction of cost vector of static model

CS_r = Capacity in TEU for a slot in range r

DA_{it}^{20} = Number of discharge 20-footers received by service i at time t

DA_{it}^{40} = Number of discharge 40-footers received by service i at time t

M = Big Number

MA_g = Maximum amount of activity allowable in group g in each time period

MY_b = Number of RMG in block b

OS = 1 if the yard plan template is in one-slot mode

$Q_{ii'}$ = Time gap between service i and i'

RY = Maximum amount of activity a yard crane can handle per time period

SH = 1 if sharing is allowed, 0 otherwise

SR_r = Number of slots in range r

TG = Threshold for maximum time gap between services assigned to adjacent yard ranges

W^k = Weight of activity concentration in the objective function

W^p = Weight of RMG contention in the objective function

W^q = Weight of cluster separation violation in the objective function

b = A set of ranges that are in the block b

c = A set of ranges, which when combined, will form cluster c

d_{irt}^{20} = Number of 20-footer spaces reserved for 20-footers in range r for service i at time t

d_{irt}^{40} = Number of 40-footer spaces reserved for 40-footers in range r for service i at time t

g = A set of ranges that belong to the group g

i = Index for service

j_{mr} = Binary indicator variable

k_{rt} = Number of excess activities in range r at time t, such that the k_{rt} contributed to some groups $g \in \mathcal{G}$ to violate MA_g

p_{bt} = Number of activities which need to wait for a yard crane, measured in number of activities above the capacity of the yard cranes in the block b at time t

q_{ic} = 1 if cluster c is assigned to service i, 0 otherwise

q_r = Number of time periods violating TG between services assigned to r and r^+

r = Index for range

r^+ = Range next to the range r within the same block with bigger slot number

r^- = Range next to the range r within the same block with smaller slot number

s_r^{20} = Number of 20-footer slots needed in range r

s_r^{40} = Number of 40-footer slots needed in range r

t = Index for time period

u = $(u_{ir})^T$

u_{ir} = 1 if service i is assigned to range r, 0 otherwise

v = $(x_{irt}^{20}, x_{irt}^{40}, y_{irt}^{20}, y_{irt}^{40}, p_{bt}, k_{rt}, j_{mr})^T$

w_r = 1 if range r has a 40-footer container, 0 otherwise

x_{irt}^{20} = Number of 20-footers received by service i into range r in time t

x_{irt}^{40} = Number of 40-footers received by service i into range r in time t

y_{irt}^{20} = Number of 20-footer retrieval activities by service i from range r in time t

y_{irt}^{40} = Number of 40-footer retrieval activities by service i from range r in time t

Chapter 1

Introduction

Most of the world's cargo is transported by sea today. The introduction of standard-sized steel containers of 20 or 40 feet long (called **20-footers** and **40-footers** respectively), has standardised transportation equipment such as trailer trucks, train wagons, marine vessel cells, and at the same time has also standardised lifting equipment such as quay cranes and yard cranes. This is a great invention that has greatly improved the efficiency of cargo transportation and hence reduction of logistic cost (Levinson 2006).

Containers transported from the shipper to the consignee pass through various parties. First, they are trucked from the shipper (e.g. a factory door) to a local sea port where they have to stay in the yard, waiting for the vessel to arrive. When the vessel arrives, they will be loaded (lifting of containers into the vessel). In most of the cases, this vessel is not the vessel that brings the containers to the destination. The containers are shipped to a hub port, where the containers are discharged (lifting of containers out of the vessel) and stacked in the yard. They will be loaded into another vessel at a later time. This process of transferring containers from one vessel to another vessel is termed as transshipment. The containers may be transshipped a few times at various hub ports until they are eventually discharged at the port of destination. Finally, the

containers will be trucked out of the port and delivered to the consignee (e.g. a warehouse). This results in a hub and spoke distribution network, which is very common in shipping as well as in other transportation systems. Vessels plying between hub ports are usually ocean liners (providing ocean voyages), and vessels plying between local ports and hub ports are feeder vessels (providing short sea services).

Ocean voyages and short sea services form the backbone of the world sea cargo transportation system. Infrastructures such as container ships and marine container terminals cost hundreds of million to a few billions of dollars to build and to operate. These assets generate revenue when cargo is moved from one port to another. Time spent by vessels alongside a terminal for **loading** and **discharging** of containers (i.e. moving containers into the vessel, and out of the vessel, respectively), while necessary, has to be as short as possible for the shipping line to maximise its revenue (Rudolft 2007). It is also a win-win situation between shipping lines and container terminal operators, where shipping lines reduce their operating cost, and more productive berths (more containers handled in the same period of time) translate to more revenue generated using the same infrastructure for terminal operators as well.

The problem has also become more challenging, as vessel new builds have grown in size (commonly measured in number of twenty-foot equivalent unit (**TEU**) containers) over the years. Rudolft (Rudolft 2007) summarises the representative sizes of vessels (and years) listed chronologically as follows: 750 TEU (1968), 1,500 TEU (1972), 3,000 TEU (1980), 4,500 TEU (1987), 7,900 TEU (1998) and finally 11,000 TEU (2006) being the Emma Maersk (Figure 1.1 shows a photo of Emma Maersk). Emma Maersk and her sisters Estelle Maersk and Eleonora Maersk that came along subsequently, have been

Figure 1.1: Emma Maersk

reported to carry up to 14,000 TEU. Then in 2011, Maersk again announces building 10 new triple-E class vessels that are capable of carrying 18,000 TEU. Wren (Wren 2007) normalises the size of the vessel to the length of the vessel, and reports that Emma Maersk carries 36.7 TEU per metre, while a usual panamax vessel (approximately 5,500 TEU) carries only 19.4 TEU per metre. This implies that terminal operators face greater pressure for the quay cranes to work at a faster rate so as not to increase the vessel port stay proportionally.

1.1 Improving quay crane work rate with better equipment

A quay crane is a lifting crane that lifts containers in and out of a vessel. While terminals aim to speed up the rate that the quay cranes can work, they are constrained at different fronts. The quay crane working rate is limited by the technical specification, as well as safety considerations. It is also constrained by the efficiency of the transportation system that brings the containers to and fro between the quay cranes and the stacking yard, and the efficiency of the stacking system in the stacking yard.

While quay cranes among various container terminals are largely similar in kind, various options for stacking systems and transportation systems are available in the market. The most common ones (used mostly in Asian ports) are the rail mounted gantry (RMG) or rubber tyre gantry (RTG). Straddle carriers are also used, and they are more commonly found in European ports. An example of the RTG and straddle carrier are shown in Figure 1.2 and Figure 1.3, respectively. Diesel engine trucks are the main work horse of the transportation system, while a small number of terminals use the automated guided vehicle (AGV) or other automated systems.

Engineering has progressed and increased the handling rate of the quay cranes to meet new standards of terminal performance required to minimise mega-vessel port stay. Crane lifting capability has evolved to carry more than one container at a time, such as twin lift (carry 2 $\times$ 20-footers length-wise), parallel lift (carry 2 $\times$ 20- or 2 $\times$ 40-footers side by side) and parallel twin lift (carry 4 $\times$ 20-footers) (ZPMC 2007). Recently, lifting 3 $\times$ 40-footers is also possible with a triple lift crane. Figure 1.4 shows a picture

Figure 1.2: Rubber tyre gantry (RTG) stacking crane

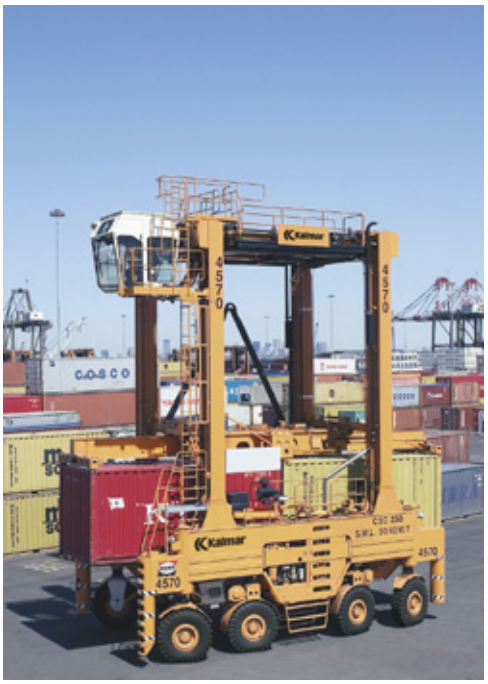


Figure 1.3: Straddle carrier stacking crane

of a twin lift quay crane, and figure 1.5 shows a picture of a triple lift quay crane.

However, having more efficient quay side handling may not reap immediate improve-

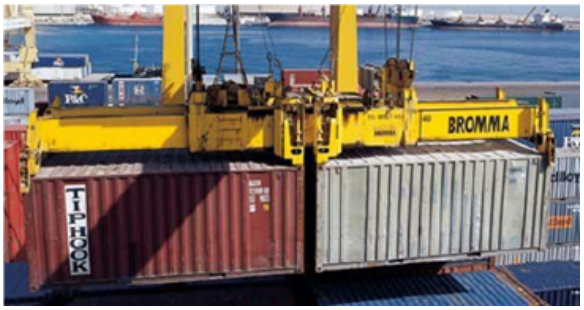


Figure 1.4: Twin lift quay crane, able to carry two 20-footers together

Figure 1.5: Triple lift quay crane, able to carry three 40-footers together

ment in performance. It passes the bottleneck from the quay side to the transportation system (trucks' efficiency) and storage and retrieval system in the stacking yard. For example, in a typical terminal that uses trucks as their transportation method, a discharging container needs to be lifted out of the vessel and **mounted** on a truck by a **quay crane**. The truck brings the container into the storage yard, where the **yard crane offloads** it and lands it onto a **stack** in the yard. A **loading** container will have to work in the reverse, i.e., the yard crane mounts the container on the truck, the truck brings it to the wharf and finally, the quay crane offloads it from the truck and loads it into

the vessel. Double stack trailers, multi-trailer-train and automated guided vehicle (AGV) systems have been engineered to increase trucking efficiency. Storage and retrieval efficiency then becomes the bottleneck when trucks waste time waiting in line in a congested yard, or wait to be served by yard cranes when the yard cranes are not available. These waiting times can be minimised if the decision of where and when to store and retrieve the containers in the yard are made with these considerations, and this is the role of **yard planning**.

1.2 Definition of yard planning

We define a **storage activity** as the activity triggered by the event of a truck arriving with a container on its trailer at a given time and at a given yard location, requiring a yard crane to offload the container from the truck and stack it onto the yard stack. A **retrieval activity**, on the other hand, is the activity triggered by the event of a truck arriving without a container at a given time and at a given yard location, requiring a yard crane to lift a specific container from the stack and mount it onto the trailer. In most terminals in the world, when containers arrive for storage, they remain in the same stack throughout its stay in the terminal, and they are retrieved only when they to be loaded onto a vessel. This is because shifting containers around incurs extra cost borne by the terminal operator. Hence, the location chosen during storage will be the location for the eventual retrieval. Next, the time at which the container arrives into the terminal and departs from the terminal is given by the shipping lines and the shipping agents, according to their vessels' schedule, and hence they are fixed. Therefore, upon arrival

of the container, the decision to place it in a location in the yard not only fixes a storage activity at this location at this given time, it also implies that there will be a retrieval activity at this location a later known time. This single decision will affect the efficiency of both the storage and retrieval activities. **Yard planning** is then defined as the decision of where to stack the containers in the yard when they arrive, while considering both the efficiency of storage and retrieval activities.

When too many storage or retrieval activities (collectively we call them **activities**) happen in near proximity of time and space, it causes congestion and the trucks have to stay in a queue, waiting to be served by the yard crane. Hence, within the vicinity, the number of activities should not exceed a given threshold (usually pegged to the handling capacity of one yard crane). When this violation of the threshold happens, we say that there is **activity concentration**. In addition, within a yard block with a given number of yard cranes, the total number of activities should not exceed the total working capacity of the cranes. If there is a violation, we say that **yard crane contention** has occurred. A violation of either case causes trucks to wait, and hence slows down the rate at which the trucks return to the quay cranes, resulting in reduction of quay side performance. Good yard planning is the key to reduce these violations.

It is also important to note that yard planning works under a complex set of constraints (such as physical constraints, policies due to regulations and safety practices), and some level of uncertainty for future activities (such as vessels not arriving on time due to bad weather). Hence, deciding the optimum stacking location of each container taking into consideration the constraints and uncertainty is too complex to be made manually, and most terminals resolve this by having yard plan strategies. A **yard plan**

strategy is a set of rules that constrains the choice of yard locations where each container may be stacked. The rules encompass the rules of thumb, based on human experience, such as no mixing of containers with different ports of destination in the same stack. These rules reduce the search space and hence reduce the complexity of the manual planning process. For example in **consolidated strategy**, containers to be loaded into the same vessels have to be stacked in groups in the yard. There are various strategies that terminals can adopt, and different terminals may find different strategies work better for their specific scenarios. We will cover more about strategies in Chapter 3.

1.3 Definition of yard plan template

Shipping lines operate most of their vessels by **services**, which have fixed schedules. A service visits a pre-determined set of ports called the ports of rotation. For example, APL operates the West Asia Express service (WAX), having the following ports of rotation: Ningbo - Busan - Kwangyang - Qingdao - Ningbo - Singapore - Jebel Ali - Dammam - Bahrain - Singapore - Ningbo. A few vessels will ply each service concurrently, such that the service will call at a port periodically (regardless of physical vessels) and usually on a fixed day of the week. This is an important characteristic that enables container terminals to generate yard plans that cycle on a weekly basis. Container terminals prefer to have repeating plans as it results in consistent performance week after week. Efforts to improve the yard plan that may result in better performance are also more persistent. In this thesis, we have assumed that all services call on a weekly ba-

sis, and hence the yard plan assumes a seven-day window that wraps around back to day 1 after day 7. We call this the weekly **yard plan template** – a plan that repeats on a weekly basis.

1.4 Focus and outline of thesis

We focus our thesis to the problem of finding the **optimum yard plan template** – a yard plan template with activity concentration and yard crane contention violations minimised. In the literature, we find very few research works are done in the area of yard plan template. The paper by Lee et al. (Lee, et al. 2006) is the only paper that we come across that addresses the yard plan template problem, other than the other paper authored by us (Ku, et al. 2010). In particular, we are interested in addressing the topic in three areas. First, we will like to define a generic model that defines the yard plan template problem for a wide variety of yard plan strategies. Secondly, we will like to solve this problem in a reasonable amount of time, assuming the inputs are deterministic. Third, noting that uncertainty prevails in reality, we will like to have a solution approach that can change the solution easily, while maintaining optimality, after the uncertainty is revealed. We call this the nimble yard plan template problem.

The outline of the thesis is as follows. We first give a literature review of related research papers on container terminal in Chapter 2. In Chapter 3, we present the motivation of this thesis. We then provide the problem scope and the assumptions made, followed by the problem definition and a mathematical formulation of the problem. In Chapter 4, we assume that the inputs are deterministic and present two solution ap-

proaches for the cases where terminals adopts the consolidated with dedicated strategy and consolidated with non-dedicated strategy. Experimental results of the run times are also presented. Next, in Chapter 5, we assume that the vessel arrival schedule is uncertain, and present the nimble yard plan template problem. The problem leads us to first discuss a generic nimble optimisation formulation. A short literature survey is presented and we also prove that the nimble optimisation is a generalisation of some of the works reviewed. Applying the formulation to yard plan template, three variations of the nimble yard plan template problem are presented. We propose heuristic approaches to solve the first two cases. Experiments are conducted and the quality of the nimble yard plan template are compared. The third case is a generic nimble yard plan template problem, and we provide some intuition to the complexity of the problem. Lastly, we conclude the thesis in Chapter 6 with a summary of the thesis contribution, as well as a short discussion of future works.

Chapter 2

Literature Survey

Container terminal operations are in the heart of all economic activities - no global trade takes place when goods cannot be transported. Hence, it has inevitably attracted a huge number of research works. Within container terminal operations, there are also many sub-problems that researchers work on, and we group them logically in the following few sections. As the thesis focuses on the yard planning problem, we start the review from this area.

2.1 Yard planning

Yard planning is the decision of where to stack the containers in the yard when they arrive. While there are many approaches to study this problem, one could look at the strategic level, where an optimum strategy needs to be found. The quality of the strategy is usually evaluated using a simulation model where a plan is generated according to the strategy, and trucks are simulated to store and retrieve containers from the yard. The waiting time of the trucks are collected as a measure of the quality of the plan. Some papers also quantify the quality of the solution analytically by computing some abstract

measures that are proxy of the eventual trucks waiting. Some papers also look at the problem at the operational level, where the exact location of the individual containers has to be decided. At this level, details of containers' exact location are considered, and the number of shuffling moves is to be minimised when the retrieved container is not the top most container in the stack.

Let us first review some papers that study the yard planning problem at the strategic level. Saanen and Dekker (Saanen & Dekker 2007a) give a good categorisation of yard plan strategies into four main characteristics, namely, dedicated versus non-dedicated, consolidated versus dispersed, housekeeping versus immediate final grounding, and discharge-optimised grounding versus loading-optimised grounding. A dedicated strategy means that a yard "location" (a unit of stacking space for containers) cannot be shared by more than two vessels, whereas a non-dedicated strategy allows sharing. A consolidated strategy means that all the containers to be loaded into the same vessel are stacked into a few groups in the yard, so that when the vessel is loading, the containers to be retrieved are already congregated into a few yard locations. A disperse strategy means that these containers should not be congregated and hence are to be dispersed throughout the whole yard. Immediate final grounding strategy means that once a container is placed in the yard, it will not be shifted around, and the next time it is retrieved is when it is loaded into a vessel. Housekeeping strategy, on the other hand, means that the containers can be shifted around the yard. Finally, discharge-optimised grounding strategy means that the strategy tries to minimise the waiting time of the trucks that are performing storage activities, at the expense of retrieval activities, while loading-optimised grounding tries to minimise the waiting time of the trucks that

are performing retrieval activities at the expense of storage activities. Note that there can be strategies that are neutral in this aspect, as they minimise the overall waiting time of trucks, regardless of its operation.

We review some works and at the same time categorise them based on the above terminology. Chung et al. (Chung, et al. 1988) compare the strategies of housekeeping versus immediate final grounding. They implement the housekeeping strategy with the use of buffer space. Simulation results show that the system with buffer could reduce the number of shuffling moves significantly and hence reduce the total container loading time. However, these are at the expense of having more yard cranes and lower utilisation of yard space. Taleb-Ibrahimi et al. (Taleb-Ibrahimi, et al. 1993) also discuss the trade-off between the immediate final grounding strategy and the housekeeping strategy, and a hybrid approach is also presented. Chen (Chen 1999) outlines the foundation of yard management, and particularly touches on the issue of housekeeping strategy. They focus on the trade-off between having more housekeeping and higher yard space utilisation. However, they do not study the efficiency of storage and retrieval in a housekeeping strategy. There are no other works that study the housekeeping strategy, as this strategy obviously is more costly to the terminals.

Next, Bruzzone and Signorile (Bruzzone & Signorile 1998) use simulation and Genetic Algorithm to find the best cluster layout. Yard plans with clusters are categorised as a consolidated strategy. Chen et al. (Chen, et al. 2000) study the storage space allocation problem with a time-space network. The allocations of containers to yard locations are made in advance. They assume no sharing of yard space and hence a dedicated strategy. Their objective is to re-use the same yard space for different ves-

sel over different time periods. However, minimising truck waiting time is not in their objective. Zhang et al. (Zhang, et al. 2003) study the dispersed strategy with block assignment. They solve the problem in two stages. In the first stage, they find an allocation of number of containers to yard blocks, so that workloads among yard blocks are balanced. In the second stage, using the solution from stage 1, they allocate the exact containers to the blocks, minimising the total distance travelled by the trucks. Murty et al. (Murty, et al. 2005b) use a fill-ratio heuristic for the import containers. This is a non-dedicated and dispersed strategy. Fill-ratio is a measure of the utilisation of each yard block. The heuristic is based on the observation that a block that has high fill-ratio usually has a high probability that a truck will arrive for retrieval, compared to a block with low fill-ratio. Hence a storage activity should try to avoid blocks that have high fill-ratio and hence avoid potential congestion with trucks performing retrieval. Petering and Murty (Petering & Murty 2006) compare the difference between consolidated strategy versus dispersed strategy. They approximate dispersed strategy by having very tiny clusters spreading over many locations, versus consolidated strategy where big clusters are spread over fewer locations. The simulation results show that disperse strategy is better than consolidated in terms of trucks' waiting time. Lee et al. (Lee et al. 2006) study the fixed-size yard range consignment strategy. This is a consolidated and dedicated strategy. Mixed integer programming and heuristic approaches are proposed. While most papers are neutral to loading- or discharge-optimised grounding, their approach is clearly a loading-optimised grounding strategy. Saanen and Dekker (Saanen & Dekker 2007b) use simulation to compare between a traditional stacking strategy versus a random stacking strategy, i.e. consolidated versus dispersed strategy. Their

objective is to increase operational yard density from 65% to 85% without compromising on trucks' productivity. The traditional stacking is known to have lower truck productivity when yard density increases beyond 65%, due to congestion of trucks. They show by simulation that random stacking is able to increase operational yard density without affecting productivity. They claim that this strategy is adopted by the Port of Rotterdam and Hamburg. Ku et al. (Ku et al. 2010) study the yard plan template problem where the yard plan repeats on a weekly basis. They propose a generic specification of the problem for the purpose of having a computer based search engine to find the best strategy. They propose solving a special case by solving the mathematical programme with CPLEX. The run time can be very long, and in some cases not able to solve within the time-out limit.

These studies are based on deterministic modelling of the problem, and do not directly address the issue of uncertainty. However, the uncertainties are not ignored totally, as most studies evaluate the solution using simulation.

Early works on modelling with uncertainty in port operations centre around uncertainty in the arrival of import and export containers, which leads to wasted shuffling during retrieval, or trucks waiting during retrieval. In the literature, shuffling moves are also commonly referred to as re-handling moves. Castilho and Daganzo (de Castilho & Daganzo 1993) study the amount of shuffling moves required when the retrieved container is not the top most container in the stack. This happens in the case of collection by trucks for local import delivery, as their arrivals are usually random. Kim et al. (Kim, et al. 2000) study the problem of deciding which slot of a yard block to place an export container. Their objective is to minimise the expected shuffling moves arise due

to a stack having lighter containers on top of heavier ones. The weights of the arrived containers are assumed to be random with probability estimated from historical information. The problem is solved with a dynamic programming approach where they model the state of a yard block as a function of the locations of empty slots and the weight of the heaviest container in a stack. Kang et al. (Kang, et al. 2006) also look into the stacking problem of having uncertainty in the weight information. Simulated annealing is used to find a good stacking strategy. Casey and Kozan (Casey & Kozan 2006) also study the problem of stacking the containers to minimise shuffling moves. An algorithm based on a meta-heuristic and simulation technique is proposed. Xu et al. (Xu, et al. 2010) consider a robust optimisation model for determining storage locations for import containers and yard crane movements. Import containers are stored in a yard area where internal trucks deposit the containers, while external trucks pick up containers. The time of container pick up is random, and the objective is to minimise the total waiting time by the trucks. Other papers that study approaches to minimise re-handling or shuffling within the yard block can be found in (Kim 1997, Kim & Bae 1998, Kim & Kim 1999a, Kim & Park 2003, Dekker, et al. 2006).

2.2 Yard crane and truck, routing and scheduling

Next, the following papers address the area of yard crane routing and scheduling. Kim and Kim (Kim & Kim 1997) and Narasimhan and Palekar (Narasimhan & Palekar 2002) study the yard crane routing problem within a yard block, such that the make span is minimised. Cheung et al. (Cheung, et al. 2002) and Linn and Zhang (Linn & Zhang

2003) study the problem of scheduling the yard crane movement from one yard block to another yard block, in order to satisfy the demand with the least delays. A mixed integer programming model and heuristic methods are presented. Kim et al. (Kim, et al. 2003) study the problem of minimising the waiting time of external trucks in the terminal, by sequencing retrieval and storage operations for the yard cranes. Ng (Ng 2005) studies the yard crane routing and scheduling problem with considerations of inter-crane interference among cranes in the same block. Recent works of Guo et al. (Guo, et al. 2008, Guo, et al. 2009) propose to simulate the yard crane gantry movements and container handling operations in order to arrive at the optimal dispatch sequence. To deal with uncertainty, their strategy is to have a very fast algorithm so that it could be recomputed when new information is available. Based on the new positioning technology such as Global Positioning System (GPS), they assume that within a given short planning window, advanced job arrival information can be determined accurately. They propose a modified exhaustive search algorithm over the space of possible sequences. For the case of a longer planning window which translates to a problem with bigger search space, they propose a hybrid A-star heuristic and recursive backtracking with prioritised search in order to accelerate the solution process. We do not find any other papers that model the yard crane routing or scheduling problem with uncertainty.

In the area of truck deployment and scheduling, Vis et al. (Vis, et al. 2001) study the problem of determining the minimum number of automated guided vehicles (AGV) required in a semi-automated terminal, where automatic stacking cranes (ASC) interface with the AGVs at the end of the yard blocks. A minimum flow algorithm is used. Li and Vairaktakakis (Li & Vairaktakakis 2004) study the vehicle scheduling problem to

minimise the make span of a single quay crane problem. Vis et al. (Vis, et al. 2005) study the problem of determining the minimum AGV fleet to fulfil the required job demand under time-window constraints. An integer linear programme (ILP) is modelled and they show that the ILP's results are close to their simulated results. Bish et al. (Bish, et al. 2007) study the vehicle despatching problem, and propose a greedy algorithm that performs close to optimum solution.

Straddle carriers are container equipment that function as yard cranes (with the ability to stack one container above another), and at the same time, they are transporters - bringing containers from one location to another in the terminal. Kim and Kim (Kim & Kim 1999b) study the routing problem of straddle carriers in the container terminal. The problem involves allocating of containers to straddle carriers and the straddle carrier routing problem. The objective is to minimise the total travel distance. Das and Spasovic (Das & Spasovic 2003) study the problem of scheduling the straddle carriers in order to service the external trucks. Their objective is to minimise the travelling time of the straddle carriers and waiting time of the external trucks. Wong and Kozan (Wong & Kozan 2006) propose an integrated approach to solve the problem of deciding where to stack the containers (yard planning) and scheduling of straddle carriers to minimise make span.

Meersmans and Wagelmans (Meersmans & Wagelmans 2001) study the integrated problem of scheduling all handling equipments in an automated container terminal that uses ASCs and AGVs. They prove NP-hard of the problem and propose a heuristic called Beam Search to solve the problem. All the above models on truck deployment and scheduling do not deal with uncertainty.

2.3 Berth planning

Berth planning is the decision of where to berth each vessel along the length of the wharf. It is common for the berth planning problem to be modelled as a continuous 2-D rectangle packing problem with side constraints, where the physical length of the wharf and vessel form one of the dimensions, and time forms another dimension.

Uncertainty of vessel arrival schedule is commonly studied in conjunction with berth planning. Moorthy and Teo (Moorthy & Teo 2006) present one of the earliest papers that study the berth plan template with vessel arrival uncertainty. They propose a simulated annealing approach to search for an optimal berth plan template which is encoded into a sequence pair. The objective value is evaluated using a probabilistic model, and the final solution is tested using Monte Carlo simulation with random arrival of vessels. Following which, Dai et al. (Dai, et al. 2007) apply a similar technique to solve the dynamic berth allocation problem which does not have time dimension wrapped around. Du et al. (Du, et al. 2010) then extend the solution method from (Moorthy & Teo 2006) to have a feedback mechanism where earlier iterations generate feedbacks to the model to adjust the time buffers for the future iterations. Hendriks et al. (Hendriks, et al. 2010) study the robust berth plan template problem with quay crane reservation. They assume that the number of quay cranes reserved for a vessel is a function of the punctuality of the vessel's arrival, and further assume that the number of quay cranes reserved is proportional to the cost of operating the berth. They propose a mixed integer linear programme to find a robust berth plan that minimises the crane reservation. Han et al. (Han, et al. 2010) consider simultaneously the vessel to berth assignment and quay crane sequencing problems and develop a stochastic mixed integer program. They then

solve it with a genetic algorithm (GA) in which the evaluation of the fitness function is done by sampling on the stochastic parameters.

There are also some earlier works that did not consider uncertainty in vessel arrival. Tong et al. (Tong, et al. 1999) study the deterministic berth allocation problem, and apply the Ant Colony Optimisation technique to solve it. Kim and Moon (Kim & Moon 2003) study the deterministic berth scheduling problem using simulated annealing.

2.4 Quay crane sequencing and stowage planning

Quay crane sequence refers to the exact order in which the quay crane will perform the load and discharge of the ship. In cases where multiple quay cranes work on the same vessel, quay crane sequencing also produces the job allocation to the quay cranes, commonly termed as crane split. The objective is usually to reduce make span. Stowage planning refers to the decision of assigning loading containers to the cells in the vessel, conforming to the stowage instruction (constraints) provided by the shipping line.

Wilson and Roach (Wilson & Roach 2000) study the problem of finding the stowage plans for a vessel at each port of call for its entire voyage. The later stowage plans are dependent on earlier ones with respect to the order in which the ports are visited, and the objective is to optimise efficiency at each port of call. Steenken et al. (Steenken, et al. 2001) study the problem of stowage planning and scheduling of straddle carrier movements for load and discharge operations such that make span is minimised. They also propose that terminals should consider real time stowage planning due to the un-

certainty of the given information. Ambrosino et al. (Ambrosino, et al. 2004) study the problem of finding the optimum stowage plan with additional hydrostatic and structural constraints of the vessels.

Gambardella et al. (Gambardella, et al. 2001) study the problem of resource allocation and scheduling of load and discharge operations. They propose to solve the problem in two phases, namely, performing the resource allocation first, and then finding the exact sequence of loading and discharging of each quay crane. Lim et al. (Lim, et al. 2002) prove the NP completeness of the quay crane scheduling problem and apply the tabu search technique to solve it. Bish (Bish 2003) studies the problem of scheduling the quay cranes and internal trucks simultaneously, with the objective of minimising the make span. The model is constrained by the choice of discharge locations, which is a pre-processed set. Kim and Park (Kim & Park 2004) study the problem of quay crane sequencing, with the objective to minimise the make span. They use the branch and bound algorithm for the optimum solution and a heuristic called greedy randomised adaptive search procedure (GRASP). Choo (Choo 2006a) study the multiple quay crane sequencing problem with quay crane clearance constraints and yard congestion considerations. They propose a mixed integer programme with Lagrangian relaxation framework. In a later work, Choo (Choo 2006b) model the same problem with a multiple self-interested agents framework. Each quay crane is modelled as an agent and they compete among themselves for limited resources in order to minimise their individual make span. All the models on quay crane sequencing and stowage planning are deterministic.

2.5 Container terminal simulation

Container terminal operations are so complex that they cannot be modelled as a single analytical model. Furthermore, the uncertainty of the input parameters due to real life dynamics, such as demand variability, varying equipment working speed, weather, labour issues, equipment breakdowns, etc, prevails. Hence, simulation has been used very widely in both academia and container terminal industry as a tool to evaluate solutions (Gambardella & Rizzoli 2000). Other than some of the works that are already mentioned earlier, the following works are papers that focus more on the application of simulation on container terminal operation.

Ottjes et al. (Ottjes, et al. 1994, Ottjes, et al. 1996) use the software MUST to build their model of Rotterdam port area. They model the transportation of the containers between the various terminals which they call inter-terminal transport (ITT). In (Ottjes et al. 1994), they simulate a scenario of having a floating terminal that moves around different terminals in Rotterdam to exchange containers between the terminals. On the other hand, in (Ottjes et al. 1996), they evaluate two types of equipment for the ITT using simulation, namely AGVs and Multi-trailer-train system, which is a manned traction unit pulling a train of trailers on which containers with a maximum of 10 20-footers can be carried. Bontempi et al. (Bontempi, et al. 1997) simulate the La Spezia container terminal where ship operations, gate operations and train operations are modelled. They use the model to evaluate various optimisation techniques applied to terminal operations in La Spezia. In a related work, Gambardella et al. (Gambardella, et al. 1998) focus on the resource allocation problem and use the simulation model to evaluate the proposed integer linear programme solution. A later work by the same group, Rizzoli

et al. (Rizzoli, et al. 1999) use a software MODSIM III to build a model of La Spezia Container Terminal, which is likely to be an enhanced version of the model presented in the 1997 paper. The model is used to convince management of new policies derived from optimisation techniques. However, truck congestion is not simulated in detail in all three papers. Yun and Choi (Yun & Choi 1999) use SIMPLE++ software to build a model of the container terminal in Pusan, Korea, with ship operations and gate operations. They adopt an object oriented design, which is claimed to be more flexible than any other available simulation package like UNCTAD, PORTSIM and the MIT port simulator. Legato and Mazza (Legato & Mazza 2001) use a queuing network model to model the arrival, berthing and departure processes of vessels at a container terminal, and a simulation model is built to perform "what if" analysis of the berth planning problem. Jarjoui et al. (Jarjoui, et al. 2001) simulate a multi-agent cooperation system to achieve good terminal performance. They build a simulation called MASH (Multi-Agent System for Harbour activities) for this purpose. Itmi et al. (Itmi, et al. 2000, Itmi, et al. 2001) present their object oriented terminal simulation system SGTC. They experiment with a concept, called Noria, of sharing of straddle carriers between two vessels. Rida et al. (Rida, et al. 2003) develop a terminal simulator using object oriented modelling, using UML for design and JAVA for programming. It is meant for evaluation of different load and discharge operations, storage policies and resource allocation algorithms. Henessey et al. (Henessey, et al. 2004) present a berth allocation simulation system coded in JAVA. It is used to evaluate different berth allocation policies for Skandia Harbour , Norfolk International Terminal, and Seagirt Terminal in Baltimore. Guenther et al. (Guenther, et al. 2006) propose using simulation for evaluating a combination of

automated transportation modes, dispatching strategies, and stochastic variations.

Note that there are also some commercial players in the arena of container terminal simulation. However, their works may not be publicly published. For example, TBA Nederland (http://www.tba.nl/) provides container terminal consultancy services using simulation, and they also provide a simulation software (called CONTROLS) that can communicate directly with industrially used terminal operating systems (**TOS**) like NAVIS (http://www.navis.com/). Total Soft Bank (http://www.tsb.co.kr/) also provides a simulation software (called Operating - CATOS Port Simulator (O-CPS)), that communicates with CATOS (a TOS). A Singapore registered company SIMPLUS (http://www.simplus.sg/) also provides simulation studies on container terminals for Maritime Port Authority of Singapore and PSA Singapore.

2.6 General

Some general papers related to container terminals are also reviewed. At the strategic planning level, terminals are concerned with their handling capacity and their business cost models so that they can maximise their revenue and profits. Many of these are modelled with simulation. Kulick and Sawyer (Kulick & Sawyer 1999) propose a simulation model to analyse container terminal handling capacity. They call the system SIMCAP, (simulation-based capacity analysis platform). Benacchio et al. (Benacchio, et al. 2001) study the economic implication of dedicated terminals. Kim and Kim (Kim & Kim 2002) create a cost model for evaluating the optimal yard configuration, consisting of the amount of storage space and number of yard crane required. Kia et al.

(Kia, et al. 2002) study using simulation the impact of terminal capacity using two approaches, namely, stacking all containers in the yard and transporting them out later, versus transporting many of them out of the terminal as soon as possible. Shabayek and Yeung (Shabayek & Yeung 2002) use software WITNESS to build the simulation model for the marine operations in the channels and the anchorage at the Kwai Chung terminals. They study how the capacity of the four terminals (HIT, HIT-COSCO, SLOT and MTL) can be limited by the marine operations. Fung et al. (Fung, et al. 2003) study the impact of terminal handling charges on overall shipping charges. Huang and Chu (Huang & Chu 2004) study the cost of operating a container terminal with varying choices of yard handling system, such as rubber tyre gantry cranes or rail mounted gantry cranes. Part of a US$238 million project to expand Alabama State Docks in Mobile Alabama, Schroer et al. (Schroer, et al. 2004) simulate the handling capacity of the newly expanded terminal with various demand scenarios. Hsu and Hsieh (Hsu & Hsieh 2005) analyse the cost of operating an ultra large container ship and find that one of the factors among many that make ultra large container ships economical is port efficiency.

Some papers discuss the issue of manpower deployment in a terminal. Kim et al. (Kim, et al. 2004) study the problem of human operator scheduling using a constraint satisfaction technique. Legato and Monaco (Legato & Monaco 2004) study the problem of generating a shift roster pattern, and then short term planning methods to handle day-to-day deployment issues. A heuristic for set covering problem is used to generate the roster pattern, while a branch and bound algorithm is used for the day-to-day planning.

Other more general papers are as follows. Rebollo et al. (Rebollo, et al. 2000)

propose a multi-agent architecture for a real time container terminal control system as it is computationally too complex for a integrated solution. Tongzon (Tongzon 2001) uses data envelopment analysis to study the efficiency of ports as there are many indicators that can measure the efficiency of ports and they are non-parametric. Murty et al. (Murty, et al. 2005a, Murty et al. 2005b) outline the decisions made by a container terminal on a daily basis and propose that a decision support system is needed for such complicated decision making. Also, general review papers on container terminal research are in (Meersmans & Dekker 2001, Steenken, et al. 2004, Vis & de Koster 2003, Chen 2003, Stahlbock & Voβ 2008).

Chapter 3

Motivation and Problem Definition

The earlier chapter on literature survey provides us with the insight on what has been done, and it motivates us to study the yard plan template problem, where few works have been done. The outline of this chapter is as follows. In Section 3.1, we present the motivation of our work, by observing what is presented in the literature, and what might be needed in the container terminal industry; we list down three areas of focus in the thesis addressing the needs and gaps of the container terminal industry. In Section 3.2, we set up the scope of the problem assumed in this thesis, and list the assumptions made. Then in Section 3.3, we address the first area of focus, by presenting a generic yard plan template problem formulation. We summarise the chapter in Section 3.4

3.1 Motivation

From the literature survey in the earlier chapter, we see that there are many papers that study the various aspects of yard planning. They have also covered various yard plan strategies, and make comparisons between them. Only one paper by Lee et. al. (Lee et al. 2006) addresses the yard plan template problem. However, they examine only

the consolidated and dedicated yard plan strategy. From all the papers reviewed, and to the best of our knowledge, we do not come across any work that presents a generic model of the yard plan template problem that caters for a variety of yard plan strategies. We also do not come across any work that solves the yard plan template problem with non-deterministic inputs. We note that uncertainty prevails in real life, and container terminals are adjusting their plans when input changes. Hence, the motivations of the thesis to focus on the yard plan template problem in the following three areas are as follows.

1. Different terminals may find that different yard plan strategies work better for their terminals with their unique combinations of services, yard layouts and equipment configurations. The yard plan template can be used as a means to compare different strategies. We find that papers that study yard plan strategies make different assumptions of the problem and hence model their problem differently. It makes comparison of results across papers difficult. To help terminals to discover the best strategy for their scenario, we will require the problem to be defined generically so that solutions that apply different strategies can be generated and compared. Hence, we first focus on formulating a generic model that defines the yard plan template problem that is applicable to a wide variety of yard plan strategies.

2. Secondly, search engines can be designed to discover the best yard plan strategy, and that requires the generic yard plan template problem to be solved in a reasonable amount of time. As the generic problem differs in complexity when applied to different strategies, we propose to approach the problem by having different solu-

tion techniques for different strategies. As the scope of covering all strategies can be too large for the thesis, we only address two strategies in this thesis, namely, consolidated and dedicated strategy, and consolidated and non-dedicated strategy.

3. Lastly, after a terminal has adopted a strategy, a yard plan template can then be generated and implemented. However, uncertainty of events prevails in reality, and terminals usually modify and adapt their yard plan template to the changes, which may not be optimum. While the community for robust optimisation may suggest developing a robust plan – a plan that remains feasible when subjected to uncertainty, we will like to have a plan that can be changed (recover) easily to an optimal solution after the uncertainty is revealed. This is because robust solutions are too conservative to remain feasible after uncertainty is revealed. On the other hand, if we allow changes without any cost to be made after the uncertainty is revealed, we obtain an optimum solution of the realised scenario. However, changes are usually not free, and hence the changes have to be "easy", and we assume the degree of difficulty of the proposed change can be measured by a cost function. This will mean that we have to have a starting solution that is "nimble" such that future changes are made "easy". Hence, we call this the nimble yard plan template problem. While many types of inputs to the problem can be uncertain, we learn from a global terminal operator that changes to the vessels arrival schedule have the greatest impact to operation efficiency for a terminal that adopts the consolidated strategy. Hence we focus on the nimble yard plan template with uncertainty in vessel arrival schedule. We explore various options

of recovery policy and compare the quality of the solutions.

3.2 Scope and assumptions

In this section, we provide the scope of our problem and assumptions made in the thesis. We first present the yard configuration, which includes yard layouts, yard equipments used, and the size of the terminal. Then we present the list of assumptions that will apply throughout this thesis.

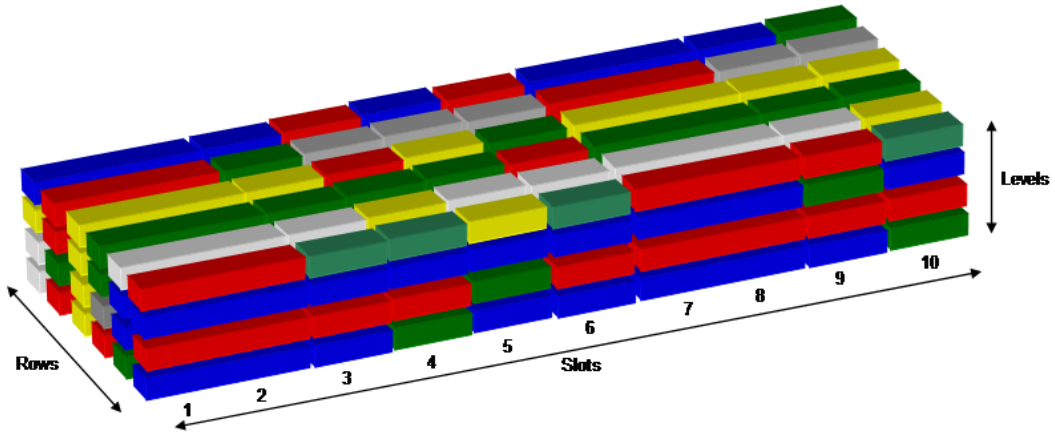


Figure 3.1: Perspective view of a yard block, showing how containers are stacked in the block and notations of slot, row and level

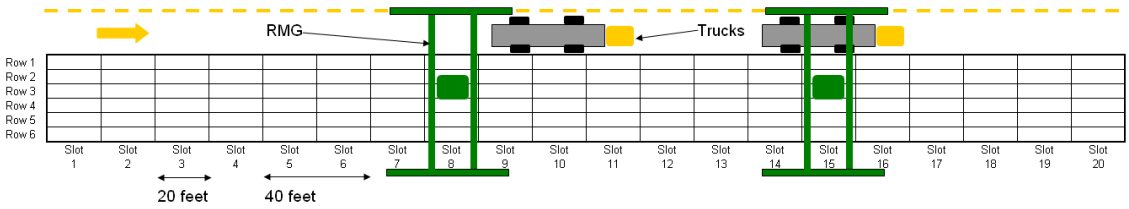


Figure 3.2: Plan view of yard block, illustrating trucks being served by yard cranes along the length of the block

3.2.1 Container yard configuration

In this thesis, we consider a container terminal with automated rail mounted gantry (**RMG**) yard cranes fixed to yard blocks. We assume containers are stacked lengthwise along the length of the block as illustrated in Figure 3.1. In this example in Figure 3.1, the block has ten slots, six rows and four levels of containers. Each slot has a length of 20 feet, capable of stacking 20-footers. 40-footers will occupy two consecutive slots (see Figure 3.1, slot 1 to 2, and slot 7 to 8 for illustration). For convenience, we call a slot that stacks 20-footers a "20-footer slot", and a pair of slots that stacks 40-footers a "40-footer slot". The slots that are used to stack 40-footers are not fixed, i.e. any pair of slots can be used. However, there should not be any mixing of 20- and 40-footers in the same slot. The RMG is able to move along the length of the yard block, and lift on or lift off containers from trucks that drive on the service lane along the length of the block (See Figure 3.2 for illustration).

The model we present in this thesis is not specific to any particular layout of where the yard blocks should be placed in the terminal. However, in all the experiments that are carried out in this thesis, we have assumed a medium sized container terminal that has the layout shown in Figure 3.3. This terminal has 20 yard blocks, each with 40 $\times$ 20-footer slots. Each yard block has three RMGs that are fixed to the yard blocks. As a reality check, we note that this layout is comparable to a medium size container terminals, such as Incheon Container Terminal in South Korea, Fuzhou Qingzhou Container Terminal in China and Asia Container Terminal in Hong Kong, as illustrated in Figure 3.4.

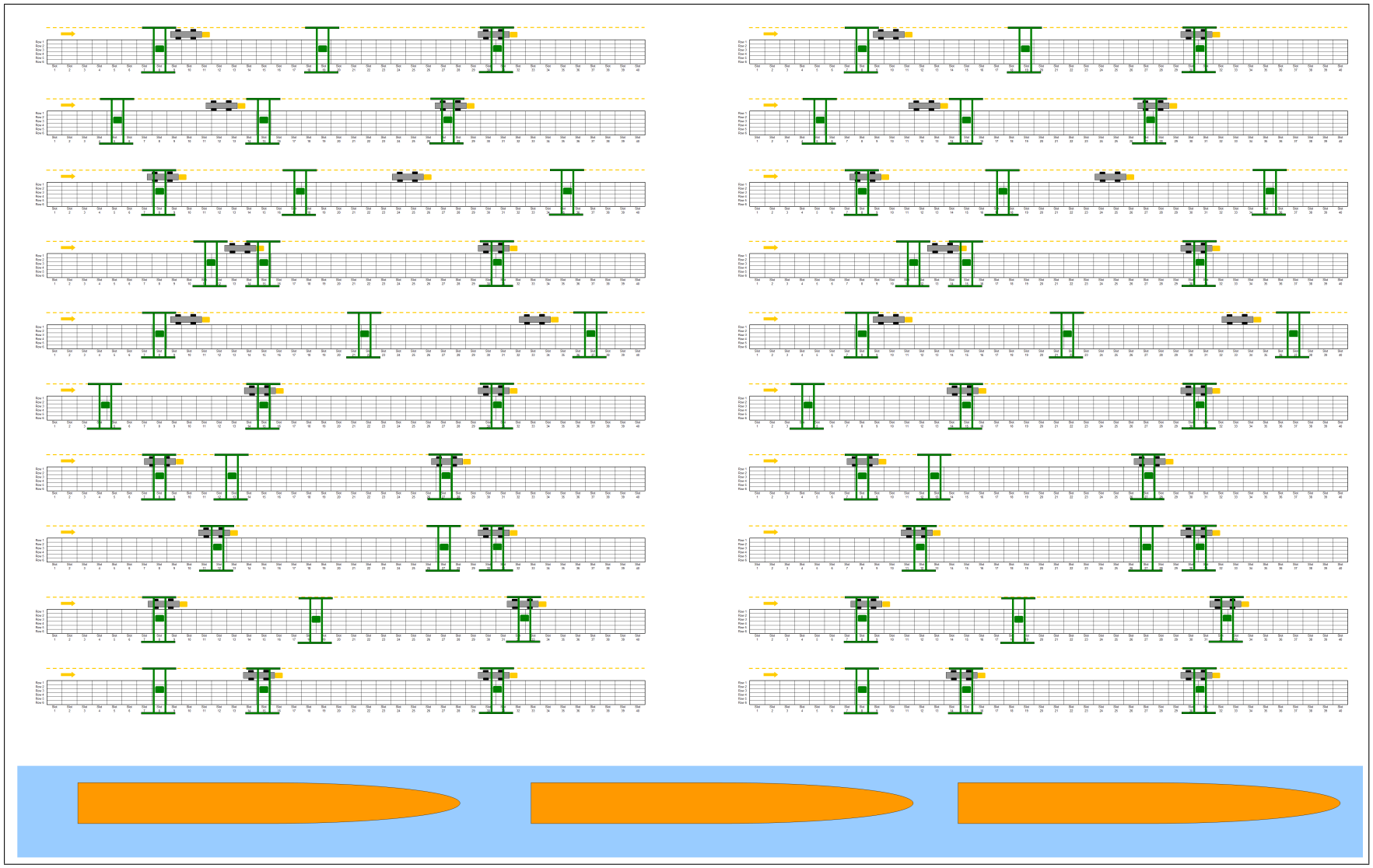
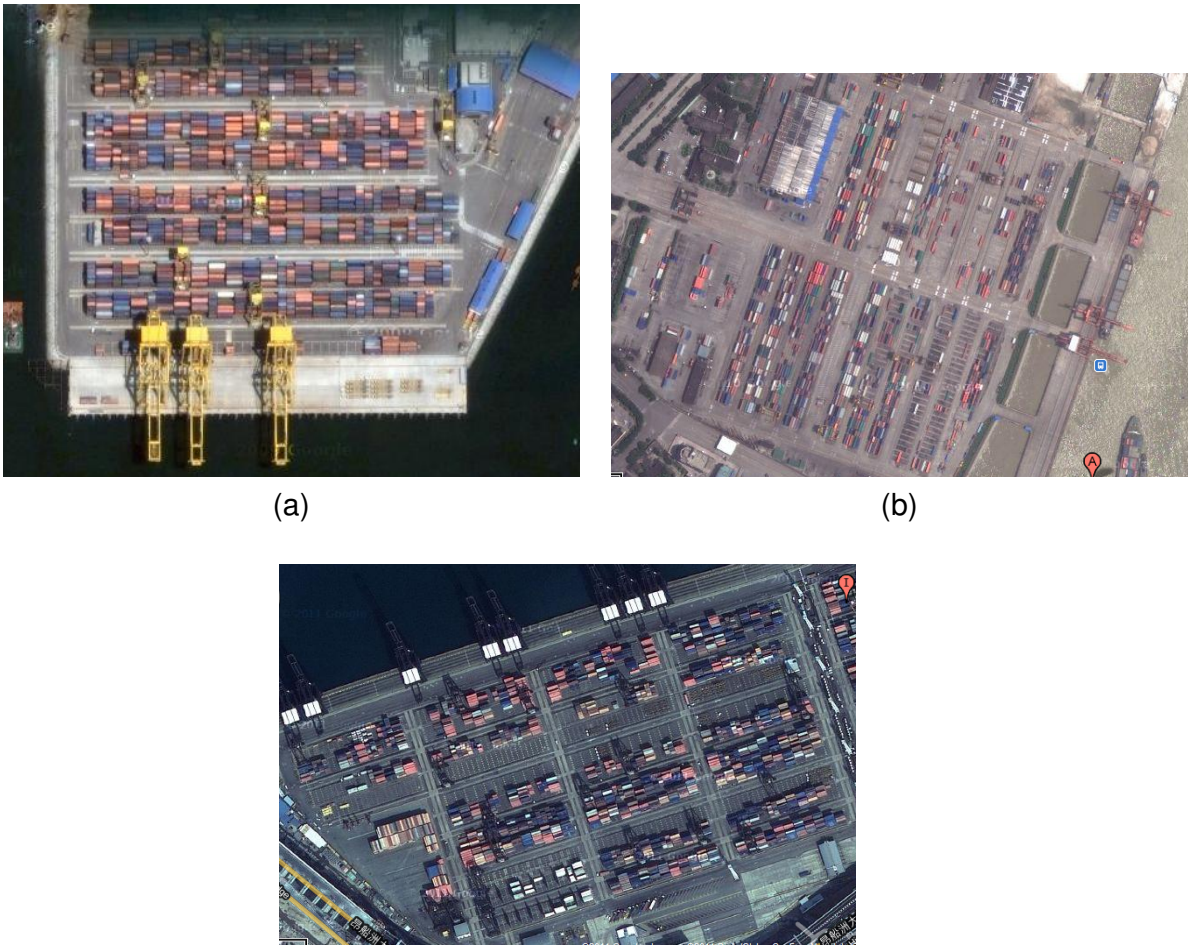


Figure 3.3: Yard layout used for the experiments - 20 blocks of 40 slots each, supporting three berths



(a) (b)

(c)

Figure 3.4: Example of yard layout comparable to the experimental layout. (a) Incheon Container Terminal in South Korea (b) Fuzhou Qingzhou Container Terminal in China (c) Asia Container Terminal in Hong Kong

3.2.2 Assumptions

We have made the following assumptions.

1. Storage and retrieval may come from ship discharge or loading at the quay, train discharge or loading from an inter-modal terminal, and trucks that bring containers by road into or out of the terminal via the container gate. As a generalisation, a

train can be viewed as a vessel where the time window to load the wagons and discharge the wagons is the port stay of the vessel. Similarly, in-gate can be viewed as a discharging vessel and out-gate as a loading vessel, and the opening hours of the gate is the port stay of the vessel. Without loss of generality, we assume that all activities in our problem come from vessel discharge and loading.

2. We assume that only vessels that belong to some services can call at the terminal. Each service calls at the terminal once a week, and the arrival schedule repeats on a weekly cycle.

3. We assume that a day has three eight-hour work-shifts, and the time resolution required for planning is in work-shifts. Therefore, a week is partitioned into 21 equal time periods.

4. We further assume that, for each service, the number of containers arriving in each time period prior to the loading of this service is known, and is repeating in a weekly cycle.

5. A container that arrives into the terminal will stay in the same yard stack throughout its stay in the terminal, until it is retrieved for loading into the vessel.

6. Containers to be loaded into a service will always arrive in the yard before the start of the loading period of the service. No containers to be loaded into a service will arrive during the loading period of the service.

7. No containers will stay longer than one week in the yard.

8. As we have assumed that the terminal uses RMG as yard cranes, the term **RMG contention** is then analogous to the term “yard crane contention” defined earlier.

9. Any violations to the activity concentration and RMG contentions collectively will be referred to as **violations**.

10. We assume that any violation can be resolved within the time period where it occurs, and there will be no carry forward of violations from one time period to another.

3.3 A generic yard plan template problem

In this section, we address the first area of focus for the thesis – a generic yard plan template problem definition. The yard plan template problem definition has to include both the scenario and the yard plan strategy, and it has to be generic to cater for different yard plan strategies. The generality of the definition relies heavily on a set of input parameters, which we refer to as the generic yard plan template specification. The various strategies are manifested in the generic yard plan template specification as a combination of a yard space partition, grouping and zoning of the partitioned yard space, threshold values that are dependent on the partition and the grouping and zoning method, and a couple of flags. The details of the specification are given in Section 3.3.1, and the formal problem definition is given in Section 3.3.2. We have also illustrated with some examples how various strategies are represented using the generic specification in Section 3.3.3. The constraints binding the input parameters from the problem specification to the solution and objective function are best represented mathematically

using a mathematical programming approach. Section 3.3.4 presents a mixed integer programming (MIP) model for the problem.

3.3.1 Generic yard plan template specification

Generically, given a time window of any duration, we partition the time window into equal sized time periods according to the required time resolution. Based on our assumption, we partition a week into a set of 21 equal time periods, and we call this set $\mathcal{T}$. For simplicity, we let the set $\mathcal{T}$ to be $\{1, 2, \ldots, 21\}$. Note also that in a yard plan template where plan repeats on a weekly basis, the time window wraps around from time period 21 back to 1.

The generic yard plan template specification is denoted by two tuples $\mathcal{D}_\mathcal{T}$ and $\mathcal{S}_\mathcal{T}$, representing the two components of the specification, namely, **demand** and **supply**, with a given $\mathcal{T}$. The components of $\mathcal{D}_\mathcal{T}$ and $\mathcal{S}_\mathcal{T}$ are given as $\mathcal{D}_\mathcal{T} = (\mathcal{V}, DA^{20}_{it}, DA^{40}_{it}, \mathcal{L}_i)$ and $\mathcal{S}_\mathcal{T} = (\mathcal{R}, \mathcal{G}, \mathcal{B}, \mathcal{C}, CS_r, SR_r, MA_g, RY, MY_b, W^k, W^p, SH, OS)$. The details are described in the following two subsections.

Demand specification $\mathcal{D}_\mathcal{T}$

$\mathcal{V}$ = Set of services

DA^{20}_{it} = Number of discharge 20-footers received by service i at time t

DA^{40}_{it} = Number of discharge 40-footers received by service i at time t

$\mathcal{L}_i$ = Set of time periods where each element t is the time period when service i is loading its containers

The set $\mathcal{V}$ contains the set of services that calls at the terminal. Each service i calls at the terminal once a week. For each service i, the terminal will receive containers to be loaded into the service i into the yard, throughout the week. When the service calls the terminal for loading, all the containers for this service will be loaded into the service. We let DA_{it}^{20} and DA_{it}^{40} denote the number of **20-footers** and **40-footers**, received by service i at time period t into the yard, respectively. Hence the total number of 20-footers and 40-footers in the terminal for loading into service i are simply $\sum_{t\in\mathcal{T}} DA_{it}^{20}$ and $\sum_{t\in\mathcal{T}} DA_{it}^{40}$, respectively. We also let $\mathcal{L}_i$ denote the set of time periods, where $t \in \mathcal{L}_i$ implies that service i is loading its containers in time period t. As an illustration of how $\mathcal{L}_i$ and DA_{it} are defined, especially for the case of wrapping around at the end of the last time period, an example is given in Figure 3.5.

Supply specification $\mathcal{S}_\mathcal{T}$

$\mathcal{R}$ = Set of ranges

$\mathcal{G}$ = Set of groups where each element g is a set of ranges that belong to the group

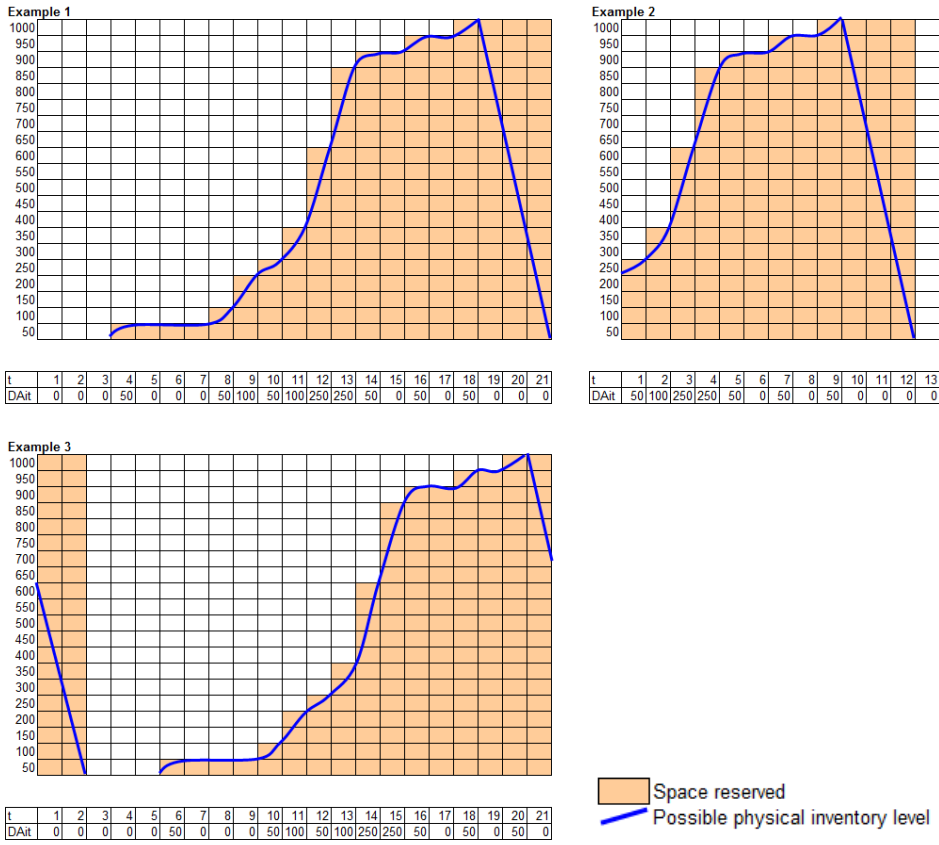
$\mathcal{B}$ = Set of blocks where each element b is a set of ranges that are in the block

$\mathcal{C}$ = Set of clusters where each element c is a set of ranges that can be combined to form a cluster

CS_r = Capacity in TEU for a slot in range r

SR_r = Number of slots in range r

MA_g = Maximum amount of activity allowable in group g in each time period



t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
DAit	0	0	0	50	0	0	0	50	100	50	100	250	250	50	0	50	0	50	0	0	0

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
DAit	50	100	250	250	50	0	50	0	50	0	0	0	0	0	0	50	0	0	0	50	100

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
DAit	0	0	0	0	0	50	0	0	0	50	100	50	100	250	250	50	0	50	0	50	0

Example 1: $\mathcal{L}_i = \{19, 20, 21\}$, $\mathcal{A}_i = \{\}$,

Example 2: $\mathcal{L}_i = \{10, 11, 12\}$, $\mathcal{A}_i = \{13, 14, 15, 16, 17, 18, 19, 20, 21\}$,

Example 3: $\mathcal{L}_i = \{1, 2, 21\}$, $\mathcal{A}_i = \{3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21\}$.

Figure 3.5: Example showing how $\mathcal{L}_i$s and $\mathcal{A}_i$s are derived.

RY = Maximum amount of activity a yard crane can handle per time period

MY_b = Number of RMG in block b

W^k = Weight of activity concentration in the objective function

W^p = Weight of RMG contention in the objective function

SH = 1 if sharing is allowed, 0 otherwise

OS = 1 if the yard plan template is in one-slot mode

The yard is made up of multiple yard blocks and each block is further subdivided into slots. Depending on the yard planning strategy being modelled, we decide on the space resolution (in number of slots) and we partition the yard into **ranges**, each made up of consecutive slots of the size of the decided resolution. We call the set of ranges $\mathcal{R}$.

We let SR_r denote the size (in slots) for range r. We allow SR_r to vary with r as the number of slots in some blocks may not be a multiple of the required resolutions, and hence some ranges are forced to be smaller or larger. The storage capacity of each slot is indicated as CS_r, which is the maximum TEU that a slot in range r may stack. We also allow CS_r to vary with r as terminals may have different types of yard block configurations in the same yard.

We let $\mathcal{G}$ denote the set of groups, where each group $g \in \mathcal{G}$ is a set of ranges that belong to the same group. Activities occurring in the ranges in the same group g are used in the computation of the activity concentration for the group g as follows. We define a threshold MA_g for each g, where MA_g is the maximum number of activities allowable in group g in each time period t. Then, the total number of activities in g at any time period t has to be less than MA_g, exceeding which will constitute to activity concentration violation. In order for the activities in a range r not to be double counted for violating thresholds in different groups, any surplus of activities in each range that constitute to this violation will be remembered by the range that causes it, and then the total violation is a sum over all the violations of the individual range (the mathematical expression will be shown in the later section). For illustration of how a yard block can

be partitioned into ranges and how the ranges can be grouped, Figure 3.6 shows an example of a yard block of 20 slots, partitioned into ranges at a resolution of five slots, and grouped into three groups, with two ranges in each group. Note that $\mathcal{G}$ is not a partition of the yard, as groups may overlap. It is also possible to model congestion due to interactions between activities in neighbouring blocks (as presented in a paper by Lee et al. (Lee et al. 2006)) by grouping ranges in neighbouring blocks together.

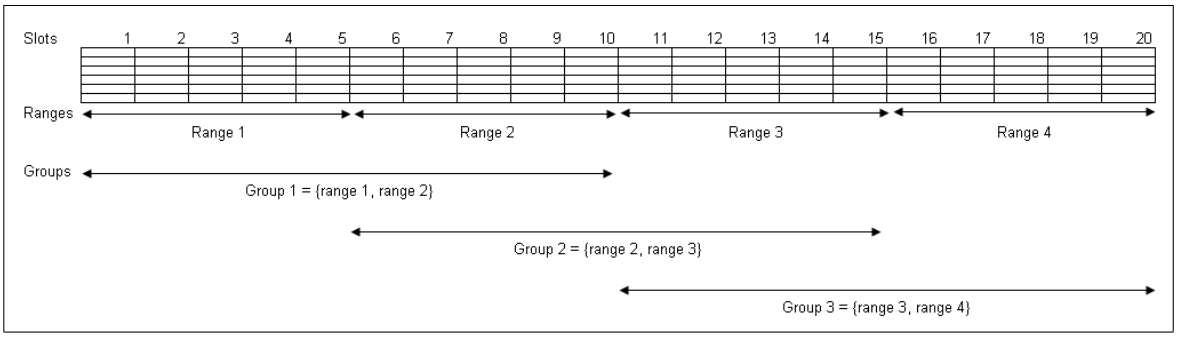


Figure 3.6: Example of a yard block with 20 slots, partitioned into four ranges of five slots each, and the ranges grouped into three groups of two ranges each.

We let $\mathcal{B}$ denote the set of blocks, where each block $b \in \mathcal{B}$ is a set of ranges that belong to the same block. The number of activities in each block in each time period should not exceed that of the RMG handling capacity, i.e. handling rate of a RMG multiplied by the number of RMGs in the block ($RY \times MY_b$). Any activity count in excess of this threshold will be added into the RMG contention violations.

W^k and W^p are the weights given to activity concentration and RMG contention violations in the objective function, respectively. However, in most practical situations, they should be equal as activity concentration and RMG violation are calculated in the same unit of measurement, i.e. number of activities in surplus of a given threshold.

$\mathcal{C}$, SH and OS are strategy-specific parameters. If the strategy requires multiple

consecutive ranges (called a **cluster**) to be assigned to the same service, then $\mathcal{C}$ is a set of all possible legitimate clusters, where each element $c \in \mathcal{C}$ is a set of consecutive ranges that will be grouped to form the cluster. If the strategy does not require cluster assignment, then $\mathcal{C}$ is an empty set. SH is a flag, which when set to 1 indicates that the strategy allows sharing, i.e. two or more services can be assigned to the same yard range, while when set to 0 indicates that we do not allow sharing, i.e. each yard range should be assigned to only one service. OS is another flag, which when set to 1 indicates that the yard plan template assumes a one-slot mode. In a one-slot mode, the yard must be partitioned into $\mathcal{R}$ such that the size of each r is only one 20-footer slot. Then, a 40-footer will occupy two consecutive ranges r and r^+, where r^+ denotes the ranges that is adjacent to r in the same yard block and having slot number bigger than r. Hence r^+ does not exist when r is the last range in the block. Note that for the case when we are not in the one-slot mode, i.e. OS is set to zero, containers will not be allowed to straddle across two ranges r and r^+.

3.3.2 Problem definition statement

We now give a formal definition of the problem.

PROBLEM DEFINITION 1: *Find the matrix X where each element x_{irt} defines the number of discharging containers received by service i at time t to range r, satisfying the specification given in $(\mathcal{D}_\mathcal{T}, \mathcal{S}_\mathcal{T})$, such that we minimise the weighted sum of*

1. *$\sum_{r\in\mathcal{R}} \sum_{t\in\mathcal{T}} k_{rt}$, where k_{rt} denotes the surplus in number of activities in range $r \in \mathcal{R}$ at time $t \in \mathcal{T}$, such that the k_{rt} contributed to some groups $g \in \mathcal{G}$ to violate*

MA_g. *(i.e. sum of violations of activity concentration) and*

2. $\sum_{b\in\mathcal{B}}\sum_{t\in\mathcal{T}} p_{bt}$, *where* p_{bt} *denotes the number of activities in excess of yard crane handling capacity in block* $b \in \mathcal{B}$ *and at time period* $t \in \mathcal{T}$. *(i.e. sum of violations of RMG contention).*

3.3.3 Specifications for different yard plan strategies

We now show in a few examples how the generic yard plan template specification can be used to specify various yard plan strategies. Let us recall that the specification is as follows: $\mathcal{D}_\mathcal{T} = (\mathcal{V}, DA^{20}_{it}, DA^{40}_{it}, \mathcal{L}_i)$ and $\mathcal{S}_\mathcal{T} = (\mathcal{R}, \mathcal{G}, \mathcal{B}, \mathcal{C}, CS_r, SR_r, MA_g, RY, MY_b, W^k, W^p, SH, OS)$.

First, let us first define some of the common settings. For all the elements in the demand tuple $\mathcal{D}_\mathcal{T}$, they are not dependent on any yard plan strategies as they specify only the demand side of the problem. For the supply tuple $\mathcal{S}_\mathcal{T}$, it is obvious that the value of CS_r (capacity of a slot), RY (handling rate of a RMG), MY_b (number of RMGs in a block), W^k (weight of activity concentration) and W^p (weight of RMG contention) can be set independently from any yard plan strategy, and hence they are fixed. Depending on the definition of the set of ranges $\mathcal{R}$ and how they are grouped into $\mathcal{G}$, which will be different among different yard plan strategies, SR_r (size or range r), $\mathcal{B}$ (grouping of ranges to blocks) and MA_g (activity thresholds) can be derived according to the definition of $\mathcal{R}$ and $\mathcal{G}$.

Hence, only five of the parameters directly control the yard plan strategy in the specification, namely $\mathcal{R}, \mathcal{G}$, $\mathcal{C}$ (set of clusters), SH (sharing) and OS (one-slot mode), and

therefore we only describe how these are set in the following discussion.

Consolidated (by block) with dedicated/non-dedicated strategy

This is probably the simplest strategy. The yard is partitioned to $\mathcal{R}$ where each range r is a yard block. Each group $g \in \mathcal{G}$ consist of only one range, i.e. a 1-to-1 mapping of groups to ranges. Since there is no clustering of multiple blocks in the assignment, $\mathcal{C}$ is an empty set. In a non-dedicated strategy where two or more services can be assigned to the same yard block, i.e. with sharing, then SH is set to 1, otherwise SH is set to 0. OS is set to zero as this strategy is not in a one-slot mode.

Consolidated (by fixed size yard range) with dedicated/non-dedicated strategy

Let $SIZE$ denote the desired fixed yard range size, and we assume $SIZE > 1$. We partition the yard into $\mathcal{R}$ by combining consecutive slots in the same block of size $SIZE$ into ranges. For blocks with the number of slots in multiples of $SIZE$, they can be partitioned exactly. For blocks that are not, we can partition the block to form ranges that are approximately equal in size, and the average size is not too different from $SIZE$. The exact policy of how this case can be handled should be decided by the terminal operator, and hence we will not prescribe a fixed policy.

The ranges are grouped into groups such that each group consists of ranges in close proximity such that high number of activities in each group can lead to activity concentration. The rules of grouping the ranges depends on the $SIZE$ as well as the geographical location of the ranges, and these are dependent on terminal layout.

Since there is no clustering, $\mathcal{C}$ is a empty set. Depending whether it is a non-

dedicated or dedicated strategy, i.e. allowing two or more services to be assigned to the same yard range or not, then SH is set to 1 or 0, respectively. OS is set to zero as $SIZE$ is bigger than one.

Consolidated (by variable size yard range) with dedicated/non-dedicated strategy

As the yard ranges are variable in size, we model it by having the yard partitioned into $\mathcal{R}$, where each range is a single slot range, i.e. $SR_r = 1$ for all $r \in \mathcal{R}$. We then define $\mathcal{C}$ such that every $c \in \mathcal{C}$ is a set of consecutive ranges and it is a legitimate cluster that can be assigned to services. The set $\mathcal{C}$ can be generated in many ways, and for example, we show a simple algorithm that can generate this set, assuming each cluster size should be least min number of ranges, and at most max number of ranges (see Figure 3.7). For illustration, given a block with five slots, and generating clusters of size three to four, the algorithm will obtain $\mathcal{C} = \{\{1,2,3\},\{2,3,4\},\{3,4,5\},\{1,2,3,4\},\{2,3,4,5\}\}$.

In this case, we model each r to be one slot, and therefore OS is set to one to indicate that we allow a 40-footer to be assigned to two consecutive ranges, i.e. in a one-slot mode. The ranges are grouped into groups $\mathcal{G}$, such that each group consists of ranges in close proximity such that high number of activities in each group can lead to activity concentration. SH is set to 1 or 0, depending on whether the strategy allows sharing or not, i.e. non-dedicated or dedicated strategy, respectively.

3.3.4 A mathematical model

We now formally describe the mathematical model. We first define all the input parameters and the decision variables, followed with the complete mathematical model. We

```
1: for all blocks b ∈ B do
2:    enumerate the ranges in b as (1, 2, ..., |b|)
3:    for size = min to max do
4:       for i = 1 to (|b| - size + 1) do
5:          create a new cluster c = {i, i + 1, ..., i+size−1} in C
6:       end for
7:    end for
8: end for
```

Figure 3.7: An algorithm for generating the set $\mathcal{C}$.

then provide the detailed explanation of each constraint and the objective function.

Input parameters to the model - constants

$\mathcal{T}$ = Set of time periods

$\mathcal{D}_{\mathcal{T}}$ = $(\mathcal{V}, DA_{it}^{20}, DA_{it}^{40}, \mathcal{L}_i)$

$\mathcal{S}_{\mathcal{T}}$ = $(\mathcal{R}, \mathcal{G}, \mathcal{B}, \mathcal{C}, CS_r, SR_r, MA_g, RY, MY_b, W^k, W^p, SH, OS)$

$\mathcal{A}_i$ = Derived from $\mathcal{L}_i$, the set of time periods that are after the loading for service i.

M = Big Number

Decision variables

x_{irt}^{20} = Number of 20-footers received by service i into range r in time t

x_{irt}^{40} = Number of 40-footers received by service i into range r in time t

y_{irt}^{20} = Number of 20-footer retrieval activities by service i from range r in time t

y_{irt}^{40} = Number of 40-footer retrieval activities by service i from range r in time t

d^{20}_{irt} = Number of 20-footer spaces reserved for 20-footers in range r for service i at time t

d^{40}_{irt} = Number of 40-footer spaces reserved for 40-footers in range r for service i at time t

u_{ir} = 1 if service i is assigned to range r, 0 otherwise

k_{rt} = Number of excess activities in range r at time t, such that the k_{rt} contributed to some groups $g \in \mathcal{G}$ to violate MA_g

p_{bt} = Number of activities which need to wait for a yard crane, measured in number of activities above the capacity of the yard cranes in the block b at time t

w_r = 1 if range r has a 40-footer container, 0 otherwise

s^{20}_r = Number of 20-footer slots needed in range r

s^{40}_r = Number of 40-footer slots needed in range r

h_{ic} = 1 if cluster c is assigned to service i, 0 otherwise.

$x^{20}_{irt}, x^{40}_{irt}, p_{bt}, k_{rt} \geq 0$ are real variables. u_{ir}, w_r, h_{ic} are binary variables. $s^{20}_r, s^{40}_r \geq 0$ are integer variables. $y^{20}_{irt}, y^{40}_{irt}, d^{20}_{irt}, d^{40}_{irt}$ are derived variables.

Model

Objective Function

$$\text{Minimise } \left(W^k \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} k_{rt} + W^p \sum_{b \in \mathcal{B}} \sum_{t \in \mathcal{T}} p_{bt} \right)$$

Constraints

$$\sum_{r\in\mathcal{R}} x_{irt}^{20} = DA_{it}^{20} \quad \forall i \in \mathcal{V}, t \in \mathcal{T} \tag{3.1}$$

$$\sum_{r\in\mathcal{R}} x_{irt}^{40} = DA_{it}^{40} \quad \forall i \in \mathcal{V}, t \in \mathcal{T} \tag{3.2}$$

$$y_{irt}^{20} = \left(\frac{\sum_{t'\in\mathcal{T}} x_{irt'}^{20}}{(1+|\mathcal{L}_i|)/2}\right) \quad \forall i \in \mathcal{V}, r\in \mathcal{R}, t \in \mathcal{L}_i \tag{3.3}$$

$$y_{irt}^{40} = \left(\frac{\sum_{t'\in\mathcal{T}} x_{irt'}^{40}}{(1+|\mathcal{L}_i|)/2}\right) \quad \forall i \in \mathcal{V}, r\in \mathcal{R}, t \in \mathcal{L}_i \tag{3.4}$$

$$\sum_{t\in\mathcal{T}} y_{irt}^{20} + \sum_{t\in\mathcal{T}} y_{irt}^{40} + OS \sum_{t\in\mathcal{T}} y_{ir^-t}^{40} \geq u_{ir} \quad \forall i \in \mathcal{V}, r \in \mathcal{R} \tag{3.5}$$

$$\sum_{t\in\mathcal{T}} y_{irt}^{20} + \sum_{t\in\mathcal{T}} y_{irt}^{40} + OS \sum_{t\in\mathcal{T}} y_{ir^-t}^{40} \leq u_{ir}M \quad \forall i \in \mathcal{V}, r \in \mathcal{R} \tag{3.6}$$

$$\sum_{i\in\mathcal{V}} u_{ir} \leq 1 + M(SH) \quad \forall r \in \mathcal{R} \tag{3.7}$$

$$\sum_{r\in g}\left(\sum_{i\in\mathcal{V}} \left(x_{irt}^{20} + x_{irt}^{40} + y_{irt}^{20} + y_{irt}^{40}\right) - k_{rt}\right) \leq MA_g \quad \forall g \in \mathcal{G}, t \in \mathcal{T} \tag{3.8}$$

$$\sum_{i\in\mathcal{V}}\sum_{r\in b} \left(x_{irt}^{20} + x_{irt}^{40} + y_{irt}^{20} + y_{irt}^{40}\right) - p_{bt} \leq MY_b(RY) \quad \forall b \in \mathcal{B}, t \in \mathcal{T} \tag{3.9}$$

$$d^{20}_{irt} = \sum_{t' \in \mathcal{A}_i \text{ and } t' \leq t} x^{20}_{irt'} \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{A}_i \tag{3.10}$$

$$d^{20}_{irt} = \sum_{t' \leq t} x^{20}_{irt'} + \sum_{t' \in \mathcal{A}_i} x^{20}_{irt'} \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{T} \text{ and } t \notin \mathcal{A}_i \tag{3.11}$$

$$d^{40}_{irt} = \sum_{t' \in \mathcal{A}_i \text{ and } t' \leq t} x^{40}_{irt'} \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{A}_i \tag{3.12}$$

$$d^{40}_{irt} = \sum_{t' \leq t} x^{40}_{irt'} + \sum_{t' \in \mathcal{A}_i} x^{40}_{irt'} \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{T} \text{ and } t \notin \mathcal{A}_i \tag{3.13}$$

$$s^{20}_r \geq \frac{\sum_{i \in V} d^{20}_{irt}}{CS_r} \quad \forall r \in \mathcal{R}, t \in \mathcal{T}. \tag{3.14}$$

$$s^{40}_r \geq \frac{\sum_{i \in V} d^{40}_{irt}}{CS_r} \quad \forall r \in \mathcal{R}, t \in \mathcal{T}. \tag{3.15}$$

$$s^{20}_r + (2 - OS)s^{40}_r \leq SR_r \quad \forall r \in \mathcal{R} \tag{3.16}$$

$$\sum_{i \in \mathcal{V}, t \in \mathcal{T}} y^{40}_{irt} \leq w_r M + M(1 - OS) \quad \forall r \in \mathcal{R} \tag{3.17}$$

$$\sum_{i \in \mathcal{V}, t \in \mathcal{T}} y^{20}_{irt} + \sum_{i \in \mathcal{V}, t \in \mathcal{T}} y^{20}_{ir^+t} + \sum_{i \in \mathcal{V}, t \in \mathcal{T}} y^{40}_{ir^+t} \leq M(1 - w_r) + M(1 - OS) \quad \forall r \in \mathcal{R}, \exists r^+ \in \mathcal{R} \tag{3.18}$$

$$\sum_{i \in \mathcal{V}, t \in \mathcal{T}} y^{40}_{irt} \leq M(1 - OS) \quad \forall r \in \mathcal{R}, \neg\exists r^+ \in \mathcal{R} \tag{3.19}$$

$$(1 - h_{ic})M + \sum_{r \in c} u_{ir} \geq |c| \quad \forall i \in \mathcal{V}, c \in \mathcal{C} \tag{3.20}$$

$$(1 - u_{ir})M + \sum_{(r \in c) \wedge (c \in \mathcal{C})} h_{ic} \geq 1 \quad \forall i \in \mathcal{V}, r \in \mathcal{R} \tag{3.21}$$

$$\sum_{(r \in c) \wedge (c \in \mathcal{C})} (h_{ic}) \leq 1 + (1 - u_{ir})M \quad \forall i \in \mathcal{V}, r \in \mathcal{R} \tag{3.22}$$

Derivation of the constraints

Constraints 3.1 and 3.2 : Number of containers received

Constraints 3.1 and 3.2 ensure that the total number of 20-footers and 40-footers received at any time period t for service i must equate to DA_{it}^{20} and DA_{it}^{40}, respectively.

Constraints 3.3 and 3.4 : Number of retrieval activities

As retrieval of the $\sum_{t \in \mathcal{T}} x_{irt}^{20}$ and $\sum_{t \in \mathcal{T}} x_{irt}^{40}$ containers for service i in range r may stretch over a few time periods $|\mathcal{L}_i|$, and we do not know exactly the sequence in which these containers will be retrieved from range r, hence we need to make an assumption on how these retrieval activities are divided over $|\mathcal{L}_i|$ time periods. While these activities may possibly be spread out evenly over the $|\mathcal{L}_i|$ periods, or happen all in one time period and leaving other periods with no activities, or in any other combinations, we want to estimate the values for y_{irt}^{20} and y_{irt}^{40} so that we do not underestimate the extent of activity concentration or RMG contention.

We note that the values of y_{irt}^{20} and y_{irt}^{40} corresponding to the above extreme cases are $\frac{\sum_{t \in \mathcal{T}} x_{irt}^{20}}{|\mathcal{L}_i|}$ and $\frac{\sum_{t \in \mathcal{T}} x_{irt}^{40}}{|\mathcal{L}_i|}$, and $\frac{\sum_{t \in \mathcal{T}} x_{irt}^{20}}{1}$ and $\frac{\sum_{t \in \mathcal{T}} x_{irt}^{40}}{1}$, respectively. The denominator is the number of periods we assume the loading will spread out. In the constraints 3.3 and 3.4, we have assumed that we take the average of the two extremes, i.e. the activities

are spread to $(1 + |\mathcal{L}_i|)/2$ number of time periods. Hence the values of y^{20}_{irt} and y^{40}_{irt} are set as $\frac{\sum_{t\in\mathcal{T}} x^{20}_{irt}}{(1+|\mathcal{L}_i|)/2}$ and $\frac{\sum_{t\in\mathcal{T}} x^{40}_{irt}}{(1+|\mathcal{L}_i|)/2}$, respectively.

Constraints 3.5, 3.6, and 3.7 : Assignment of ranges to services

Constraints 3.5 and 3.6 ensure that the value of u_{ir} is set to 1 when range r is assigned to service i (i.e. when y_{irt}'s are not zero), and 0 otherwise. Note that for convenience, we have also let r^- denote the previous range to r, i.e. $r^{+-} = r$. In the case when $OS = 1$, if a 40-footer belonging to service i is assigned to range r^-, it makes range r to be assigned to i as well, as a 40-footer occupies two consecutive ranges. The term $OS \sum_{t\in\mathcal{T}} y^{40}_{ir^-t}$ in the constraints 3.5 and 3.6 ensures the assignment of a 40-footer to range r^- is considered for the setting of u_{ir}.

In the case when $SH = 0$ (a dedicated yard plan strategy where we do not allow sharing of ranges between services), at most one service i should have $u_{ir} = 1$ for each r. Hence constraint 3.7 ensures $\sum_{i\in\mathcal{V}} u_{ir} \leq 1$, and in the case where $SH = 1$, $\sum_{i\in\mathcal{V}} u_{ir}$ is unbounded.

Constraint 3.8 : Activity concentration violation

We let k_{rt} denote the surplus number of activities in range r at time t, such that the k_{rt} contributes to some groups $g \in \mathcal{G}$ to violate MA_g. To further elaborate, if the activities in every range r at time t can be reduced by k_{rt} simultaneously, then all groups $g \in \mathcal{G}$ will have activity levels within the limits of MA_g, and this is ensured in constraint 3.8. Although k_{rt} has no upper bound, k_{rt} is minimised in the objective function. Hence, the value of k_{rt} will be the minimum number of activity concentration violations each range r at time t can get.

Constraint 3.9 : RMG contention violation

Constraint 3.9 ensures that the number of activities in each block is less than or equal to the total handing capacity of the RMGs in the block (i.e. number of RMGs multiplied by the handling rate of each RMG). Any violation will be captured in the slack variable p_{bt} as RMG contention. Although p_{bt} has no upper bound, we also minimise p_{bt} in the objective, and hence the value of p_{bt} will be the minimum number of violations of the RMG contention.

Constraints 3.10, 3.11, 3.12 and 3.13 : Derived variables d^{20}_{irt} and d^{40}_{irt}

We let d^{20}_{irt} and d^{40}_{irt} denote the number of yard spaces reserved for 20- and 40-footers, respectively, for service i in range r at time period t. Yard space is reserved for the service i as long as its containers are in the yard, and we continue to reserve the space during the loading periods $\mathcal{L}_i$ of the service. For illustration, Figure 3.5 (in Page 39) gives three scenarios and shows graphically how the yard spaces are reserved with respect to the number of TEUs in the yard (inventory level).

As the yard plan template assumes that time will wrap around from the last time period back to 1, we note that any containers received for service i after the loading period $\mathcal{L}_i$ will occupy the yard space until the end of the time window, and continue to do so at time period 1, until the end of the loading period $\mathcal{L}_i$. This can also be observed from the Figure 3.5. For convenience, we define a derived input parameter $\mathcal{A}_i$ to be the set of time periods after the loading period of service i. Figure 3.5 also illustrates how each $\mathcal{A}_i$ is defined with respect to the $\mathcal{L}_i$. Constraints 3.10 and 3.12 ensure that yard spaces are reserved for time periods in $\mathcal{A}_i$ by accumulating received

containers only in $\mathcal{A}_i$. On the other hand, Constraints 3.11 and 3.13 ensure that yard spaces are reserved for time periods outside $\mathcal{A}_i$ by accumulating received containers outside $\mathcal{A}_i$ (i.e. $\sum_{t' \leq t} x^{20}_{irt'}$ and $\sum_{t' \leq t} x^{40}_{irt'}$), and all the containers received inside $\mathcal{A}_i$ (i.e. $\sum_{t' \in \mathcal{A}_i} x^{20}_{irt'}$ and $\sum_{t' \in \mathcal{A}_i} x^{40}_{irt'}$) as they are wrapped around.

Constraints 3.14, 3.15 and 3.16 : Slot constraint

Since 20-footers and 40-footers cannot be stacked in the same slot in a range, we have to compute the number of slots reserved for 20- and 40-footers separately and ensure that the total is less than the number of slots in the range. We let the integer variables s^{20}_r and s^{40}_r denote the number of 20-footer slots and 40-footer slots required in range r, respectively.

The number of slots needed for 20-footers or 40-footers in a time period is computed using the total space reservation for 20-footers or 40-footers divided by the slot capacity, respectively. Since the demand for slots changes over time, we take the maximum over all time periods. Hence, $s^{20}_r = \left\lceil \frac{\max_{t \in \mathcal{T}} \left(\sum_{i \in V} d^{20}_{irt} \right)}{CS_r} \right\rceil$ and $s^{40}_r = \left\lceil \frac{\max_{t \in \mathcal{T}} \left(\sum_{i \in V} d^{40}_{irt} \right)}{CS_r} \right\rceil$. These can be re-written as Constraints 3.14 and 3.15.

Note that in the case when $OS = 0$ (not in a one-slot mode), s^{40}_r denote the number of 40-footer slots, and therefore it requires double the number in terms of slots (which are 20 feet wide) compared to s^{20}_r. Hence, the total number of slots required in range r is expressed as $s^{20}_r + (2 \times s^{40}_r)$. However, in the case when $OS = 1$, when a 40-footer occupies a range r, it also occupies the next range r^+. The usage of the other slot r^+ is not accounted for in range r and hence the total number of slots required in range r is expressed as $s^{20}_r + s^{40}_r$. Combining the two cases, the constraint is then given by

Constraint 3.16.

Constraints 3.17, 3.18 and 3.19 : Modelling of 40-footers occupying two consecutive ranges when $OS = 1$

Since we cannot stack 20-footers and 40-footers in the same slot, and when $OS = 1$, we have assumed that a 40-footer occupying range r also occupies r^+, we have to ensure that there is no conflict in size in the assignment of range r and r^+. Hence, when a 40-footer occupies range r, we will constrain the range r not to have a 20-footer assigned, and the next range r^+ not to have any containers assigned. This is achieved by Constraint 3.17 setting the indicator variable w_r to 1 whenever range r has at least a 40-footers assigned, and Constraint 3.18 to ensure that no 20-footers in range r and no containers in range r^+. Constraint 3.19 ensures that a 40-footer cannot be assigned to the last range in a block (indicated by $\neg\exists r^+ \in \mathcal{R}$), otherwise it will protrude out of the block.

Constraints 3.20, 3.21 and 3.22 : Constraints for cluster formation

In a strategy where $\mathcal{C}$ is specified (i.e. consecutive ranges are to be assigned to the same service as a cluster), we have to ensure feasibility in the assignment of ranges to the services such that if a range r is assigned to a service i, then there must exist exactly a cluster c in $\mathcal{C}$, where $r \in \mathcal{C}$ and for all $r' \in c$, they must be also assigned to i. This is achieved by introducing a binary indicator variable h_{ic}, which when set to 1 means that cluster c is assigned to i, and zero otherwise. Constraint 3.21 and 3.22 ensure that when r is assigned to a service i, there is exactly one c such that $h_{ic} = 1$. To ensure that all ranges $r' \in c$ are also assigned to the same service, we sum the

assignment variable u_{ir} for all the ranges in c and check that it is exactly equal to the size of c, and this is done in Constraint 3.20.

The objective function

The objective function is the weighted sum of the k and p variables. The values of both k and p variables store the number of activities that have to wait within a time period due to either activity concentration or RMG contention. The weights W^k and W^p are used to provide the relative importance of these two types of waiting. As both k and p are of the same unit of measurement, in most practical cases, both the weights should be set to 1, that is, they should be weighted equally.

3.4 Summary

In this chapter, we present the motivation of the thesis in three areas, namely, defining a generic yard plan template problem, solving the problem in a reasonable time, and lastly finding nimble solutions that can recover easily after uncertainty is revealed. We define the scope of the problem by providing the terminal configuration used in the thesis, and a list of assumptions made. We then present the first area of focus, a generic yard plan template problem, by first giving a generic problem specification, followed by a detailed mathematical model. In the following two chapters, we will present the solution approaches to solve the yard plan template problem quickly when inputs are deterministic, and solution approaches to a good yard plan template when inputs are non-deterministic, respectively.

Chapter 4

Static Yard Plan Template Model

A search engine can be designed to discover the best yard plan strategy for a container terminal given its scenario. However, that requires the generic model presented in Chapter 3 to be solved repeatedly, and hence it has to be solved in a reasonable amount of time. As the complexity of the problem depends on the strategy, we propose having different solution approaches for different strategies. Instead of covering all yard plan strategies, which could be too large a scope for the thesis, in this chapter, we present solution approaches to solve some cases of the generic yard template problem. We will cover only two strategies, namely, 1) consolidated and dedicated strategy, and 2) consolidated and non-dedicated strategy. By consolidated, we mean that the containers to be loaded into the same service will be stacked into groups with fixed number of slots in each group. In a dedicated strategy, containers to be loaded into different services cannot be mixed in a same group, while in a non-dedicated strategy, containers to be loaded into different services can be mixed in the same group.

As the mathematical model given in Chapter 3 Section 3.3.4 is meant to be generic for all strategies, we reproduce a simplified version of the same model in Section 4.1

for the two above-mentioned strategies. We then present two solution approaches as follows. In Section 4.2, we present the first solution approach as solving the mathematical model using ILOG CPLEX. However, we find that the run time is extremely long, and that motivates us to re-examine some of the constraints and we find that by re-modelling some of the constraints, we reduce the run time tremendously. The new model is presented, and the results of the experiments are also discussed. In Section 4.3, we present the second solution approach, which is a heuristic algorithm based on local search. This heuristic is designed only for consolidated and dedicated strategy, exploiting the non-sharing property. The algorithm and the experimental results are presented. Finally, we summarise the chapter in Section 4.4.

4.1 A Mathematical Model

We simplify the mathematical model for the consolidated strategy by setting $OS = 0$ and $\mathcal{C}$ to be an empty set (i.e. no clusters). We partition the yard into $\mathcal{R}$, and the ranges in $\mathcal{R}$ are the groups that the model will assign services to. Let's call this simplified mathematical model Model **ORIG**, and describe it as follows.

Input parameters to the model - constants

$\mathcal{T}$ = Set of time periods

$\mathcal{D}_{\mathcal{T}}$ = $(\mathcal{V}, DA_{it}^{20}, DA_{it}^{40}, \mathcal{L}_i)$

$\mathcal{S}_{\mathcal{T}}$ = $(\mathcal{R}, \mathcal{G}, \mathcal{B}, \mathcal{C}, CS_r, SR_r, MA_g, RY, MY_b, W^k, W^p, SH, OS)$

$\mathcal{A}_i$ = Derived from $\mathcal{L}_i$, the set of time periods that are after the loading for service i

M = Big Number

Decision variables

x_{irt}^{20} = Number of 20-footers received by service i into range r in time t

x_{irt}^{40} = Number of 40-footers received by service i into range r in time t

y_{irt}^{20} = Number of 20-footer retrieval activities by service i from range r in time t

y_{irt}^{40} = Number of 40-footer retrieval activities by service i from range r in time t

d_{irt}^{20} = Number of 20-footer spaces reserved for 20-footers in range r for service i at time t

d_{irt}^{40} = Number of 40-footer spaces reserved for 40-footers in range r for service i at time t

u_{ir} = 1 if service i is assigned to range r, 0 otherwise

k_{rt} = Number of excess activities in range r at time t, such that the k_{rt} contributed to some groups $g \in \mathcal{G}$ to violate MA_g

p_{bt} = Number of activities which need to wait for a yard crane, measured in number of activities above the capacity of the yard cranes in the block b at time t.

s_r^{20} = Number of 20-footer slots needed in range r

s_r^{40} = Number of 40-footer slots needed in range r

$x^{20}_{irt}, x^{40}_{irt}, p_{bt}, k_{rt} \geq 0$ are real variables. u_{ir} are binary variables. $s^{20}_{r}, s^{40}_{r} \geq 0$ are integer variables. $y^{20}_{irt}, y^{40}_{irt}, d^{20}_{irt}, d^{40}_{irt}$ are derived variables.

Objective function

$$\text{Minimise} \left(W^k \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} k_{rt} + W^p \sum_{b \in \mathcal{B}} \sum_{t \in \mathcal{T}} p_{bt} \right)$$

Constraints

$$\sum_{r \in \mathcal{R}} x^{20}_{irt} = DA^{20}_{it} \quad \forall i \in \mathcal{V}, t \in \mathcal{T} \tag{4.1}$$

$$\sum_{r \in \mathcal{R}} x^{40}_{irt} = DA^{40}_{it} \quad \forall i \in \mathcal{V}, t \in \mathcal{T} \tag{4.2}$$

$$y^{20}_{irt} = \left(\frac{\sum_{t' \in \mathcal{T}} x^{20}_{irt'}}{(1 + |\mathcal{L}_i|)/2} \right) \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{L}_i \tag{4.3}$$

$$y^{40}_{irt} = \left(\frac{\sum_{t' \in \mathcal{T}} x^{40}_{irt'}}{(1 + |\mathcal{L}_i|)/2} \right) \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{L}_i \tag{4.4}$$

$$\sum_{t \in \mathcal{T}} y^{20}_{irt} + \sum_{t \in \mathcal{T}} y^{40}_{irt} \geq u_{ir} \quad \forall i \in \mathcal{V}, r \in \mathcal{R} \tag{4.5}$$

$$\sum_{t \in \mathcal{T}} y^{20}_{irt} + \sum_{t \in \mathcal{T}} y^{40}_{irt} \leq u_{ir} M \quad \forall i \in \mathcal{V}, r \in \mathcal{R} \tag{4.6}$$

$$\sum_{i \in \mathcal{V}} u_{ir} \leq 1 + M(SH) \quad \forall r \in \mathcal{R} \tag{4.7}$$

$$\sum_{r \in g} \left(\sum_{i \in \mathcal{V}} \left(x^{20}_{irt} + x^{40}_{irt} + y^{20}_{irt} + y^{40}_{irt} \right) - k_{rt} \right) \leq MA_g \quad \forall g \in \mathcal{G}, t \in \mathcal{T} \tag{4.8}$$

$$\sum_{i\in\mathcal{V}}\sum_{r\in b}\left(x^{20}_{irt}+x^{40}_{irt}+y^{20}_{irt}+y^{40}_{irt}\right)-p_{bt} \quad \leq \quad MY_b(RY) \quad \forall b\in\mathcal{B}, t\in\mathcal{T} \tag{4.9}$$

$$d^{20}_{irt} = \sum_{t'\in\mathcal{A}_i \text{ and } t'\leq t} x^{20}_{irt'} \quad \forall i\in\mathcal{V}, r\in\mathcal{R}, t\in\mathcal{A}_i \tag{4.10}$$

$$d^{20}_{irt} = \sum_{t'\leq t} x^{20}_{irt'} + \sum_{t'\in\mathcal{A}_i} x^{20}_{irt'} \quad \forall i\in\mathcal{V}, r\in\mathcal{R}, t\in\mathcal{T} \text{ and } t\notin\mathcal{A}_i \tag{4.11}$$

$$d^{40}_{irt} = \sum_{t'\in\mathcal{A}_i \text{ and } t'\leq t} x^{40}_{irt'} \quad \forall i\in\mathcal{V}, r\in\mathcal{R}, t\in\mathcal{A}_i \tag{4.12}$$

$$d^{40}_{irt} = \sum_{t'\leq t} x^{40}_{irt'} + \sum_{t'\in\mathcal{A}_i} x^{40}_{irt'} \quad \forall i\in\mathcal{V}, r\in\mathcal{R}, t\in\mathcal{T} \text{ and } t\notin\mathcal{A}_i \tag{4.13}$$

$$s^{20}_r \geq \frac{\sum_{i\in V} d^{20}_{irt}}{CS_r} \quad \forall r\in\mathcal{R}, t\in\mathcal{T}. \tag{4.14}$$

$$s^{40}_r \geq \frac{\sum_{i\in V} d^{40}_{irt}}{CS_r} \quad \forall r\in\mathcal{R}, t\in\mathcal{T}. \tag{4.15}$$

$$s^{20}_r + 2s^{40}_r \leq SR_r \quad \forall r\in\mathcal{R} \tag{4.16}$$

4.2 Approach 1: Solving as a mathematical model

We solve the problem by solving the mathematical model in ILOG CPLEX. Experiments with the model ORIG show that the run time is too long and lead us to re-model some of the constraints into a new model IMPR. We then present experimental results comparing ORIG and IMPR.

4.2.1 Experimental setup

We conduct the experiments with the model ORIG coded with CPLEX 11.0 running a single thread on an Intel Core 2 Duo 3.16 GHz machine with 3 GB RAM. Experiments

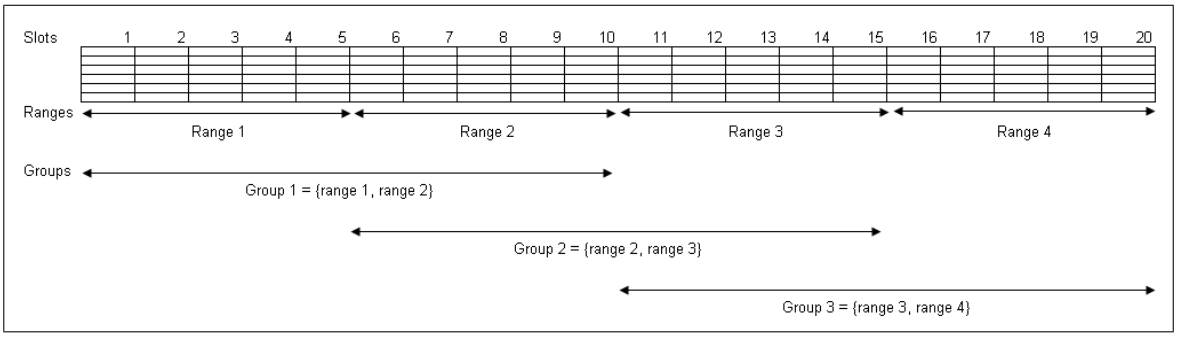


Figure 4.1: Example of a yard block with 20 slots, partitioned into four ranges of five slots each, and the ranges grouped into three groups of two ranges each.

are conducted on a medium-sized terminal capable of berthing three vessels at the same time as described in Chapter 3, Section 3.2. We further partition the blocks into ranges of five slots for the set $\mathcal{R}$. Adjacent ranges are grouped in pairs into $g \in \mathcal{G}$ as illustrated in Figure 4.1. MA_g are set to 120. We also assume a RMG is able to handle 120 moves per time period, and each yard block has three RMGs. The weights for activity concentration and RMG contention, W^k and W^p, are both set to 1. This is because both measurements (i.e. k_{rt} and p_{bt}) are in the same unit of measure, being the number of activities exceeding the thresholds, and hence a unit of violation in k_{rt} is equivalent to one unit in p_{bt}.

We model a seven-day time window of three time periods per day, hence a total of 21 time periods in the time window. 21 services will arrive throughout the week, with one service arriving in each time period, and staying alongside for loading for three consecutive time periods. Hence in any time period, three services are loading concurrently. All services are assumed to be homogeneous, and will load the same number of 20-footers and 40-footers. The number of TEUs received by each service prior to the loading time periods follows the same arrival schedule. Without loss of generality, let's

assume that a service is to load in periods 19, 20 and 21, as shown in Figure 4.2. The schedule depicts that 3.4% of the TEUs will arrive into the yard in time period 7, 6.6% in time period 8, and so on, and continue to arrive in subsequent periods to period 17. Figure 4.2 also shows graphically the cumulative percentage of the TEUs should have arrived by each time period, where 100 percent will be the final TEUs the service will load.

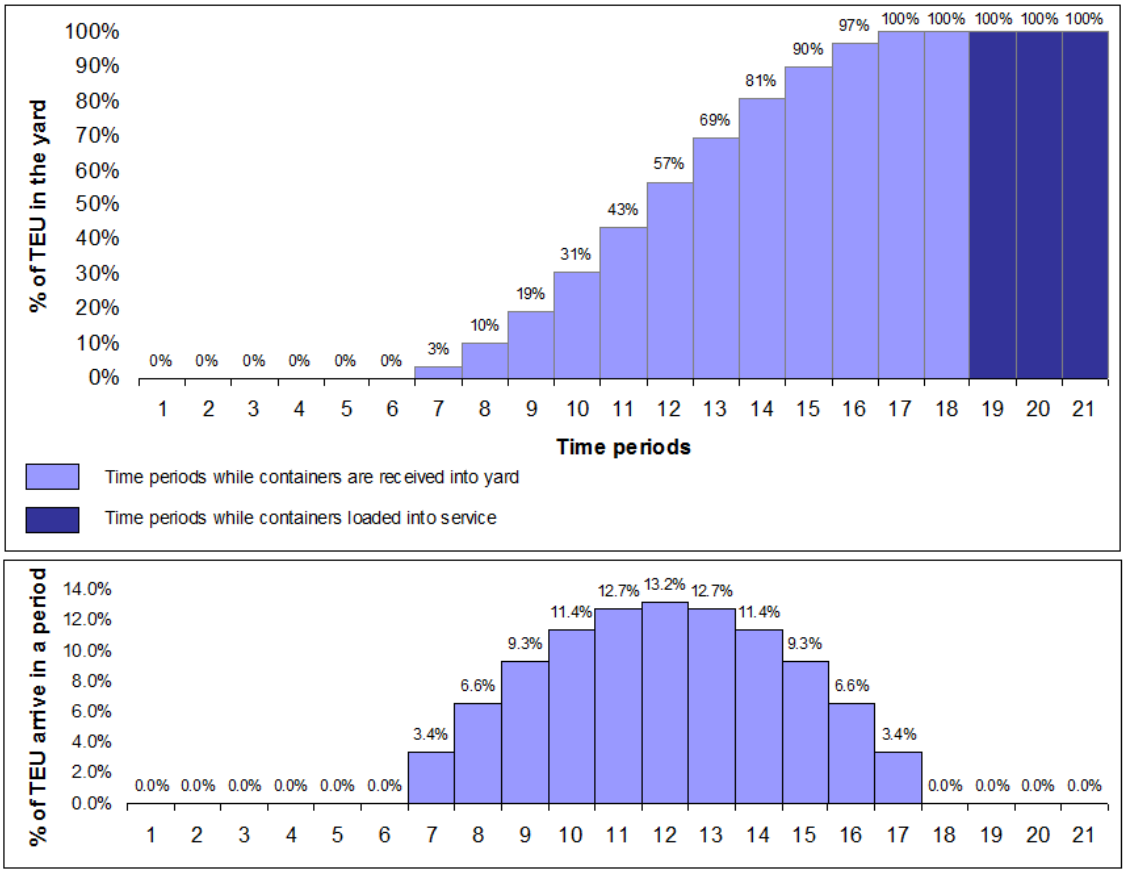


Figure 4.2: Percentage TEU of the total loaded containers received into the yard prior to the loading period for each service.

We run multiple experiments by varying the **workload** from 0.1 to 1.0. Workload is defined to be the ratio of total number of TEUs to be loaded in a week to the maximum

stacking capacity of the entire yard in TEUs. In the experiments, we assume each 20-footer slot has a stacking capacity of 24 TEUs. Hence the maximum stacking capacity of the entire yard is $20 \times 40 \times 24 = 19200$ TEUs. Table 4.1 tabulates for each value of workload, the number of containers to be loaded in the whole week as well as the number of containers to be loaded for each service.

Table 4.1: Table shows, for each workload, the number of containers loaded for the whole week, and containers loaded for each service, respectively.

	Total containers in a week			Containers loaded to a service		
Workload	20-footers	40-footers	TEU	20-footers	40-footers	TEU
0.1	630	630	1890	30	30	90
0.2	1260	1260	3780	60	60	180
0.3	1911	1911	5733	91	91	273
0.4	2541	2541	7623	121	121	363
0.5	3192	3192	9576	152	152	456
0.6	3822	3822	11466	182	182	546
0.7	4473	4473	13419	213	213	639
0.8	5103	5103	15309	243	243	729
0.9	5754	5754	17262	274	274	822
1.0	6384	6384	19152	304	304	912
1.1	7035	7035	21105	335	335	1005
1.2	7665	7665	22995	365	365	1095
1.3	8316	8316	24948	396	396	1188
1.4	8946	8946	26838	426	426	1278
1.5	9597	9597	28791	457	457	1371
1.6	10227	10227	30681	487	487	1461
1.7	10878	10878	32634	518	518	1554
1.8	11508	11508	34524	548	548	1644
1.9	12159	12159	36477	579	579	1737
2.0	12789	12789	38367	609	609	1827

4.2.2 Experimental results

For comparison, we have relaxed the model ORIG by relaxing s_r^{20} and s_r^{40} to be real numbers instead of integers, and we call this relaxed model **ORIG-Relax**. Note that when we set $SH = 0$, i.e. we do not allow sharing of ranges between services in a dedicated strategy, we find that there is no feasible solution for ORIG with workload at and above 0.9. This is because at a workload of 0.9, each service will load 274 $\times$ 20-footers and 274 $\times$ 40-footers, which require 12 $\times$ 20-footer slots and 12 $\times$ 40-footer slots, or a total demand of 36 $\times$ 20-footer slots. As each range has only 5 $\times$ 20-footer slots, we need at least eight ranges to be allocated to each service, and since we have 21 services, we need a total of 168 ranges. However, the entire yard has only 160 ranges, and that implies that there will be no feasible solution. Also, there is no feasible solution for ORIG-Relax for a workload at and above 1.0, as each service will load 912 TEUs, which translates to 38 $\times$ 20-footer slots (ignoring the integer allocation of 20- and 40-footer slots). With each service to be assigned with eight ranges, a total of 168 ranges is needed for all services, and hence there will be no feasible solution.

Table 4.3 (in Page 70) provides the model size in terms of the size of the matrix that CPLEX solves and also the number of integer and binary variables. The models are considerably large, with more than one or two million non-zero entries in the matrix, and about 150 thousand variables, of which more than 3500 are binary variables, and 320 are integers (only for ORIG). In Table 4.4 (in Page 71), columns (A) and (B) summarise the run times of the experiments for ORIG and ORIG-Relax, respectively, with SH set to zero. All the experiments are terminated with a time-out limit of eight hours, and those that run beyond the time-out limit are indicated by "$*$". All the runs that terminate

produce zero as objective value.

From the Table 4.4, we see that model ORIG runs more than four hours at workload 0.3, and at higher workload, it fails to terminate within the time-out limit. For model ORIG-Relax, it runs with the maximum workload of 0.9, and the run time is only about 12 minutes. This shows that the integer variables s_r^{20} and s_r^{40} are causing the CPLEX a huge amount of time for the branch-and-bound. It motivates us to the next section where we explore a re-modelling of these variables and a tremendous speed up of run time is observed.

4.2.3 Re-modelling of Constraints 4.14, 4.15 and 4.16

We note that for cases of small SR_r, we could re-model the problem by adopting a similar concept as column generation, i.e. to pre-determine all possible combinations of the integer variables s_r^{20} and s_r^{40}, and apply suitable constraints in choosing them. In the following sub-sections, we illustrate the case of $SR_r = 5$, and then give a generalised formulation for any value of $SR_r \geq 2$.

Illustration of model when $SR_r = 5$

In the case where SR_r is 5 (i.e. a range has 5 $\times$ 20-footer slots), when there is no 40-footer in the range, there can be up to five 20-footer slots, and when there is one 40-footer slot in the range, there can be up to three 20-footer slots, since one 40-footer slot will occupy two of the five slots. Similarly, when there are two 40-footer slots occupied, there can be only one 20-footer slot. Hence, the combinations of values of (s_r^{20}, s_r^{40}) are $(s_r^{20} \leq 5$ and $s_r^{40} \leq 0)$ or $(s_r^{20} \leq 3$ and $s_r^{40} \leq 1)$ or $(s_r^{20} \leq 1$ and $s_r^{40} \leq 2)$. It can be

shown logically that it is equivalent to have $s_r^{20} \leq 5$ and $s_r^{40} \leq 2$ and $(s_r^{20} \leq 3$ or $s_r^{40} \leq 0)$ and $(s_r^{20} \leq 1$ or $s_r^{40} \leq 1)$.

These can be modelled in the following constraints with two sets of binary variables $j_{1,r}$ and $j_{2,r}$.

$$\frac{\sum_{i \in V} d_{irt}^{20}}{CS_r} \leq 5 \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.17}$$

$$\frac{\sum_{i \in V} d_{irt}^{40}}{CS_r} \leq 2 \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.18}$$

$$\frac{\sum_{i \in V} d_{irt}^{20}}{CS_r} \leq M(j_{1,r}) + 3 \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.19}$$

$$\frac{\sum_{i \in V} d_{irt}^{40}}{CS_r} \leq M(1 - j_{1,r}) \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.20}$$

$$\frac{\sum_{i \in V} d_{irt}^{20}}{CS_r} \leq M(j_{2,r}) + 1 \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.21}$$

$$\frac{\sum_{i \in V} d_{irt}^{40}}{CS_r} \leq M(1 - j_{2,r}) + 1 \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.22}$$

These constraints replace the constraints 4.14, 4.15 and 4.16 in ORIG, for the case of $SR_r = 5$. We call this improved model **IMPR-M** (indicating big-M is used).

Although the big-M method implements the constraints easily, we find that we can make use of the unique structure of the constraints and replace big-M with a number that is just big enough to serve its purpose, and tight enough to define the bounds. By doing so, we eliminate the big-M from the constraints, and also reduce the number of constraints by removing constraints 4.17 and 4.18 as they become redundant. The following constraints implement the same constraints without the big-M.

$$\frac{\sum_{i\in V} d^{20}_{irt}}{CS_r} \leq 2(j_{1,r}) + 3 \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.23}$$

$$\frac{\sum_{i\in V} d^{40}_{irt}}{CS_r} \leq 2(1 - j_{1,r}) \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.24}$$

$$\frac{\sum_{i\in V} d^{20}_{irt}}{CS_r} \leq 4(j_{2,r}) + 1 \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.25}$$

$$\frac{\sum_{i\in V} d^{40}_{irt}}{CS_r} \leq (1 - j_{2,r}) + 1 \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.26}$$

For an illustration of how $j_{1,r}$ and $j_{2,r}$ works, we consider an example. When $j_{1,r} = 1$, we have constraints 4.23 and 4.24 to be $\frac{\sum_{i\in V} d^{20}_{irt}}{CS_r} \leq 5$, and $\frac{\sum_{i\in V} d^{40}_{irt}}{CS_r} \leq 0$, respectively, imposing no slots in r to be used for 40-footers. When $j_{1,r} = 0$, we have $\frac{\sum_{i\in V} d^{20}_{irt}}{CS_r} \leq 3$, and $\frac{\sum_{i\in V} d^{40}_{irt}}{CS_r} \leq 2$, and in this case, it depends on value of $j_{2,r}$. When $j_{2,r} = 1$, we have constraints 4.25 and 4.26 to be $\frac{\sum_{i\in V} d^{20}_{irt}}{CS_r} \leq 5$, and $\frac{\sum_{i\in V} d^{40}_{irt}}{CS_r} \leq 1$, respectively, hence together with constraints 4.23 and 4.24, imposing three slots in r for 20-footers and one slot in r for 40-footers. When $j_{2,r} = 0$, we have constraints 4.25 and 4.26 to be $\frac{\sum_{i\in V} d^{20}_{irt}}{CS_r} \leq 1$, and $\frac{\sum_{i\in V} d^{40}_{irt}}{CS_r} \leq 2$, respectively, hence imposing one slot in r for 20-footers and two slots in r for 40-footers.

Constraints 4.23, 4.24, 4.25 and 4.26 will replace the constraints 4.14, 4.15 and 4.16 in ORIG, and we called this improved model **IMPR**. The following section gives a generalised formulation for IMPR with $SR_r = n$.

Generalised model with $SR_r \geq 2$

We provide here the generalised model for any value of $SR_r = n$ where $n \geq 2$. Note that when n is even, the model formulation of the cases where $SR_r = n$ and $SR_r = n + 1$ is

similar with the same number of constraints and same number of binary variables j_{mr}. For simplifying the explanation that follows, we temporarily introduce a binary indicator $nODD$ which is set to 1 when n is odd, and 0 otherwise.

First, we determine all possible combinations of (s_r^{20}, s_r^{40}) for any given n. Using the basis where a 40-footer will occupy two 20-footer slots, the combination of value of (s_r^{20}, s_r^{40}) can be derived as $(s_r^{20} \leq n$ and $s_r^{40} \leq 0)$ or $(s_r^{20} \leq n-2(1)$ and $s_r^{40} \leq 1)$ or $(s_r^{20} \leq n-2(2)$ and $s_r^{40} \leq 2)$ or ... or $(s_r^{20} \leq n-2(i)$ and $s_r^{40} \leq i)$ or ... or $(s_r^{20} \leq nODD$ and $s_r^{40} \leq \frac{(n-nODD)}{2})$. It can be shown logically that it is equivalent to have $s_r^{20} \leq n$ and $s_r^{40} \leq \frac{(n-nODD)}{2})$ and $(s_r^{20} \leq n-2(1)$ or $s_r^{40} \leq 0)$ and ... and $(s_r^{20} \leq n-2(i)$ or $s_r^{40} \leq i-1)$ and ... and $(s_r^{20} \leq nODD)$ or $s_r^{40} \leq \frac{(n-nODD)}{2}-1)$.

As there are $\frac{(n-nODD)}{2}$ (which is $\lfloor \frac{SR_r}{2} \rfloor$) sets of disjunctive constraints for each range r, we will then need $\lfloor \frac{SR_r}{2} \rfloor$ number of binary variables for each range r, i.e. j_{mr} for $1 \leq m \leq \lfloor \frac{SR_r}{2} \rfloor$, to implement these disjunctive constraints. We then have the following constraints to replace the constraints 4.14, 4.15 and 4.16 in ORIG to give us model IMPR-M.

$$\frac{\sum_{i \in V} d_{irt}^{20}}{CS_r} \leq SR_r \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.27}$$

$$\frac{\sum_{i \in V} d_{irt}^{40}}{CS_r} \leq \left\lfloor \frac{SR_r}{2} \right\rfloor \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{4.28}$$

$$\frac{\sum_{i \in V} d_{irt}^{20}}{CS_r} \leq M(j_{mr}) + SR_r - 2m \quad \forall r \in \mathcal{R}, t \in \mathcal{T}, 1 \leq m \leq \left\lfloor \frac{SR_r}{2} \right\rfloor \tag{4.29}$$

$$\frac{\sum_{i \in V} d_{irt}^{40}}{CS_r} \leq M(1 - j_{mr}) + m - 1 \quad \forall r \in \mathcal{R}, t \in \mathcal{T}, 1 \leq m \leq \left\lfloor \frac{SR_r}{2} \right\rfloor \tag{4.30}$$

As described earlier for the case of $SR_r = 5$, we replace big-M with a tighter bound and re-write the constraints without the big-M as follows.

$$\frac{\sum_{i \in V} d^{20}_{irt}}{CS_r} \leq 2m(j_{mr}) + SR_r - 2m \quad \forall r \in \mathcal{R}, t \in \mathcal{T}, 1 \leq m \leq \left\lfloor \frac{SR_r}{2} \right\rfloor \tag{4.31}$$

$$\frac{\sum_{i \in V} d^{40}_{irt}}{CS_r} \leq \left(\left\lfloor \frac{SR_r}{2} \right\rfloor - m + 1 \right)(1 - j_{mr}) + m - 1 \quad \forall r \in \mathcal{R}, t \in \mathcal{T}, 1 \leq m \leq \left\lfloor \frac{SR_r}{2} \right\rfloor \tag{4.32}$$

The generic model for the improved model IMPR will have the constraints 4.14, 4.15 and 4.16 in ORIG replaced with the above constraints 4.31 and 4.32.

We conduct experiments for IMPR-M and IMPR for the same scenario as described in Section 4.2.1. In the following subsections, we show the run times of IMPR and IMPR-M and how they compare with ORIG. We observe big improvements in run time over the model ORIG. We then do more experiments to compare between dedicated versus non-dedicated strategy. The experiments also give insights on how the run time responds to the tightness of the problem in terms of the number of RMGs.

Table 4.2: Summary of the models used in the experiments

Models	s^{20}_r and s^{40}_r modelled	j_{mr} modelled	j_{mr} with Big-M
ORIG	Integer	-	-
ORIG-Relax	Real	-	-
IMPR-M	-	Binary	Yes
IMPR	-	Binary	No

Run time improvement after re-modelling

For ease of reference, the models and their corresponding names are summarised in Table 4.2. Table 4.3 shows the size of the models in CPLEX, and Table 4.4 compares the run times of four different models ORIG, ORIG-Relax, IMPR-M and IMPR, as shown in column A, B, C and D, respectively. These experiments are based on experiments with $SH = 0$. All the experiments are terminated with a time-out limit of eight hours and those that run beyond the time-out limits are indicated by "$*$".

Table 4.3: Summary of Model sizes of Model ORIG, IMPR-M, IMPR and ORIG-Relax.

	Model Size in CPLEX					
	Matrix Size			Number of Variables		
	Rows	Columns	Non-Zero	Integer Vars	Binary Vars	All Vars
ORIG	37582	101380	1377180	320	3520	148741
ORIG-Relax	37582	101380	1377180	0	3520	148741
IMPR-M	51022	101540	2862300	0	3840	148741
IMPR	44302	101540	2123100	0	3840	148741

4.2.4 Experimental results

Comparing ORIG with the other three models confirms the earlier observation that CPLEX runs very slowly when the variables s_r^{20} and s_r^{40} are modelled as integers. It is obvious that re-modelling of s_r^{20} and s_r^{40} into j_{mr} is indeed an improvement of the model, as shown in the experiments for IMPR and IMPR-M. Comparing IMPR versus IMPR-M, the run time of IMPR hovers at the same level as that of IMPR-M for workload up to 0.4, but at workload 0.5 and beyond, IMPR is obviously running much faster than IMPR-M. At a workload of 0.5 and 0.6, IMPR is taking about 10% of the run time of

Table 4.4: Summary of the running times of the models

	Model used			
	ORIG	ORIG-Relax	IMPR-M	IMPR
Workload	(A)	(B)	(C)	(D)
0.1	158.0	13.4	10.36	8.19
0.2	3694.7	21.9	20.63	20.69
0.3	15949.4	80.9	24.05	18.69
0.4	*	118.4	18.53	22.44
0.5	*	245.8	295.63	29.34
0.6	*	312.6	305.75	36.66
0.7	*	395.5	*	107.30
0.8	*	441.9	*	703.19
0.9	*	710.1	*	*
1.0	*	*	*	*

* run did not complete

IMPR-M, and at a workload of 0.7 and 0.8, IMPR is able to terminate in 107 and 703 seconds, respectively, while IMPR-M is not able to terminate within the time-out limit of eight hours. The run time difference is largely attributed to the smaller size of the IMPR, having 13% less rows than IMPR-M. The experiments also show that IMPR is able to find the solution at higher workload while IMPR-M cannot. Hence, the experiments suggest that IMPR is a better model than IMPR-M.

While the purpose of ORIG-Relax is to act as a benchmark against ORIG, an interesting observation is also made while comparing between IMPR and ORIG-Relax. At lower workloads up to 0.7, the IMPR runs faster than ORIG-Relax, even though IMPR has more binary variables, and more rows compared to ORIG-Relax. Note also that ORIG-Relax is also a possible solution approach for strategies with larger ranges, typically 20 slots and above, which makes s_r^{20} and s_r^{40} less important to be solved as integers.

Table 4.5: Summary of the yard utilisation at different workload

	Total containers in yard at any one time			
Workload	20-footers	40-footers	TEU	Yard Utilization
0.1	300	300	900	4.7%
0.2	600	600	1800	9.4%
0.3	910	910	2730	14.2%
0.4	1210	1210	3630	18.9%
0.5	1520	1520	4560	23.8%
0.6	1820	1820	5460	28.4%
0.7	2130	2130	6390	33.3%
0.8	2430	2430	7290	38.0%
0.9	2740	2740	8220	42.8%
1.0	3040	3040	9120	47.5%
1.1	3350	3350	10050	52.3%
1.2	3650	3650	10950	57.0%
1.3	3960	3960	11880	61.9%
1.4	4260	4260	12780	66.6%
1.5	4570	4570	13710	71.4%
1.6	4870	4870	14610	76.1%
1.7	5180	5180	15540	80.9%
1.8	5480	5480	16440	85.6%
1.9	5790	5790	17370	90.5%
2.0	6090	6090	18270	95.2%

Comparing dedicated vs non-dedicated strategy

Earlier experiments test IMPR on the dedicated strategy. In this subsection, we conduct more experiments using IMPR to see how it performs with non-dedicated strategy by setting $SH = 1$. We also include more runs to cover workloads from 1.1 to 2.0, where feasible solutions should exist in a non-dedicated strategy. This is because even though each service is being allocated at least eight or more ranges, these ranges are not exclusively assigned to the services. Hence, more than one service may be allocated

to the same range, and therefore we are not limited by having only 160 ranges in the entire yard. Assuming yard utilisation is defined as the percentage of the yard space being occupied by containers, Table 4.5 summarises the corresponding yard utilisation at various workloads.

Table 4.6: Summary of running times of the models

	Model = IMPR			
Sharing	SH=0	SH=1	SH=0	SH=1
Number of RMGs	3 RMG	3 RMG	1 RMG	1 RMG
Workload	(D)	(E)	(F)	(G)
0.1	8.19	2.95	8.4	3.3
0.2	20.69	2.94	21.3	2.9
0.3	18.69	2.91	23.5	2.8
0.4	22.44	2.91	22.8	2.9
0.5	29.34	3.03	35.3	3.4
0.6	36.66	3.30	390.1	5.6
0.7	107.30	4.33	1165.6	10.6
0.8	703.19	18.84	*	243.1
0.9	*	155.77	*	323.9
1.0	*	180.18	*	503.8
1.1	*	259.31	*	841.7
1.2	*	322.43	*	1094.3
1.3	*	403.24	*	1271.3
1.4	*	443.73	*	1820.0
1.5	*	503.14	*	5001.9
1.6	*	901.11	*	*
1.7	*	1514.09	*	*
1.8	*	1533.50	*	*
1.9	*	1921.17	*	*
2.0	*	2339.06	*	*

* run did not complete

Using the same scenario, we run IMPR with $SH = 1$ for workloads from 0.1 to 2.0. The results are shown in column (E) in Table 4.6. We also create another set of runs with a slightly different scenario, where number of RMG per yard block is set to 1

Table 4.7: Summary of the RMG utilisation at different workload with one and three RMGs per block

	Total activities in yard at any time period			RMG utilization	
Workload	Discharge	Load	Total	3 RMG	1 RMG
0.1	60	90	150	2%	6%
0.2	120	180	300	4%	13%
0.3	182	273	455	6%	19%
0.4	242	363	605	8%	25%
0.5	304	456	760	11%	32%
0.6	364	546	910	13%	38%
0.7	426	639	1065	15%	44%
0.8	486	729	1215	17%	51%
0.9	548	822	1370	19%	57%
1.0	608	912	1520	21%	63%
1.1	670	1005	1675	23%	70%
1.2	730	1095	1825	25%	76%
1.3	792	1188	1980	28%	83%
1.4	852	1278	2130	30%	89%
1.5	914	1371	2285	32%	95%
1.6	974	1461	2435	34%	101%
1.7	1036	1554	2590	36%	108%
1.8	1096	1644	2740	38%	114%
1.9	1158	1737	2895	40%	121%
2.0	1218	1827	3045	42%	127%

instead of 3. IMPR is then run with $SH = 0$ and $SH = 1$ and the results are tabulated in columns (F) and (G) in Table 4.6, respectively. We note that all the completed runs produce zero objective values.

First, we compare the run times as tabulated in Table 4.6. Comparing columns (D) versus (E), and columns (F) versus (G), we see that run the time for the model for dedicated strategy ($SH = 0$) is very long (can be more than 100 times slower) compared to the runs for non-dedicated strategy ($SH = 1$) at the same workload. This

is because when $SH = 1$, the constraints are less tightly bounded, and hence it is able to achieve shorter run time (see Constraint 4.7). Next, comparing columns (E) and (G), which are both runs for the non-dedicated strategy ($SH = 1$), the case with one RMG per block can run up to 10 times slower than the case with three RMGs per block. This is because for the case with one RMG per block, the RMG utilisation is higher than that of the three RMGs case with the same workload (see Table 4.7), and hence the RMG contention constraints are more tightly bounded (see Constraint 4.9). The experiments show that the model's run time is very sensitive to the tightness of the constraints. We conclude that this approach is more suitable for the non-dedicated strategy as the run time for dedicated strategy is too long.

Next, we try to explain some of the run time behaviour. We note that there does not exist solutions that have zero RMG contention violations for the case of one RMGs with workload above 1.5. This is because the RMG utilisation has exceeded 100% (See Table 4.7). Therefore, we expect feasible solutions for workload 1.6 to 2.0 with non-zero violations (objective value), but the model did not find a solution within the time-out limit of eight hours. An empirical observation is that as long as an optimum solution exists with non-zero violation (non-zero objective value), the model will not terminate within the time-out limit of eight hours. This could possibly be an explanation for the case of one RMG per block on a dedicated strategy ($SH = 0$) and at a workload of 0.8. The model did not terminate within the time-out limit. We are not certain whether a feasible solution with zero violation exists for this case or not. However, we do know that a feasible solution exists since a solution exists for the case with three RMGs at this workload. We also note that in this case, since a service loads 243 $\times$ 20-footers and

243 $\times$ 40-footers, the total number of ranges allocated to a service is at least seven. In a typical time period, as there will be three services loading concurrently, hence at least 21 ranges will have their containers loaded concurrently. Since the entire yard has only 20 yard blocks, at least one of the yard blocks will have two of its ranges involved in loading (by pigeon hole principle). We think that these blocks are likely to have violated the RMG contention, resulting in non-zero objective value. Hence, we further conclude that this model is suitable for scenarios where feasibility (no violation) is sought, and does not work for scenarios where violations exist and need to be minimised.

Lastly, a note to compare the goodness of the two strategies, we see that non-dedicated strategy is better than dedicated strategy. This is because non-dedicated strategy produces solutions at much higher workload, and results in better yard utilisation, while keeping the number of violations at zero (objective value is zero). In the case with three RMGs per block, a solution with 95% yard utilisation (at workoad 2.0) is generated, while in the case with one RMG per block, a solution with 75% yard utilisation (at workload 1.5) is generated. Correspondingly, for the case with dedicated strategy, it generates solutions with yard utilisation at 38% and 33.3%, respectively. Hence, non-dedicated strategy has double the yard utilisation of dedicated strategy.

4.3 Approach 2: Heuristics algorithm - for the case of consolidated and dedicated strategy

We note from earlier experiments that the run time for the IMPR with dedicated strategy ($SH = 0$) can be very long for cases of high workload. In this section, we present a

heuristic algorithm for the dedicated strategy.

Earlier experiments show that run time increases with increased tightness of the constraints ($SH = 0$ to $SH = 1$, and three RMGs per block reduce to one RMG per block). We further explore this property by changing the thresholds for the activity concentration (MA_g), which will affect the tightness of the Constraint 4.8. We conduct a new set of experiments on the same scenario as Section 4.2.4 using IMPR, three RMGs per block, and $SH = 0$. We vary MA_g from 50 to 120, and workload from 0.05 to 0.85 in steps of 0.05. The run times in seconds are tabulated in Table 4.8. The experiments confirm that as we reduce MA_g and increase workload, the problem gets more constrained and hence the run time increases. There are many cases where the model did not terminate within the time-out limit of eight hours.

We note that for the case of dedicated strategy, only one service can be assigned to a range, i.e. $\sum_{i \in \mathcal{V}} u_{ir} \leq 1, \ \forall r \in \mathcal{R}$. It means that the solution lies within the solution space of the assignment vector $A = (a_1, a_2, \ldots, a_r, \ldots, a_{|\mathcal{R}|})$, where $a_r = i$ if and only if service i is assigned to range r, i.e. $u_{ir} = 1$, and $a_r = 0$ if and only if range r is not assigned to any service, i.e. $u_{ir} = 0 \ \forall i \in \mathcal{V}$. Hence, u_{ir} can be derived given A, and vice versa. Next, we modify the model IMPR to read in u_{ir} as an input instead of solving it as a decision variable, i.e., we fix the assignment of service to the yard ranges, and we call the modified model as IMPR-u. We then perform a search based algorithm on the assignment of services to the range, i.e., find the vector $A = (a_1, a_2, \ldots, a_r, \ldots, a_{|\mathcal{R}|})$ that gives the lowest objective value in the model IMPR-u.

Table 4.8: Summary of the running times of IMPR

Workload	MA_g							
	50	60	70	80	90	100	110	120
0.05	4.19	4.08	4.09	4.09	4.08	4.09	4.09	4.16
0.10	8.03	4.38	8.52	8.44	8.98	9.08	4.39	4.41
0.15	8.86	9.02	9.25	4.39	9.20	8.95	4.44	8.59
0.20	10.94	18.49	10.23	17.47	9.64	17.44	9.14	9.72
0.25	24.17	19.78	19.11	18.30	18.88	10.44	17.64	16.52
0.30	32.81	26.02	24.56	20.92	18.66	21.33	20.77	19.19
0.35	43.33	36.66	30.19	26.22	23.34	22.74	25.67	22.75
0.40	260.50	55.91	35.74	28.14	28.19	26.25	23.06	25.08
0.45	374.60	254.58	71.80	233.47	30.55	37.67	25.86	28.27
0.50	584.15	373.00	401.11	267.21	33.77	79.28	39.67	31.16
0.55	1028.18	729.73	419.11	324.56	226.10	35.14	106.13	66.30
0.60	*	762.66	328.39	345.50	225.39	215.77	54.38	240.53
0.65	*	9818.20	1103.10	470.57	399.75	264.29	220.74	119.22
0.70	*	5416.69	2026.22	1057.74	820.92	663.98	99.28	281.69
0.75	*	15217.50	4205.56	1638.48	991.18	814.46	382.69	386.26
0.80	*	*	*	*	*	1178.87	750.61	730.36
0.85	*	*	*	*	*	*	1131.21	749.66

4.3.1 Loading Separation Assignment and Hill Climbing local search (LSA-HC)

The proposed heuristic algorithm is made up of two parts, namely a greedy algorithm to find an initial vector A, and a hill climbing local search to improve the vector A. The objective value of A is computed as follows. We first convert A into its equivalent u_{ir}s. We then solve IMPR-u with the inputs u_{ir}s and then obtain its objective value.

We use a Loading Separation Assignment heuristics (LSA) to find an initial solution A. The motivation of LSA is that for the case of consolidated and dedicated strategy,

retrieval activities in the range happen in concentration during the time periods when the service is loading, and storage activities in the range happen in smaller numbers over larger numbers of time periods, as containers received for a service will arrive over multiple time periods. Hence, the activity concentration violations are largely, but not always, contributed by the sum of retrieval activities of two neighbouring ranges happening at the same time that exceeds the activity thresholds. Therefore, the heuristic tries to ensure that for every neighbouring pair of ranges, they should not be assigned to services that have common loading time periods. Note that this heuristic works for consolidated strategy, where the yard range are sufficiently large (in our example five slots), so that activities happening in two non consecutive ranges do not interfere with each other as they are separated by at least one yard range in between.

The heuristic is described as follows. Let $\mathcal{V}'$ be an ordered list of services obtained from $\mathcal{V}$. For convenience of notation, we let $i.next$ indicate the service that comes after i in the ordered list $\mathcal{V}'$. If i is the last item, then $i.next$ gives the first item in the list instead. We create the ordered list $\mathcal{V}'$ from $\mathcal{V}$ such that every consecutive pair of services i and $i.next$ in $\mathcal{V}'$ has no overlapping loading period, i.e., $\mathcal{L}_i \cap \mathcal{L}_{i.next} = \emptyset$. Since most container terminals only have a small number of services, usually in the 10's, and loading periods for each service are usually about one to three time periods out of the 21 time periods in the entire time window, we reason that the construction of $\mathcal{V}'$ can be done easily by hand. For each range $r \in \mathcal{R}$ sorted in order of slot number, we greedily assign the next service in $\mathcal{V}'$ to r in a round robin fashion. The pseudo code for the LSA algorithm is given in Figure 4.3.

Next the hill climbing local search is done as follows. First, the objective value of

```
1: LSA Algorithm
2:
3: Input: V' = ordered list of services
4: Input: R = set of ranges
5:
6: let i = first item in V'
7: for all r ∈ R in ascending order of slot number do
8:     assign i to r
9:     i = i.next
10: end for
```

Figure 4.3: Pseudo code for LSA Algorithm.

the current solution $A = (a_1, a_2, \ldots, a_r, \ldots, a_{|\mathcal{R}|})$ is computed by running IMPR-u. We then compute $N(A)$, the neighbourhood set of A as follows. A neighbour of A is an assignment vector with all the assignments equal to the assignments in A except that one of the a_i is replaced with other services in $\mathcal{V}$ other than the original value of a_i. This means that one of the range r is re-assigned to another service. Hence each a_i can be replaced with $|\mathcal{V}| - 1$ possible value, and there are $|\mathcal{R}|$ number of a_i, hence the size of $N(A)$ is $(|\mathcal{V}| - 1) \times |\mathcal{R}|$. All the neighbours in $N(A)$ are evaluated for their objective values using IMPR-u. If all the neighbours give higher objective values than the current solution A, then the local search terminates with A as the best solution. Otherwise the neighbour with the lowest objective value will be chosen to be the next best solution, and it takes the role as the current solution, and the local search repeats. The flowchart of the heuristic is given in Figure 4.4 and the detailed pseudo code of the algorithm is given in Figure 4.5. Note that for the purpose of speeding up execution time, we terminate the search whenever a solution with zero objective value (optimum) is found (see Line 8 and Line 18 in Figure 4.5).

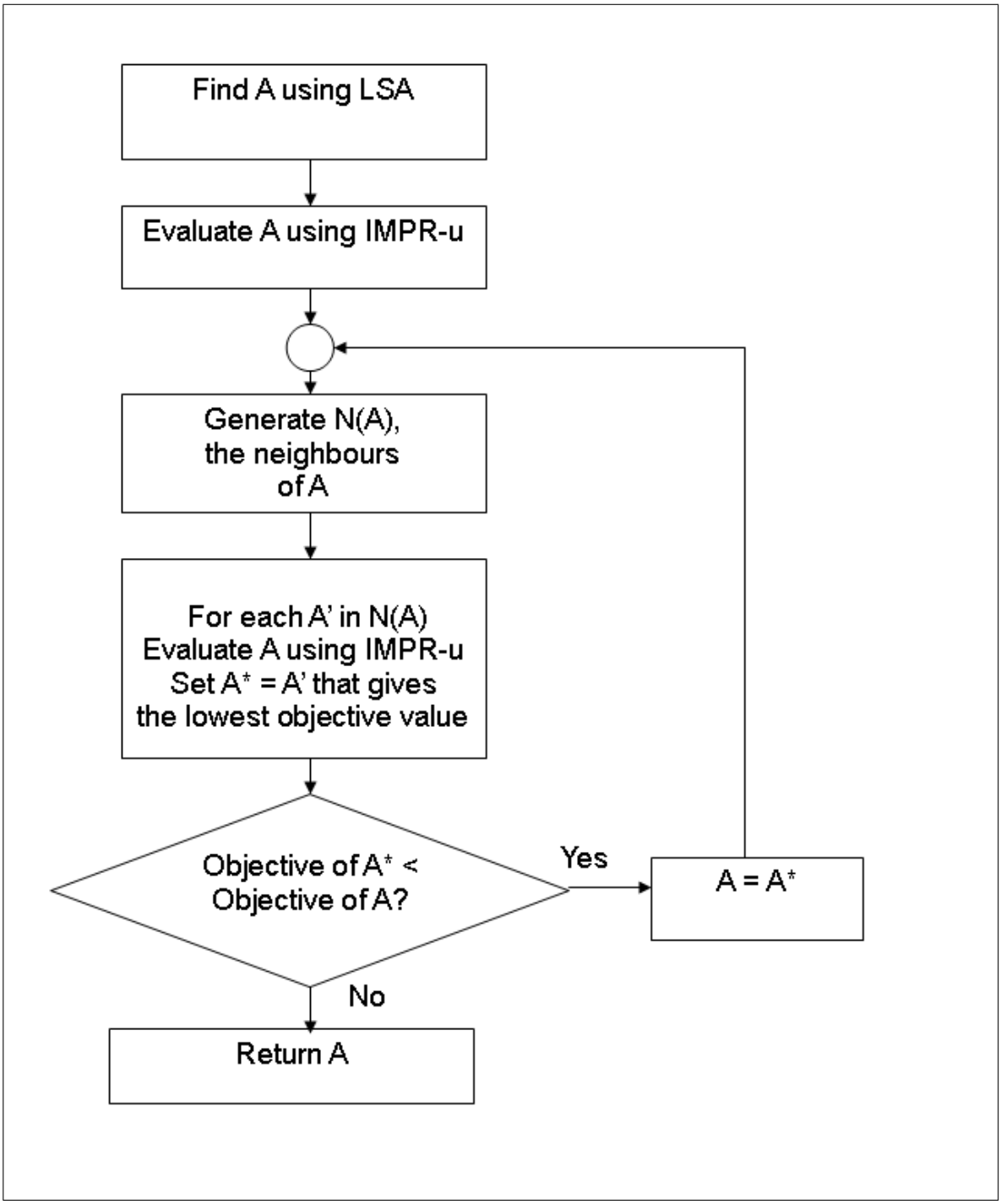


Figure 4.4: Flowchart for LSA-HC Algorithm.

4.3.2 Experimental results

We conduct another set of experiments with the LSA-HC algorithm using the same settings as those used to generate results in Table 4.8 for the model IMPR. The run

```
1: LSA-HC Algorithm
2:
3: let A = initial solution from LSA algorithm
4:
5: while not converge do
6:    best = Objective value of A by running IMPR-u
7:    if best = 0 then
8:       return A as the optimum solution
9:    end if
10:   compute neighbours of A = N(A)
11:   set A* = null
12:   for all A' in N(A) do
13:      obj = Objective value of A' by running IMPR-u)
14:      if obj < best then
15:         A* = A'
16:         best = obj
17:         if best = 0 then
18:            return A* as optimum solution
19:         end if
20:      end if
21:   end for
22:   if A* = null then
23:      converge and return A as the optimal solution
24:   else
25:      set A = A*
26:   end if
27: end while
```

Figure 4.5: Pseudo code for LSA-HC Algorithm.

time in seconds for this set of experiments is tabulated in Table 4.9.

To ease comparison of the run time, Table 4.10 shows the run time of LSA-HC in percentage of the run time of IMPR with the corresponding workload and MA_g. When the percentage value for a specific setting of MA_g and workload is more than 100%, LSA-HC runs slower than IMPR for this specific setting, and when the value is less than 100%, LSA-HC runs faster than IMPR for this specific setting. For the case when IMPR did not terminate, “NA” is shown in the table. We can see that for a high workload and smaller value of MA_g, LSA-HC is taking only a small percentage of the run time

Table 4.9: Summary of the running times of LSA-HC

Workload	MA_g							
	50	60	70	80	90	100	110	120
0.05	9.95	10.33	10.52	10.70	10.67	10.53	9.86	9.92
0.10	9.92	9.75	9.77	9.77	9.73	9.91	9.89	9.89
0.15	10.05	9.73	9.75	9.75	9.75	9.92	9.95	9.95
0.20	9.81	9.78	9.83	9.77	9.77	9.92	9.95	9.95
0.25	9.86	9.80	9.81	9.80	9.80	9.97	9.97	9.94
0.30	9.91	9.81	9.97	9.81	9.83	10.00	10.02	9.97
0.35	9.89	9.81	9.83	9.83	9.83	10.00	10.00	10.00
0.40	9.89	9.81	10.13	9.81	9.81	9.97	10.03	9.98
0.45	9.88	9.80	10.19	9.80	9.78	9.98	10.00	9.97
0.50	9.92	9.81	9.81	9.78	9.81	9.98	10.02	10.00
0.55	33.44	9.83	9.81	9.81	9.80	10.00	10.03	9.98
0.60	4721.77	9.92	9.98	9.89	9.88	10.03	9.97	10.08
0.65	21657.60	29.64	9.92	9.89	9.89	10.22	10.06	10.05
0.70	22033.60	29.48	9.94	9.81	9.91	10.09	10.03	9.98
0.75	27569.40	10983.00	30.73	9.94	9.91	10.06	10.05	10.27
0.80	30994.20	12223.40	21.16	10.31	10.23	10.27	10.22	10.14
0.85	269389.00	23277.40	12115.10	21.02	10.20	10.17	10.14	10.05

compared to IMPR, and in some cases, less than 1%. There are cases where LSA-HC runs slower (more than 100%), specifically for workloads less than or equal 0.2, but that is not a concern as no container terminal should be operating at such low workload (equivalent to less than 10% of the yard space utilised).

Note that in the model, the objective value is non-negative, and hence when it is zero, the solution must be at optimum. The objective value of the respective runs for LSA-HC are given in Table 4.11. For convenience of discussion, we shaded the cells in Table 4.10 if the objective value given by LSA-HC is non-zero, and non-shaded cells means that LSA-HC generated a solution with zero objective value. For those cells

Table 4.10: Running times of LSA-HC compared to IMPR in percentage

	MA_g							
Workload	50	60	70	80	90	100	110	120
0.05	237.7%	253.3%	256.9%	261.4%	261.7%	257.2%	240.8%	238.7%
0.10	123.5%	222.9%	114.7%	115.7%	108.3%	109.1%	225.2%	224.5%
0.15	113.4%	108.0%	105.4%	222.0%	105.9%	110.8%	224.3%	115.8%
0.20	89.7%	52.9%	96.0%	55.9%	101.3%	56.9%	108.9%	102.4%
0.25	40.8%	49.5%	51.3%	53.5%	51.9%	95.5%	56.5%	60.2%
0.30	30.2%	37.7%	40.6%	46.9%	52.7%	46.9%	48.2%	51.9%
0.35	22.8%	26.8%	32.6%	37.5%	42.1%	44.0%	39.0%	44.0%
0.40	3.8%	17.6%	28.3%	34.9%	34.8%	38.0%	43.5%	39.8%
0.45	2.6%	3.8%	14.2%	4.2%	32.0%	26.5%	38.7%	35.3%
0.50	1.7%	2.6%	2.4%	3.7%	29.1%	12.6%	25.2%	32.1%
0.55	3.3%	1.3%	2.3%	3.0%	4.3%	28.5%	9.5%	15.1%
0.60	NA	1.3%	3.0%	2.9%	4.4%	4.6%	18.3%	4.2%
0.65	NA	0.3%	0.9%	2.1%	2.5%	3.9%	4.6%	8.4%
0.70	NA	0.5%	0.5%	0.9%	1.2%	1.5%	10.1%	3.5%
0.75	NA	72.2%	0.7%	0.6%	1.0%	1.2%	2.6%	2.7%
0.80	NA	NA	NA	NA	NA	0.9%	1.4%	1.4%
0.85	NA	NA	NA	NA	NA	NA	0.9%	1.3%

in Table 4.10 that are non shaded, we are absolutely sure that the LSA-HC found the optimum solution at a fraction of the run time needed by the model IMPR for the exact same problem. There are also cases where IMPR is not able to terminate within the time-out limit of eight hours (example $MA_g = 100$ and workload = 0.85), and LSA-HC is able to find the optimum solution (non-shaded cells with “NA” shown). For the remaining cases where IMPR is not able to terminate, LSA-HC is able to find a solution with non-zero objective (shaded cells with “NA” shown), but we cannot be certain that LSA-HC finds the optimum. This is because we do not know what is the true optimum objective value in these cases as the IMPR is not able to terminate. The only exception

Table 4.11: Summary of the objective values of LSA-HC

Workload	MA_g							
	50	60	70	80	90	100	110	120
0.05	0	0	0	0	0	0	0	0
0.10	0	0	0	0	0	0	0	0
0.15	0	0	0	0	0	0	0	0
0.20	0	0	0	0	0	0	0	0
0.25	0	0	0	0	0	0	0	0
0.30	0	0	0	0	0	0	0	0
0.35	0	0	0	0	0	0	0	0
0.40	0	0	0	0	0	0	0	0
0.45	0	0	0	0	0	0	0	0
0.50	0	0	0	0	0	0	0	0
0.55	0	0	0	0	0	0	0	0
0.60	18	0	0	0	0	0	0	0
0.65	120	0	0	0	0	0	0	0
0.70	252	0	0	0	0	0	0	0
0.75	366	96	0	0	0	0	0	0
0.80	474	204	0	0	0	0	0	0
0.85	686	318	88	0	0	0	0	0

to the above is when $MA_g = 60$ and workload $= 0.75$. In this case, IMPR terminates in 4.2 hours with an optimum solution with zero objective value, but LSA-HC terminates in three hours (faster) with objective value 96. This illustrates a case where the heuristic may fail to find the optimum.

To conclude, the experiments show that for some cases, LSA-HC is able to find the optimum solution faster than IMPR, and for some cases, LSA-HC is able to find the optimum solution where IMPR is not able to. For the rest of the cases, LSA-HC is able to generate a feasible solution (which may not be optimum), but IMPR is not able to. We find one case where IMPR finds an optimum, while LSA-HC does not, but nonetheless,

LSA-HC finds the solution at a much shorter time. We conclude that LSA-HC is a good heuristic to solve for the cases with dedicated strategy.

4.4 Summary

Having the motivation of designing search engines to discover the best yard plan strategy for a container terminal given its scenario, we require the generic model presented in Chapter 3 to be solved in a reasonable amount of time. As the complexity of the problem depends on the strategy, we propose having different solution approaches for different strategies. Instead of covering all yard plan strategies, we present solution approaches to solve the problem for two strategies, namely, 1) consolidated and dedicated strategy, and 2) consolidated and non-dedicated strategy. Two solutions approaches are presented, namely solving as a mathematical problem using CPLEX, and a heuristic algorithm. With re-modelling of the original mathematical model to become the model IMPR, experiments show that IMPR reduces the run time tremendously. Empirically, it is only able to find the optimum solution with zero violations if they exist. The run time of the model also highly depends on the tightness of the constraints. For the case of dedicated strategy, the model runs very much slower than the case with non-dedicated strategy. We conclude that IMPR is a good approach for the non-dedicated strategy. As a side note in the perspective of quality of the solution, the experiments also show that non-dedicated strategy is a better strategy as it produces solutions with zero violations and with higher yard utilisation.

In the second approach, we propose a heuristic algorithm LSA-HC. The experiments

show that the run time of the model for high workload and lower MA_g thresholds is faster than that of IMPR for the same problem instance, and in many cases less than 1% of the IMPR run time. It also find solutions in cases where IMPR is not able to, and some of them are optimum. However, being a heuristic algorithm, we find one case where LSA-HC did not find the optimum solution, but it terminated at a shorter time compared to IMPR for the same case. We conclude that LSA-HC is a good heuristic for solving the cases with the dedicated strategy.

In the next chapter, we explore solving the problem when the vessel arrival schedule is uncertain, and we introduce the notion of nimble yard plan template, where we are interested in finding a solution that can be changed easily when uncertainty is revealed.

Chapter 5

Yard Plan Template with Uncertainty - Nimble Optimisation

After a terminal has decided to adopt a yard plan strategy, a yard plan template can then be generated and implemented according to the strategy. However, uncertainty of events prevails in reality, and terminals usually modify and adapt their yard plan template to the changes, which may not be optimum. While the community for robust optimisation may suggest developing a robust plan – a plan that remain feasible subjected to uncertainty, we will like to have a plan that can be changed easily to an optimal solution after the uncertainty is revealed. We call this the **nimble yard plan template problem**. This lead us to a generic problem formulation, where in a situation when input parameters are uncertain, we like to find a optimum solution before the uncertainty is revealed, and this solution can be changed easily after the uncertainty is revealed. We call this a **nimble optimisation** problem. This is a relatively new area with only some related works found in the literature. By defining a generic problem formulation, it allows the non-container terminal operation research community to also explore other problems that can be classified into this class of problems, and hence allowing a bigger research

community to be exposed to this problem. It is interesting to note that this problem has never been addressed in the literature, except some related works where we show that our formulation is a generalisation of these models (See Appendix A).

The chapter is organised as follows. In Section 5.1, we study the nimble optimisation problem. We present a short literature survey on some related works, and then present the formulation of the generic nimble optimisation formulation. We also propose a solution approach to solve the problem. Next in Section 5.2, we apply the nimble optimisation problem into the yard plan template problem. Three variations of the nimble yard plan template problem are presented, and solution approaches and experimental results are discussed. We summarise the chapter in Section 5.3.

5.1 Nimble Optimisation (Nimo)

Uncertainty prevails in real life, and decisions are often made before the uncertainty is revealed. Uncertainties are revealed in stages (event triggers), and usually decision makers are able to change their decisions in every stage. There are generally two schools of study in this area. First, the uncertainty in question follows a known probability distribution, and the objective at every decision stage is to optimise the expected objective value. This generally falls under the stochastic programming approach. The other school of study assumes the probability distribution of the uncertainty is unknown and instead it is defined by an uncertainty set. This falls under the area of robust optimisation.

Nimble optimisation does not fall into any of these groups. A good analogy will be the

chameleon. A chameleon is able to change its colour to optimally match its changing environment. Nimble optimisation (Nimo), in layman terms, is a solution where it is like a chameleon, able to change optimally to match its changing environment. With many practical applications in real life problem such as in the yard template problem, we are motivated to look at a generic nimble optimisation formulation.

In Section 5.1.1, we give a short literature review on related topics, and in Section 5.1.2, we define our formulation of Nimo. We then show a possible solution architecture for Nimo in Section 5.1.3.

5.1.1 Literature survey

Optimisation models taking uncertainty into considerations have been gaining interest among researchers. More importantly, they provide more practical solutions for industries since it is hard to find a truly deterministic problem in the real world. There are two perspectives to view uncertainty. The first approach is *distributional*, where a probability distribution is associated with the parameters of the problems. The second one is the *robust* approach where it is only possible to associate the uncertainty to a set of values. The distributional approach mainly covers areas of Stochastic Programming and Markov Decision Process, which we will not dwell into. Instead we are interested in the robust approach, and we review some of the major works done.

In one of the earliest papers in robust optimisation, Soyster (Soyster 1973) considers a mathematical model where there are column wise uncertainties, such that each column of the matrix belongs to a convex uncertainty set. The model solves for a solution that remains feasible in all possible realisations of the uncertainty, making the approach

very conservative. Motivated by an engineering problem on truss design, Ben-Tal and Nemirovski (Ben-Tal & Nemirovski 1997) then propose a robust approach with an uncertain ellipsoidal set, which is then shown to be equivalent to a semi-definite programme. Ben-Tal and Nemirovski (Ben-Tal & Nemirovski 1998, Ben-Tal & Nemirovski 1999) further generalise the problem and define the terms *robust counterpart* and *uncertainty set*, and describe the notion of *optimality* in an uncertain optimisation problem. They define a *uncertain* mathematical programme as one where some of the data of the problem is known to be contained in a range or set. Hence some parameters are not known in advance, which makes it not possible to solve as it is. Its robust counterpart is a solvable mathematical programme of the same problem, whose solution is robust regardless of the realisation of the random parameters. Ben-Tal and Nemirovski (Ben-Tal & Nemirovski 2000) later study the robust problem with a new uncertain ellipsoid set that comes with a parameter to control the degree of conservativeness of the model. They show that the problem can be formulated as a second order cone programme. The approach by Soyster is conservative but retains a linear structure, whereas the approach by Ben-Tal and Nemirovski is non-linear but allows greater flexibility. Bertsimas and Sim (Bertsimas & Sim 2004) introduce an alternative model with the flexibility to adjust robustness while keeping linearity. They formulate a robust counterpart, such that if the number of perturbed parameters is within a pre-determined threshold, they prove that the robust solution will be feasible deterministically, and if there are more changes, the robust solution will be feasible with high probability.

Some papers model their problem as a 2-stage decision making problem, which is similar to Nimo. Early work on this was presented by Ben-Tal et al. (Ben-Tal, et al. 2004).

They extend the robust framework by introducing the concept of *adjustable* and *non-adjustable* variables. Their model separates the decision variables into two types, those made in stage 1 before the uncertainty is revealed (non-adjustable), and those made in stage 2, after the uncertainty is revealed (adjustable). However, their objective function is only dependent on the non-adjustable variables. For tractability, they further propose the *affinely adjustable robust solution*, in which the *adjustable* variable is a linear function of the uncertain data set. Ben-Tal et al. (Ben-Tal, et al. 2006) subsequently provide an extension to the adjustable robust counterpart where they do not assume that the uncertainty set is bounded. Chen and Zhang in (Chen & Zhang 2009) propose the extended affinely adjustable robust counterpart. In this model, instead of allowing the adjustable variables to be affinely dependent on the uncertainty parameters, they also allow for additional affine dependency on other parameters. Bertsimas and Caramanis (Bertsimas & Caramanis 2010) introduce the notion of *complete adaptability*. For tractability, he further proposes the k-adaptability problem in which the decision-maker chooses one of the k second stage solutions after observing the realisation of the uncertainty. This is then shown to be equivalent to a bilinear optimisation problem and is solved exactly for the case of $k = 2$. Liebchen et al. (Liebchen, et al. 2009) propose the concept of recoverable robustness and discuss a special case, the linear recovery robust programme. Linear robust programme is very similar to nimble optimisation, where stage 2 decision is a recovery of the decision made in stage 1, and a penalty cost is associated to the difference between the stage 1 and stage 2 decision. However, this cost is not optimised in the objective, and instead the sum is bounded by a budget in the constraints. Hence the objective only depends on the stage 1 decision, while the

various possibilities of stage 2 decisions are constraints to the model.

Other than the paper by Liebchen et al. (Liebchen et al. 2009), we did not see any paper that measures the cost of changing the stage 1 solution to the recovery solution in stage 2. However, they did not attempt to minimise this cost in the model. We present in the next section a mathematical definition of nimble optimisation, where we solve for a stage 1 decision, such that it minimises both the maximum cost of all the possible recovery options that could be taken after the uncertainty is revealed, plus the cost of changing the decision already made in stage 1 to that of a stage 2 decision.

5.1.2 Defining Nimo

Consider the following static linear optimisation problem

$$\begin{aligned} &\underset{x \in \Re^n}{\text{Minimise}} \;\; c^T x \\ &s.t. \;\; Ax \geq b \end{aligned} \tag{5.1}$$

We let $\omega \in \Omega$ represent the uncertainty in the uncertainty set Ω, and $A(\omega)$ and $b(\omega)$ be the adjusted parameters with the revealed uncertainty ω. Ω, in most cases, cannot be determined theoretically (unless they can be derived from physical laws). It has to be determined empirically via experiments, historical data collection, forecasting models or expert opinions. The cost vector c of applying the final decision x_1 is known and assumed to be independent of the uncertainty without loss of generality. If c depends on the uncertainty, then we add a new constraint $z \geq c^T x_1$ into the constraints and replace the objective with z.

Uncertainty usually unfolds as a sequence of events, and decisions made in the face of uncertainty have to take into consideration the sequence of uncertain events occurrences. However, modelling an infinite sequence of events will be highly complex, and hence a planning time horizon has to be determined. Within the time horizon, one could change their decision on every event occurrence or in stages after accumulation of a number of occurrences of uncertain events. We assume that decisions are made in stages and for simplicity (and also a limitation to the model), we assume a two stage decision process, where a stage 1 decision is made before the uncertainty is revealed, and a stage 2 decision is made at the end of the time horizon, which is a recovery after observing all the uncertain events occurrences.

Defining the notation for the model, in stage 1, a decision x_0 is made before the uncertainty ω is revealed. After the uncertainty is revealed, we make a stage 2 decision x_1, which is a recovery of the stage 1 decision. x_0 and x_1 are of the same dimension, i.e. $x_0 \in \Re^n$ and $x_1 \in \Re^n$, as x_1 is the recovery decision that replaces the original decision x_0. If $x_0 = x_1$, then there is no recovery after the uncertainty is revealed.

We consider a recovery algorithm after the uncertainty ω is revealed. If the recovery does not respect the stage 1 decision that is made before ω is revealed, the mathematical problem is represented as follows:

$$\begin{aligned} &\underset{x_1 \in \Re^n}{\text{Minimise}} \;\; c^T x_1 \\ &s.t. \;\; A(\omega) x_1 \geq b(\omega) \end{aligned} \tag{5.2}$$

However, there is also a cost of transforming the stage 1 decision to the stage 2

decision, represented by $\Delta(x_0, x_1)$, and this could be set to ∞ if it is impossible to recover from x_0 to x_1. We further assume that the cost of recovery can be measured in a unit that is linear to the cost function of the original problem, and without loss of generality, we assume that we can define $\Delta(x_0, x_1)$ using the same unit of measurement as the original problem.

In the **Nimble Optimisation** (**Nimo**) problem, we want to find a x_0 in stage 1 before the uncertainty ω is revealed, such that we minimise among all $\omega \in \Omega$, the *worst case* cost of transforming x_0 to x_1 plus the cost of x_1 after the uncertainty is revealed. Hence, regardless of how the uncertainty is revealed, x_0 is agile enough to be changed to a final decision x_1 while the total cost is bounded by the objective value solved during stage 1. Nimo is then defined mathematically as

$$\underset{x_0 \in \Re^n}{\text{Minimise}} \quad \max_{\omega \in \Omega} \quad \min_{x_1(\omega) \in \Re^n} \left[\begin{array}{l} c^T x_1(\omega) + \Delta(x_0, x_1(\omega)) \\ \text{subject to} \\ A(\omega) x_1(\omega) \geq b(\omega) \\ A(\omega_0) x_0 \geq b(\omega_0) \;\; \exists\, \omega_0 \in \Omega \end{array} \right] \qquad \text{(Model V.I)}$$

To explain the min-max-min structure, we start from the inner most structure. The innermost structure is a minimisation problem of finding the optimum $x_1(\omega)$ given a realisation of the uncertainty ω as well as the decision made in stage 1 x_0. That is, the innermost structure defines the optimum recovery problem given the stage 1 decision x_0 and a realised ω. Next, among all possible realisations of ω, the maximization problem

finds the ω that gives largest (or worst) optimum recovered solution. Then, finally, the nimble optimisation problem finds the optimum x_0 such that the worst case outcome of the recovery is minimised.

Note that in the formulation, we write $x_1(\omega)$ instead of x_1 as $x_1(\omega)$, at the point of making decision in stage 1, is not deterministic, but a mapping depending on ω, i.e. $x_1(\omega) \vdash \Re^n$. Hence, the solution generated after solving the stage 1 problem is the solution x_0 and an implicit mapping function $x_1(\omega)$. Note also that the formulation caters for the case where we also assume that the decision in stage 1, x_0, should also be feasible under a nominal setting of the uncertainty ω_0 if it exists in the model.

5.1.3 Solution approach

To solve this problem, one could possibly work on special cases of the problem, by making specific assumptions of the problem structure (for e.g. assume a fixed number of recovery solutions), properties of the recovery solution x_1 (for e.g. assume x_1 is affinely dependent on ω), or the cost of changing x_0 to x_1 (for e.g. check for feasibility with zero being feasible and ∞ being infeasible). Specific solution techniques can then be explored for various special cases. However, we like to explore a general method for solving this class of problems. We propose a simulation guided local search architecture, where uncertainties are handled as part of the simulation process. We think that this is a good architecture, and the reasons are as follows.

Firstly, many real life problem do not have a very structured definition of their uncertainty. Chances are that it cannot be defined by a known probability distribution. However, in simulation, as long we can generate a cumulative distribution function (either

via histogram from historical data, or hand plotted based on expert opinion or forecast), the scenarios can be randomly generated to evaluate the solution. Hence the approach is not limited in this aspect.

Next, while the usual range of values for the uncertain parameters could be known, the true range can be very wide with extreme points occurring with very low probability. These very rare occurrences of extreme values are usually ignored in analysis, but some might cause devastating consequence to the solution. For example a major earth quake in Japan in March 2011 disrupted supply chains, and caused material flow to be disrupted. In another example, a financial crisis struck in late 2008, that caught many people off guard, caused many shipping routes and schedules to change. Many ships were left empty in the anchorage due to over supply of ships, and those still in service adopted slow steaming (reduce travelling speed) to save on fuel cost. These rare but high impact events are termed as "Black Swan events" by Taleb (Taleb 2007). These black swan events can be evaluated if the simulation framework generates such extreme points intentionally to simulate the extreme in case they do happen.

Lastly, as the problem can be of very high complexity, the solution architecture allows us to exploit distributed computing technology to speed up the runs. This is elaborated further with the details of the framework explained below.

The algorithm flow chart is given in Figure 5.1. It follows a standard hill climbing local search routine, where an initial solution x_0 is generated, followed by the hill climbing iterations. In each iteration, the neighbourhood set $N(x_0)$ is generated, and for each neighbour x'_0, it is evaluated for its objective value, and the algorithm chooses the best neighbour as the next best solution, and terminates when no better solution is found.

The simulation comes into play in the evaluation routine, where K number of uncertainty ω are generated, simulating K scenarios, and for each scenario, the recovery algorithm is executed to see how well the solution can be recovered. The largest (worst case) objective value is returned.

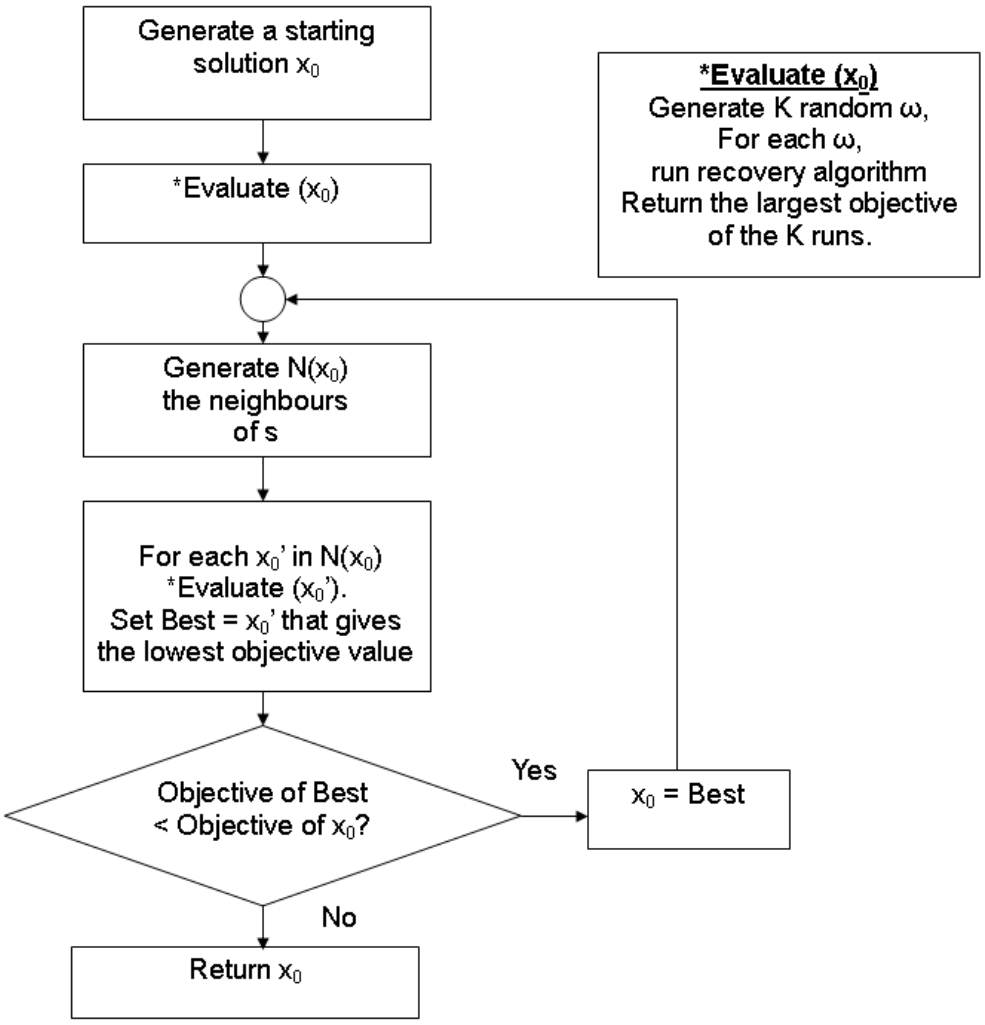


Figure 5.1: Simulated guided local search flow chart for solving Nimo

We note that the approach naturally decomposes the computation into well defined

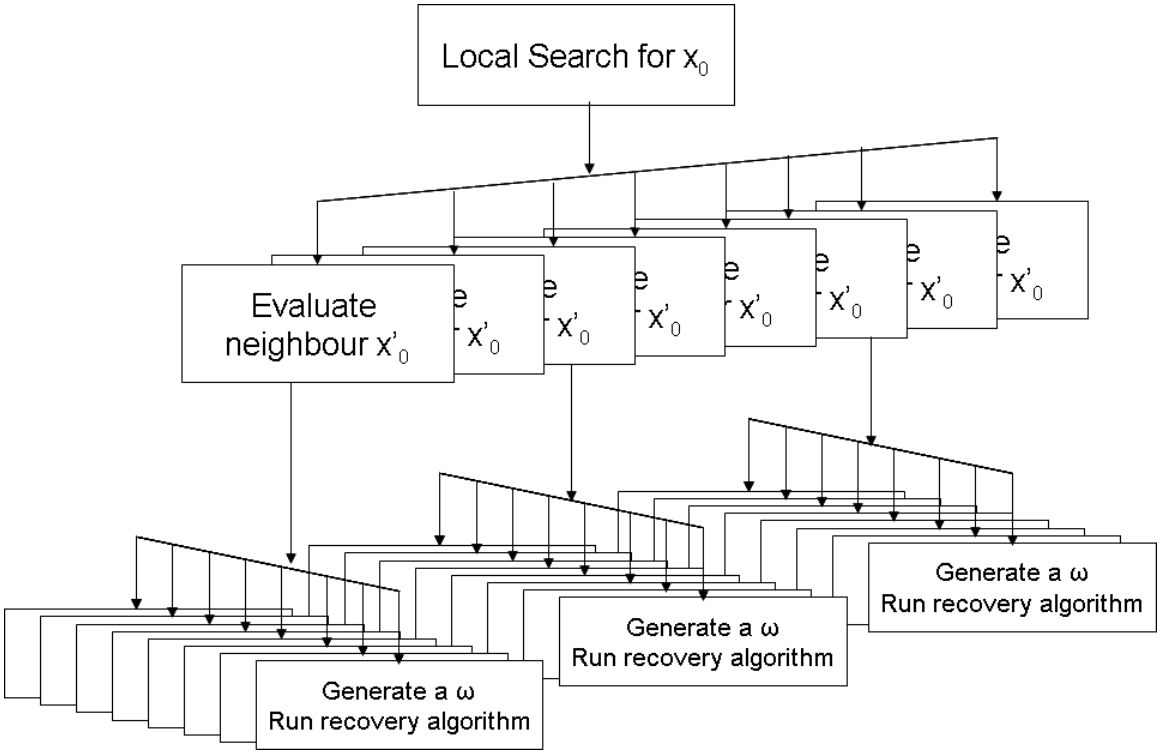


Figure 5.2: Solution approach architecture for solving Nimo

units. Hence, it is very natural to move it to a distributed architecture, where processes are spawn as new tasks in different cores. Cores are computing units either on the same CPU chip, different chip in the same computing box, or across many computing boxes spread over different geographical locations connected via network. We propose the architecture to solve the problem as shown in Figure 5.2. Each box is an independent process that can be executed on a different core. We have a main controller process that executes the local search. Evaluation of each neighbour is spawn off as a separate process, and in turn, spawns off multiple processes, each simulating one instance of the random scenario ω. This architecture may sit nicely on a cloud computing framework, or a cluster of computers in a network with some scheduling programmes that control

the computers' workloads. While distributed computing on cloud may still sound very academic, in the recent years, cloud technology has moved on closer to consumer arena (for example, Google Chrome and Apple iCloud). Although these examples are still mainly on distributed storage more than computing, the time when cloud computing becomes a norm should be near.

5.2 Nimble yard plan template problem

We now apply the nimble optimisation formulation into the yard plan template problem. In Section 5.2.1, we first give an introduction and motivation to why we focus on vessel arrival schedule uncertainty in a container terminal that adopts consolidated and dedicated yard plan strategy. We then give a description of the nimble yard plan template problem in Section 5.2.2 followed by the mathematical definition of the problem. As there can be many variations of the nimble yard plan template problem, depending on the recovery policy used, we explore three cases. Section 5.2.3 explores the first case where a strict assignment policy is used in the recovery. We present two heuristic approaches to solve the problem. The first approach uses a surrogate model and the second uses the simulation guided local search algorithm as described earlier. However, the experiments are conducted on a single core computer. Experimental results are presented and discussed. Section 5.2.4 explores the second case where a "one change policy" is used. A modified simulation guided local search algorithm is presented to solve this problem, and experiments are conducted and discussed. In Section 5.2.5, we discuss the third case of solving a generic nimble yard plan template

problem, where a generic recovery policy is used. We discuss the complexity of solving this problem.

5.2.1 Motivation

In the problem studied in Chapter 4, it is assumed that the inputs to the yard plan template problem are deterministic. However, in real life application, uncertainty prevails. Vessels plying a specific service may arrive at slightly different times from week to week, as they may experience different weather conditions during their voyage, as well as delays that may have occurred in their earlier port of visits. The number of containers to be loaded may also vary from call to call, as the shipping demand may change due to shipper-consignee ordering uncertainty. This will not only change the number of containers to be loaded, but also the number of containers received into the yard. Furthermore, the arrival schedule of the containers into the yard may also vary, depending on the trucking companies and local shippers (in the case of local export containers) as well as the arrival schedule of vessels (in the case of transshipment containers).

While all these areas of uncertainty pose complexity to yard planning, we have learnt from a global container terminal operator that the uncertainty in vessel schedule has the highest impact to yard operations for terminals that adopt the consolidated and dedicated yard plan strategy. Most of the terminals that we come across are adopting this strategy, i.e. the containers to be loaded into a vessel are stacked in groups and no two services can be assigned to the same group. The reasons for them to adopt this strategy are as follows. When a vessel is alongside, although the number of retrieval activities at these locations are high, it is well controlled as terminal planners will have considered

the composition of the containers grouped together, and hence the sequence of which the containers are retrieved will not create an activity concentration nor RMG contention. They will station a yard crane at each yard location, and the trucks will shuttle between the quay crane and the yard crane in a conveyor belt fashion. However, the assumption made is that there will not be another vessel's containers that are concurrently retrieved in the vicinity, as the yard locations are well chosen based on the expected arrival time of the vessels. When the vessels arrive at times different from the expected, there are possibilities that two vessels' containers are close to each other, and they happen to retrieve the containers simultaneously. Terminal operators can then choose to de-conflict by moving some of the containers to a further location before the vessels arrive, but that will mean incurring extra cost to the terminal operations. However, if they are not moved, it results in massive activity concentration and likely contention for yard cranes if extra yard cranes are not able to move in to relieve the contention. These may last for many hours in a stretch or even the duration of the whole port stay, as consolidated strategy usually stacks a substantial number of containers for the same vessel in the same location. These result in the vessels' departure time to be delayed due to the bottleneck in retrieving the containers from the yard. Hence, instead of considering all the areas of uncertainty, which is a huge optimisation model, in this chapter, we focus only on the uncertainty of the vessel arrival schedule, that impacts the yard operations the most.

To the best of our knowledge, we do not come across any paper that directly addresses the yard planning problem with respect to the uncertainty of vessel arrival schedule. In the earlier chapter, the deterministic yard plan template problem is studied

to find where the containers will be stored at different parts of the yard blocks, such that violations of the activity thresholds can be minimised. We assume that the vessel arrival schedule is fixed and given. In the scenario where vessel arrival schedule is uncertain, we can approach the problem by finding a *robust solution* - a solution that remain feasible when when subjected to uncertainty. However, in this chapter, we would like to find a solution that enables us to change easily when the uncertainty is revealed. The decision is made in two stages, where the stage 1 decision is made before the uncertainty is revealed, and stage 2 is a recovery of the stage 1 solution, after the uncertainty is revealed. This makes the problem challenging, due to having to find a stage 1 solution such that the cost of recovery is minimised, after the uncertainty is revealed. Since we do not know how the uncertainty will be revealed, we minimise the cost of the worst case scenario. We call the stage 1 problem as the **nimble yard plan template problem** because the solution is deemed as "nimble" (agile to changes in stage 2 when the uncertainty is revealed).

5.2.2 Problem definition

We consider a container terminal as described in Chapter 3 Section 3.2. We further assume that the container terminal adopts a consolidated and dedicated yard plan strategy, and the size of the groups are at least two slots each. We consider the same mathematical formulation for model IMPR as presented Chapter 4 Section 4.2.3. The solution depicts, the assignment of services to the yard ranges, as well as the number of the containers received into each of the yard ranges, such that the amount of activity concentration and RMG contention are minimised.

In the nimble yard plan template problem, we consider uncertainty in service arrival schedule. Terminal operators would like their yard plan template to be easily changeable when the service arrival schedule changes. However, as the terminal operates continually, the yard is always filled with some containers belonging to some services at a different stage of build-up. It will be costly to change the assignment of services to yard ranges all over again when a service changes its arrival schedule, as it may incur extra physical resources to move the containers around. It is cheaper to change the number of containers received into the ranges as the containers may not have arrived yet. For simplicity, we have assumed that the cost of changing the assignment of services to ranges is infinitely large if the number of changes exceeds a pre-set threshold, (i.e., we do not allow too many changes of the assignment of services to ranges), and the cost of changing the number of containers received into the ranges is zero. Given the possible changes in the service arrival schedule, our problem is to find a good starting solution before the uncertainty is revealed, such that we minimise the objective value (violations) of the worst case scenario - a combination of possible vessel arrival schedules that gives the worst objective value at optimum. Hence, regardless of how the uncertainty is revealed, we will be able to change the number of receiving containers in the ranges and the assignments of services to ranges, such the amount of violations is bounded.

In the following subsections, we introduce the mathematical notations and formulation for the nimble yard plan template problem. We first present the mathematical model assuming deterministic inputs. Following that, we define the problem as a two stage decision process, and the nimble yard plan template problem is to be solved in stage 1

before the uncertainty is revealed, taking all possible outcomes of the uncertainty into consideration.

Static Model

As we refer to the model IMPR in Chapter 4 Section 4.2.3 in our formulation, we reproduce the model in a simplified form (without the $\mathcal{C}$ and SH as they are not used here) as follows. We refer it as the **static model**.

Input parameters to the model - constants

$\mathcal{T}$ = Set of time periods

$\mathcal{D}_{\mathcal{T}}$ = $(\mathcal{V}, DA_{it}^{20}, DA_{it}^{40}, \mathcal{L}_i)$

$\mathcal{S}_{\mathcal{T}}$ = $(\mathcal{R}, \mathcal{G}, \mathcal{B}, CS_r, SR_r, MA_g, RY, MY_b, W^k, W^p)$

M = Big Number

Decision variables

x_{irt}^{20} = Number of 20-footers received by service i into range r in time t

x_{irt}^{40} = Number of 40-footers received by service i into range r in time t

y_{irt}^{20} = Number of 20-footer loaded by service i from range r in time t

y_{irt}^{40} = Number of 40-footer loaded by service i from range r in time t

d^{20}_{irt} = Number of 20-footer spaces reserved for 20-footers in range r for service i at time t

d^{40}_{irt} = Number of 40-footer spaces reserved for 40-footers in range r for service i at time t

u_{ir} = 1 if service i is assigned to range r, 0 otherwise

k_{rt} = Number of excess activities in range r at time t, such that the k_{rt} contributed to some groups $g \in \mathcal{G}$ to violate MA_g

p_{bt} = Number of activities which need to wait for a yard crane, measured in number of activities above the capacity of the yard cranes in the block b at time t

j_{mr} = Indicator variable used in constraints 5.16 and 5.17

$x^{20}_{irt}, x^{40}_{irt}, p_{bt}, k_{rt} \geq 0$ are real variables. u_{ir}, j_{mr} are binary variables. $y^{20}_{irt}, y^{40}_{irt}, d^{20}_{irt}, d^{40}_{irt}$ are derived variables.

Objective function

$$\text{Minimise } obj = W^k \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} k_{rt} + W^p \sum_{b \in \mathcal{B}} \sum_{t \in \mathcal{T}} p_{bt}$$

Constraints

$$\sum_{r \in \mathcal{R}} x^{20}_{irt} = DA^{20}_{it} \quad \forall i \in \mathcal{V}, t \in \mathcal{T} \tag{5.3}$$

$$\sum_{r \in \mathcal{R}} x^{40}_{irt} = DA^{40}_{it} \quad \forall i \in \mathcal{V}, t \in \mathcal{T} \tag{5.4}$$

$$y^{20}_{irt} = \left(\frac{\sum_{t' \in \mathcal{T}} x^{20}_{irt'}}{(1+|\mathcal{L}_i|)/2}\right) \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{L}_i \tag{5.5}$$

$$y^{40}_{irt} = \left(\frac{\sum_{t' \in \mathcal{T}} x^{40}_{irt'}}{(1+|\mathcal{L}_i|)/2}\right) \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{L}_i \tag{5.6}$$

$$\sum_{t \in \mathcal{T}} y^{20}_{irt} + \sum_{t \in \mathcal{T}} y^{40}_{irt} \geq u_{ir} \quad \forall i \in \mathcal{V}, r \in \mathcal{R} \tag{5.7}$$

$$\sum_{t \in \mathcal{T}} y^{20}_{irt} + \sum_{t \in \mathcal{T}} y^{40}_{irt} \leq u_{ir} M \quad \forall i \in \mathcal{V}, r \in \mathcal{R} \tag{5.8}$$

$$\sum_{i \in \mathcal{V}} u_{ir} \leq 1 \quad \forall r \in \mathcal{R} \tag{5.9}$$

$$\sum_{r \in g} \left(\sum_{i \in \mathcal{V}} \left(x^{20}_{irt} + x^{40}_{irt} + y^{20}_{irt} + y^{40}_{irt}\right) - k_{rt} \right) \leq M A_g \quad \forall g \in \mathcal{G}, t \in \mathcal{T} \tag{5.10}$$

$$\sum_{i \in \mathcal{V}} \sum_{r \in b} \left(x^{20}_{irt} + x^{40}_{irt} + y^{20}_{irt} + y^{40}_{irt}\right) - p_{bt} \leq RY\,(MY_b) \quad \forall b \in \mathcal{B}, t \in \mathcal{T} \tag{5.11}$$

$$d^{20}_{irt} = \sum_{t' \in \mathcal{A}_i \text{ and } t' \leq t} x^{20}_{irt'} \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{A}_i \tag{5.12}$$

$$d^{20}_{irt} = \sum_{t' \leq t} x^{20}_{irt'} + \sum_{t' \in \mathcal{A}_i} x^{20}_{irt'} \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{T} \text{ and } t \notin \mathcal{A}_i \tag{5.13}$$

$$d^{40}_{irt} = \sum_{t' \in \mathcal{A}_i \text{ and } t' \leq t} x^{40}_{irt'} \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{A}_i \tag{5.14}$$

$$d^{40}_{irt} = \sum_{t' \leq t} x^{40}_{irt'} + \sum_{t' \in \mathcal{A}_i} x^{40}_{irt'} \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{T} \text{ and } t \notin \mathcal{A}_i \tag{5.15}$$

$$\frac{\sum_{i \in V} d^{20}_{irt}}{CS_r} \leq 2m(j_{mr}) + SR_r - 2m \qquad \forall r \in \mathcal{R}, t \in \mathcal{T}, m \in \mathbb{Z}\ s.t.\ 1 \leq m \leq \left\lfloor \frac{SR_r}{2} \right\rfloor \tag{5.16}$$

$$\frac{\sum_{i \in V} d^{40}_{irt}}{CS_r} \leq \left(\left\lfloor \frac{SR_r}{2} \right\rfloor - m + 1\right)(1 - j_{mr}) + m - 1 \qquad \forall r \in \mathcal{R}, t \in \mathcal{T}, m \in \mathbb{Z}\ s.t.\ 1 \leq m \leq \left\lfloor \frac{SR_r}{2} \right\rfloor \tag{5.17}$$

For convenience of later discussions that follow, we introduce an abstraction of the formulation as follows. In the static model, the decision variable u_{ir} is set to 1 when service i is assigned to range r, and zero otherwise (see constraints 5.7 and 5.8). We separate the decision variable into two vectors, the vector $u = (u_{ir})^T$, and vector v for remaining decision variables, i.e. $(x^{20}_{irt}, x^{40}_{irt}, y^{20}_{irt}, y^{40}_{irt}, p_{bt}, k_{rt}, j_{mr})^T$. Then the static model can be written as

$$\begin{bmatrix} \underset{(u\ v)^T}{\text{Minimise}} \quad \mathbb{C}^T v \\ s.t. \quad \mathbb{A} \begin{pmatrix} u \\ v \end{pmatrix} \geq \mathbb{B} \end{bmatrix} \tag{Model V.II}$$

where $\mathbb{A}$, $\mathbb{B}$ and $\mathbb{C}$ are the respective matrices of the static model.

Nimble model

Now consider that the service i arrival schedule is uncertain, and is offset by ϵ_i. We assume ϵ_i, the difference between the new scheduled time to the original scheduled time, is measured in units of time periods, and we assume it to be integer and uniformly distributed from $-\delta$ to δ. We let ω represent the uncertainty vector of $(\epsilon_i | \forall i \in \mathcal{V})$. Note

that there may be more than one service that may change their schedule, i.e. more than one ϵ_i can be non-zero in ω. We also let Ω represent the set of all possible uncertainty vectors ω.

Consider the problem being decomposed into a 2-stage decision process, called Stage 1 and Stage 2. In Stage 1, we determine u^* (the assignment of services to ranges) before the realisation of ω, and in Stage 2, we determine the optimum v (which includes the number of containers received into the ranges), and the new u which should have limited differences from u^*, given the realisation of ω. Stage 2 can be viewed as a recovery stage where it has to find a good solution (minimise objective value) based on a given u^* (the decision already made in Stage 1). We let $\mathbb{F}(u, v|\ u^*, \omega)$, a function of u and v given u^* and ω, denote the objective value of Stage 2. The optimum solution u^* to the Stage 1 problem is when the maximum of the objective values of all Stage 2 decisions among all possible realisations of ω is minimised. We call the Stage 1 problem the nimble yard plan template problem. Hence, in mathematical notation, the nimble yard plan template problem can be expressed as

$$\underset{u^*}{\text{Minimise}} \ \max_{\omega} \ \min_{u,v} \ \mathbb{F}(u, v|\ u^*, \omega) \qquad \text{(Model V.III)}$$

Many variations of the nimble yard plan template problem can be defined depending on how Stage 2 recovery is defined, i.e. the cost of change of u in Stage 2 is measured and limited. Specifically, it depends on how $\mathbb{F}(u, v|\ u^*, \omega)$ is defined. We consider only three cases in our discussion in this chapter. In the first case, we assume a **strict**

assignment policy, where Stage 2 is not allowed to change u, i.e., the cost of changing u in Stage 2 is infinitely large, and hence $u = u^*$. In the second case, we assume a **one change policy** such that one of the assignments of services to ranges can be changed, and this is elaborated as follows. Let $A = (a_1, a_2, \ldots, a_{|\mathcal{R}|})$ represent the assignment vector in Stage 1 where $a_r = i$ if and only if service i is assigned to range r, i.e., $u_{ir} = 1$, and $a_r = 0$ if and only if range r is not assigned to any service, i.e. $u_{ir} = 0, \forall i \in \mathcal{V}$. Hence A and u are inter-convertible. In the one change policy, we assume that at most one of the assignments a_r in the assignment vector A can be modified by Stage 2 at zero cost, and infinitely large cost if more than one assignment in A are modified. In the third case, we define a generic nimble yard plan template problem, where in Stage 2, there is no limit to the changes that can be made in Stage 2. However, the cost of changes is measured and minimised.

In the following sections, we will explore solution approaches for the first two cases. For the third case, we present the problem definition, followed with some discussion on the complexity. We do not explore further with any experiments but we think it could be solved on a distributed computing environment.

5.2.3 Solution approaches for case 1: strict assignment policy

Stage 2 - a recovery algorithm RA

We can solve for Stage 2 optimally by adapting the static model to solve for v given u^* and ω as follows. For simplicity of the model, we have assumed that the arrivals of the containers to be loaded to service i are also offset by ϵ_i. Hence, the constraints 5.3,

5.4, 5.5 and 5.6, in the static model are modified respectively as follows:

$$\sum_{r\in\mathcal{R}} x^{20}_{irt+\epsilon_i} = DA^{20}_{it} \quad \forall i \in \mathcal{V}, t \in \mathcal{T} \tag{5.18}$$

$$\sum_{r\in\mathcal{R}} x^{40}_{irt+\epsilon_i} = DA^{40}_{it} \quad \forall i \in \mathcal{V}, t \in \mathcal{T} \tag{5.19}$$

$$y^{20}_{irt+\epsilon_i} = \left(\frac{\sum_{t'\in\mathcal{T}} x^{20}_{irt'}}{(1+P_i)/2}\right) \quad \forall i \in \mathcal{V}, r\in \mathcal{R}, t \in \mathcal{L}_i \tag{5.20}$$

$$y^{40}_{irt+\epsilon_i} = \left(\frac{\sum_{t'\in\mathcal{T}} x^{40}_{irt'}}{(1+P_i)/2}\right) \quad \forall i \in \mathcal{V}, r\in \mathcal{R}, t \in \mathcal{L}_i \tag{5.21}$$

As ω adjusts only the subscripts of the static model, we let $\mathbb{A}(\omega)$ represent the adjusted matrix $\mathbb{A}$ with the uncertainty ω. Hence, given u^* and a realisation of ω, we minimise $\mathbb{F}(u, v|\ u^*, \omega)$ where $\mathbb{F}(u, v|\ u^*, \omega)$ is defined as follows, and we call this the recovery algorithm (RA).

$$\mathbb{F}(u, v|\ u^*, \omega) = \begin{bmatrix} \mathbb{C}^T v \\ \text{subject to} \\ \mathbb{A}(\omega) \begin{pmatrix} u \\ v \end{pmatrix} \geq \mathbb{B} \\ u = u^* \end{bmatrix} \tag{5.22}$$

Stage 1 - nimble solution assuming RA

Finding the optimum solution for Stage 1 could be intractable and can be shown intuitively as follows. Since Ω is a finite set, we can enumerate $\{v_\omega|\forall\omega \in \Omega\}$, where v_ω is the optimum solution for Stage 2 given u^* and ω. Hence, to solve for optimum u^* in

Stage 1, the set $\{v_\omega | \forall \omega \in \Omega\}$ has to be determined. Since each service has a possibility of arriving δ time periods earlier or later, i.e. $-\delta \leq \epsilon_i \leq \delta$, and the terminal has $|\mathcal{V}|$ services, that will mean that Ω has the size of $(2\delta + 1)^{|\mathcal{V}|}$, and in the most practical sense, it will be a very large number. Hence there will be $O(\delta^{|\mathcal{V}|})$ number of variables and we do not expect that we can get a model of this size to run successfully with the current state of computing power.

Loading Optimised Surrogate Model (LOSM)

As the nimble yard plan template problem could be intractable, we explore a surrogate model, called Loading Optimised Surrogate Model (LOSM) for solving Stage 1 with the following motivation. Since we are adopting a consolidated strategy, it means that the retrieval activities are concentrated during loading of the service at any yard range. Hence it will be bad if two adjacent yard ranges are loading at the same time period. Since the service arrival schedule is non-deterministic, we therefore want to increase the time buffer between two adjacent retrieval activities such that any changes to service arrival schedule that may cause overlapping of loading time periods can be minimised.

The details are as follows. We first pre-compute a matrix $Q_{ii'}$ that denotes the time gap between service i and i', defined as the number of time periods from the departure of service i to the arrival of service i', or the number of time periods from the departure of service i' to the arrival of the service i, whichever is smaller. To illustrate with an example, if $\mathcal{L}_i = \{3, 4, 5\}$, and $\mathcal{L}_{i'} = \{18, 19, 20\}$, and assuming there are 21 time periods in a week, then $Q_{ii'} = 3$ as we have time periods $21, 1$ and 2 between time periods 20 and 3, with time wrapping around from time period 21 back to 1. In another

example, if $\mathcal{L}_i = \{5, 6, 7\}$ and $\mathcal{L}_{i'} = \{6, 7, 8\}$, i.e. they overlap, then $Q_{ii'}$ will be the negative of the number of time periods that they overlap, i.e. $Q_{ii'} = -2$. The LOSM is an adaptation of the static model (Model V.II) by imposing a new constraint set and an extra term in the objective function. For any pair of consecutive yard ranges, r and r^+ (r^+ denotes the range next to the range r in the same yard block), whose assigned services are i and i' (i.e. $u_{ir} = 1$ and $u_{i'r^+} = 1$), we impose the rule that $Q_{ii'}$ has to be bigger or equal than a fixed threshold TG, i.e. $Q_{ii'} + q_r \geq TG$. The slack variable q_r is introduced to capture the violation when $Q_{ii'}$ is smaller than TG. Then, we add the term $W^q \sum_{r \in \mathcal{R}} q_r$ into the objective function, where W^q is the weight for this violation. Hence the LOSM is expressed as follows.

$$\left[\begin{array}{c} \underset{(u\ v)^T}{\text{Minimise}} \ \ \mathbb{C}^T v + W^q \sum_{r \in \mathcal{R}} q_r \\ s.t. \quad \mathbb{A} \begin{pmatrix} u \\ v \end{pmatrix} \geq \mathbb{B} \\ Q_{ii'} + q_r + (1 - u_{ir})M + (1 - u_{i'r^+})M \geq TG \\ \forall r \in \mathcal{R},\ \forall i, i' \in \mathcal{V} \text{ and } \exists\ r^+ \end{array}\right] \qquad \text{(Model V.IV)}$$

Simulation Guided Local Search Algorithm (SGLS)

The LOSM depends on a parameter TG, and it should be set such that $TG \geq 2\delta$. This is because when a pair of consecutive ranges have their assigned services' loading time periods separated by 2δ, then regardless of how the service arrival time period is adjusted within δ, they will not have any overlapping of loading periods. However, from the experiments, we find that LOSM is not able to terminate within the time-out set at

eight hours when TG is set beyond six. Hence, we propose to use the simulation guided local search (SGLS) where we use LOSM with TG set to six to obtain a good starting solution, and then apply the hill-climbing local search algorithm to further improve the solution to cater for scenario when $\delta > 3$.

The details are as follows (see flowchart in Figure 5.3). A starting solution u is found by solving LOSM. The solution is evaluated (using the subroutine *Evaluate(u) in the flowchart) by simulating Stage 2 with 50 random scenarios ω, and each is solved with the RA given u and ω to obtain the objective value. The maximum of the objective values of the 50 samples is returned. The algorithm then goes into a loop, where each loop performs one step of the local search. The neighbourhood set $N(u)$ for the local search is generated as follows. Note that u encodes an assignment of services to yard ranges, and to simplify discussion, we let $(a_1, a_2, \ldots, a_{|\mathcal{R}|})$ represent the assignment, such that a_r denotes the service that is assigned to range r, i.e. $a_r = i$ if and only if $u_{ir} = 1$. A neighbour to the solution u is defined to have a pair of assignment swapped, i.e. the values of a_r and $a_{r'}$ are exchanged, for some r and r' in $\mathcal{R}$. The neighbourhood set $N(u)$ is the set of all possible neighbours of u. Each neighbour is evaluated using the subroutine *Evaluate(u'), and the neighbour that gives the lowest objective value is chosen as "Best". If the objective value of "Best" is not better than that of u, the local search terminates, otherwise, "Best" will be chosen as the next solution in the local search, and the loop repeats.

The detailed algorithm in pseudo code is given in Figure 5.4 for reference. A couple of short cuts in the algorithm are inserted to speed up the computation time. First and most obvious, we terminate the local search whenever a solution with zero objec-

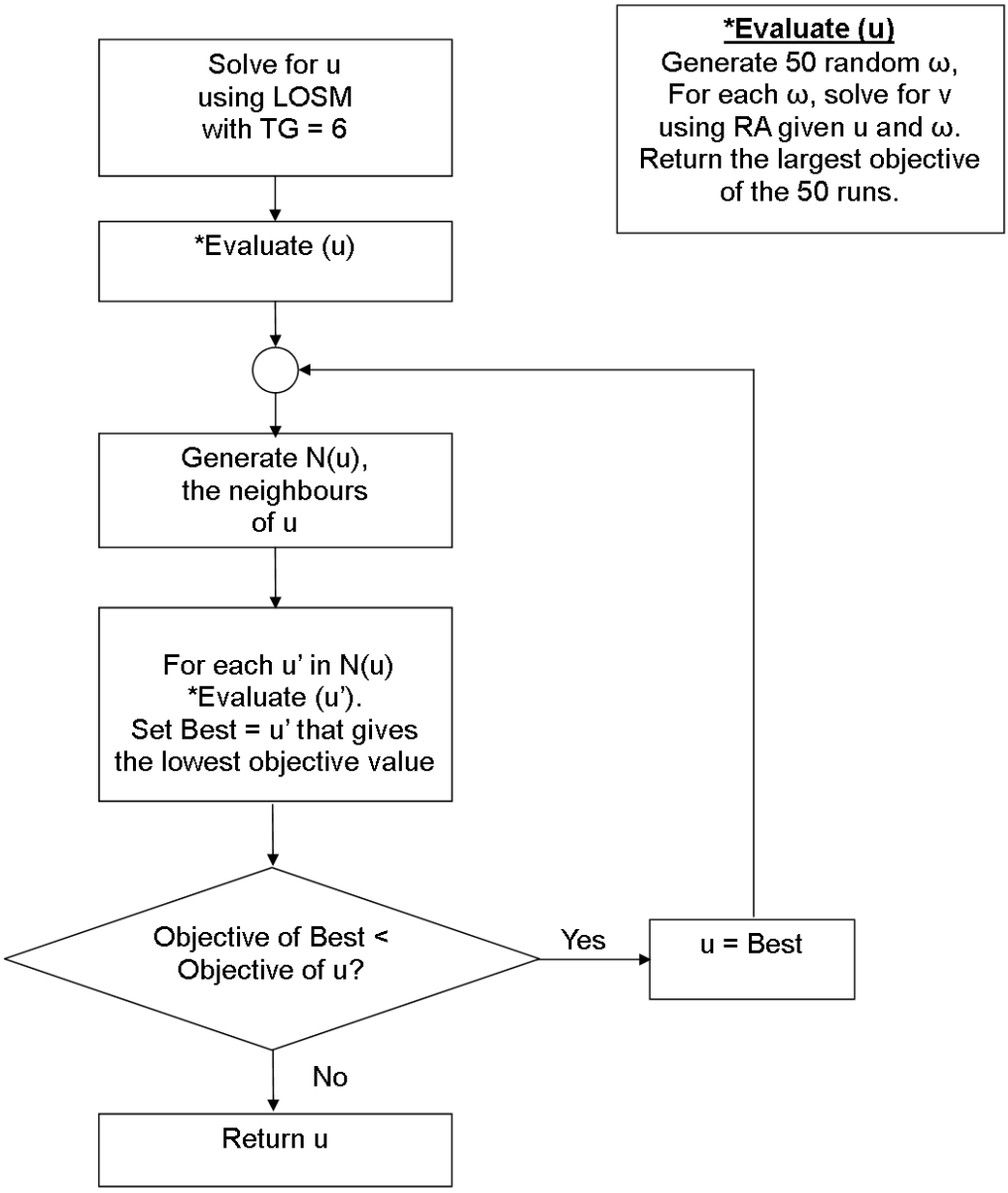


Figure 5.3: Flowchart for SGLS Algorithm

tive is found (see line 32). Next, we terminate the sampling process of generating 50 samples when we encounter any of them that gives an objective value worse than the best solution found so far (since this neighbour will never be chosen anyway, see line

SGLS Local Search Algorithm
Initialise Obj_{max} as -1
Generate a starting solution vector u with LOH (Model V.IV)
repeat
    Generate a random ω and solve RA (5.24) to determine the optimum solution
    if solution.objective $> Obj_{max}$ **then**
        Remember ω as ω_{worst}
        Remember solution.objective as Obj_{max}
    end if
    Remember Obj_{max} as $Best$
until 50 times
while Not converge **do**
    Generate the neighbours of $u = N(u)$
    Set $Neighbour_{best}$ as null
    for all $u' \in N(u)$ **do**
        Solve RA (5.24) with ω_{worst} to determine the optimum solution
        if solution.objective $\geq Best$ **then**
            Skip this u' and continue with next item in the **for** loop
        end if
        Remember the solution.objective value as Obj_{max}
        repeat
            Generate a random ω and solve RA (5.24) to determine the optimum solution
            if solution.objective $> Obj_{max}$ **then**
                Remember ω as ω_{worst}
                remember the solution.objective as Obj_{max}
            end if
            if solution.objective $> Best$ **then**
                Skip this u' and continue with next item in for loop
            end if
        until 50 times
        if $Obj_{max} = 0$ **then**
            Terminate local search
        end if
        if $Obj_{max} < Best$ **then**
            $Best = Obj_{max}$
            Remember u' as $Neighbour_{best}$
        end if
    end for
    if $Neighbour_{best}$ not null **then**
        Set $u = Neighbour_{best}$
    else
        Convergence found, terminate local search
    end if
end while

Figure 5.4: Pseudo code for SGLS Algorithm.

28). Lastly, since we will terminate the sampling process when we encounter an ω that gives an objective value worse than the best, as we want to find an ω that gives a lousy solution as early as possible (see line 18). Hence, we store the ω that is responsible for the worst case so far, and this ω is always used first before generating the next 50 random samples. From the experiments, we find that the algorithm indeed does not need to generate the remaining 50 samples in most of the solutions evaluated in the local search.

Experimental results

We conduct the experiments as follows. For Stage 1, we experiment using LOSM and SGLS. For the purpose of comparison, we also experiment using a naive model, where Stage 1 is solved using the static model, ignoring the uncertainty. Stage 2 is solved using the recovery algorithm RA, but for comparison, we have also experimented using a "stubborn" policy, where we do not perform any recovery, i.e. we retain the assignment of services to yard ranges and the number of containers received into each range. The stubborn policy can be implemented simply by adapting the static model as follows.

$$\left[\begin{array}{c} \underset{(u'\ v')^T}{\text{Minimise}}\ \ \mathbb{C}^T v' \\ s.t. \quad \mathbb{A}(\omega)\begin{pmatrix} u' \\ v' \end{pmatrix} \geq \mathbb{B} \\ u'_{ir} = u_{ir} \quad \forall i \in \mathcal{V}, r \in \mathcal{R} \\ x'^{20}_{irt} = x^{20}_{irt-\epsilon_i} \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{T} \\ x'^{40}_{irt} = x^{40}_{irt-\epsilon_i} \quad \forall i \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{T} \end{array}\right] \qquad \text{(Model V.V)}$$

For simplicity of notation, we label the various approaches in the experiments as given in Table 5.1.

Table 5.1: Abbreviations used to label various approaches in experiments

Approaches	Base	RA	LOSM	LOSM-R	SGLS-R
Stage 1	static	static	LOSM	LOSM	SGLS
Stage 2	stubborn	RA	stubborn	RA	RA

Experimental set up

We conduct the experiments with the terminal that is as described in Chapter 3 Section 3.2. We further partition the blocks into ranges of five slots for the set of ranges $\mathcal{R}$ and adjacent ranges are grouped in pairs in the set of groups $\mathcal{G}$. Each group has an allowable maximum activity count of 90, i.e. $MA_g = 90$. We also assume an RMG is able to handle 120 moves per time period and each yard block has three RMGs.

We model the time window with 21 time periods. 21 services will arrive throughout the week, with one service arriving in each time period, and staying alongside for loading for one time period. It is intentional to have loading of each service to be within one time period, so that the retrieval activities in each range will be more intense and hence activity concentration violations can be manifested easily with changes of service arrival schedule. All services are assumed to be homogeneous, and will load the same number of 20-footers and 40-footers. The number of TEUs received by each service prior to the loading time periods follow the same arrival schedule as follows. Without loss of

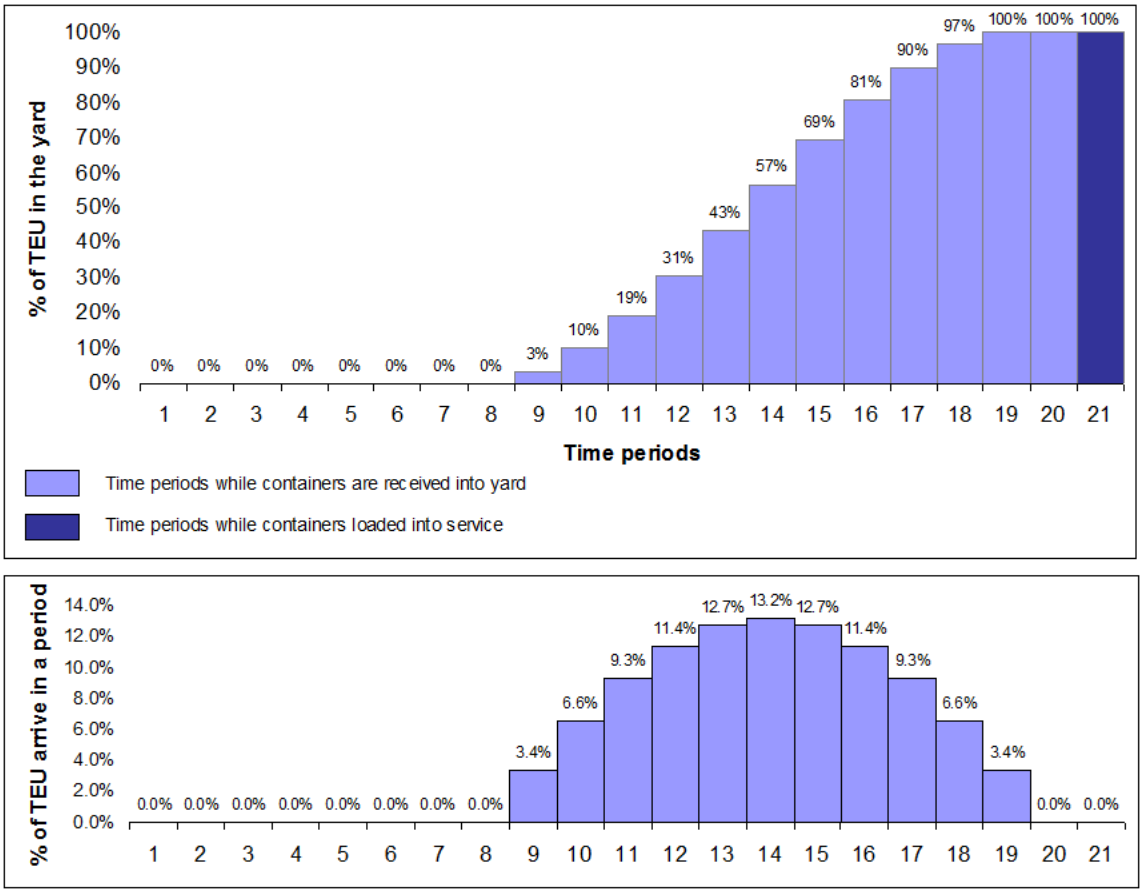


Figure 5.5: Percentage TEU of the total loaded containers received into the yard prior to the loading period for each service.

generality, let's assume that a service is to load in period 21, as shown in Figure 5.5. The schedule depicts that 3.4% of the TEUs will arrive into the yard in time period 9, 6.6% in time period 10, and so on, and continue to arrive in subsequent periods to period 19. Figure 5.5 also shows graphically the cumulative percentage of the TEUs which should have arrived by each time period, where 100 percent will be the final TEUs the service will load. The weights for activity concentration and RMG contention, W^k and W^p, are set to 1. This is because both measurements (i.e. k_{rt} and p_{bt}) are in the same unit of measure, being the number of activities exceeding the thresholds, and

hence a unit of violation in k_{rt} is equivalent to one unit in p_{bt}. For LOSM, we further set $TG = 6$ and $W^q = 10$. We find that the solution is not sensitive to the value of W^q as all the runs of LOSM terminate with optimum solution at zero objective value. As the LOSM is not able to terminate with $TG > 6$, and we want TG to be as big as possible, we set TG to the maximum at six.

For each solution approach as tabulated in Table 5.1, an experiment set comprises of running experiments by varying the δ value from 1 to 10. When δ is 10, the uncertainty is practically random, i.e. a service is able to come at any time period of the week. For each value of δ, we apply the strategies for Stage 1, and then we randomly generate 400 samples of $\omega = (\epsilon_1, \epsilon_2, \ldots, \epsilon_i, \ldots, \epsilon_{|\mathcal{V}|})$, where $-\delta \leq \epsilon_i \leq \delta$, and for each ω we apply the various strategies for Stage 2. We collect the objective value of the Stage 2 solution (i.e. amount of violations) for each ω , and observe the average and maximum (worst case) violations of the 400 samples. One experiment set is conducted for each of the approaches in Table 5.1 except for SGLS-R, as SGLS-R solves for Stage 1 non-deterministically using simulation in the local search. The experiment set for SGLS-R is replicated 10 times, and the means and standard deviations of the 10 replications are computed.

Results

Table 5.2 tabulates the results of the experiments, showing the average and worst case objective values. Note that the objective value measures the violations, hence, the smaller the value, the better the solution. As the experiment set for SGLS-R is replicated 10 times, the means and standard deviations of the average and worst case objective

Table 5.2: Experimental results: average and worst case violation

	Average Violation						Worst Case Violation					
δ	Base	RA	LOSM	LOSM-R	SGLS-R		Base	RA	LOSM	LOSM-R	SGLS-R	
					Mean	Std Dev					Mean	Std Dev
0	0.00	0.00	0.00	0.00	0.00	0.00	0	0	0	0	0.00	0.00
1	156.56	16.45	82.94	0.00	0.00	0.00	490	70	182	0	0.00	0.00
2	158.09	16.62	99.68	0.00	0.00	0.00	465	88	216	0	0.00	0.00
3	158.96	12.13	100.35	0.00	0.00	0.00	504	114	217	0	0.00	0.00
4	163.14	10.42	109.68	0.00	0.00	0.00	438	88	227	0	0.00	0.00
5	160.42	9.53	110.22	0.00	0.00	0.00	479	88	236	0	0.00	0.00
6	164.56	8.18	102.61	0.00	0.00	0.00	472	114	242	0	0.00	0.00
7	173.64	6.71	130.52	0.24	0.00	0.00	454	88	405	6	0.00	0.00
8	180.32	6.49	139.87	2.03	0.42	0.50	441	120	373	44	13.60	16.02
9	175.91	5.51	152.39	1.45	0.45	0.56	523	120	370	50	15.60	16.24
10	172.28	5.42	171.02	2.32	0.76	0.78	535	90	521	48	29.00	24.17

Table 5.3: Detailed results of 10 replications of SGLS-R experiments - average violation

	Experiments sets											
δ	run 1	run 2	run 3	run 4	run 5	run 6	run 7	run 8	run 9	run 10	Mean	Std Dev
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0.15	0.31	0.16	0.25	0.21	1.48	1.22	0.15	0.16	0.12	0.423	0.497
9	0.19	0.22	0.19	1.10	0.25	0.27	0.00	0.24	0.24	1.82	0.454	0.563
10	0.37	2.40	0.19	0.18	1.89	0.73	0.31	0.99	0.16	0.37	0.761	0.783

value of the 10 replications are shown instead. The details of the average case and worst case objective values for each of the 10 runs are tabulated in Table 5.3 and 5.4, respectively.

The purpose of the experiments is to observe the quality of the yard plan template (in terms of number of violations) under various approaches. First, we compare the

Table 5.4: Detailed results of 10 replications of SGLS-R experiments - worst case violation

	Experiments sets											
δ	run 1	run 2	run 3	run 4	run 5	run 6	run 7	run 8	run 9	run 10	Mean	Std Dev
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	6	6	6	6	6	44	44	6	6	6	13.60	16.02
9	6	6	6	45	18	12	0	6	12	45	15.60	16.24
10	12	68	12	6	66	50	28	30	6	12	29.00	24.17

approaches that have recovery algorithm RA in Stage 2 versus those with the stubborn policy (i.e., RA versus Base, LOSM-R and SGLS-R versus LOSM). The violations with recovery are very much lower compared to those without recovery, and this can be seen in both the average and worst case objective values. This shows that regardless of whether the Stage 1 solution is nimble or not, having a recovery in Stage 2 to change the solution after the uncertainty is revealed is very important to reduce the number of violations after the uncertainty is revealed. Next, we examine the added value of finding a nimble solution in Stage 1 compared to the static model. For the case without nimble solution in Stage 1 (RA versus Base), the average violations with RA ranges from 3% to 10% to that of the solution without recovery (Base). But for the case with nimble solution in Stage 1 (SGLS-R versus LOSM), the violations can be totally eliminated with recovery in some of the cases, and for the rest of the cases, the average violation ranges from 0.3% to 0.4% to that of the solution without recovery (LOSM). This shows that having

nimble solution provides a very good starting solution for the recovery algorithm to repair from, when the uncertainty is revealed.

Having a nimble solution in Stage 1 followed by a recovery algorithm in Stage 2 is a good approach to handle uncertainty, and the two solution approaches presented are the LOSM-R and SGLS-R. Next, we examine the value of using the local search in SGLS to improve the solution from LOSM, i.e. comparing the objective value of LOSM-R and SGLS-R. The experiments show that with $\delta \leq 6$, both approaches have zero violations for all the runs. This is an interesting result, as LOSM with $TG = 6$ provides protection to uncertainty only up to $\delta = 3$. Note that six time periods equate to two days. Hence, the experiments show that LOSM is able to generate good solutions when the service arrival schedule is off by two days. For $\delta > 6$, we start to see violations to the activity thresholds. However, we note that SGLS-R performs better than LOSM-R, i.e. the local search has improved the nimbleness of the solution generated by LOSM. However, due to the difference between LOSM-R and SGLS-R being not large, and with high standard deviations, we are not able to show, using Student's t-test, that SGLS-R is significantly better.

In conclusion, the experiments show that the nimble yard plan template followed by a recovery stage with strict assignment policy is able to achieve zero violation when service arrival schedule is off by two days. For scenarios where the uncertainty of service arrival schedule is beyond two days, SGLS-R is a preferred approach as it gives solutions with lower violations, both in average and worst case. However, statistically, the results fail to show significant difference between SGLS-R and LOSM-R.

5.2.4 Solution approaches for case 2: one change policy

Stage 2 - a recovery algorithm RA-1

Recall that in the recovery algorithm, RA, presented earlier, the new u is the same as the given u^* in Stage 1. With the one change policy, the recovery algorithm will be slightly different such that we limit the number of changes in the assignment of services to ranges to one, and hence we need to derive an expression to measure the number of changes and bound the value in a constraint. We modify the static model for this purpose as follows.

$$\mathbb{F}(u, v|\ u^*, \omega) \ = \ \left[\begin{array}{c} \mathbb{C}^T v \\ s.t. \quad \mathbb{A}(\omega) \begin{pmatrix} u \\ v \end{pmatrix} \geq \mathbb{B} \\ \sum_{r \in \mathcal{R}} (\sum_{i \in \mathcal{V} \wedge (u^*_{ir}=1)} (u^*_{ir} - u_{ir}) + \\ \sum_{i \in \mathcal{V} \wedge (\sum_{i \in \mathcal{V}} u^*_{ir}=0)} (u_{ir})) \leq 1 \end{array} \right] \tag{5.23}$$

The derivation is as follows. For a given range r, if it is assigned to service i in Stage 1, then $u^*_{ir} = 1$. In the Stage 2 solution, if $u_{ir} = 0$, it means that the assignment of range r to service i has been changed in Stage 2, and hence the term $\sum_{i \in \mathcal{V} \wedge (u^*_{ir}=1)} (u^*_{ir} - u_{ir})$ will be set to 1 if this happens. Next, if range r is not assigned to any service in Stage 1, then $\sum_{i \in \mathcal{V}} u^*_{ir} = 0$. In Stage 2, if $u_{ir} = 1$ for some i, it means that the range r is now assigned and hence there is a change. The term $\sum_{i \in \mathcal{V} \wedge (\sum_{i \in \mathcal{V}} u^*_{ir}=0)} (u_{ir})$ will then be set to 1 if this happens. Summing the above two expressions over all $r \in \mathcal{R}$ will give the

total changes in assignment of ranges to services from u^* to u, and hence the constraint $\sum_{r\in\mathcal{R}}\left(\sum_{i\in\mathcal{V}\wedge(u^*_{ir}=1)}(u^*_{ir}-u_{ir})+\sum_{i\in\mathcal{V}\wedge(\sum_{i\in\mathcal{V}}u^*_{ir}=0)}(u_{ir})\right)\leq 1$ ensures the number of changes is limited to one.

However, this model took hours to solve in CPLEX. In order to conduct meaningful experiments, we propose a new recovery algorithm, such that it will give a solution in a shorter amount of time, but possibly with a sub-optimal solution. The proposed recovery algorithm is made up of two steps. In the first step, we run the model RA (assuming u = u^*). Recall that in the model, k_{rt} are decision variables that encode the number of violations of the activity concentration thresholds in range r at time t. We let k^*_{rt} be the values of k_{rt} of this step 1 solution. In step 2, we run a model similar to equation 5.23, with an additional constraint as follows.

$$\mathbb{F}(u,v|\ u^*,\omega)\ =\ \left[\begin{array}{c}\mathbb{C}^T v\\ s.t.\quad \mathbb{A}(\omega)\begin{pmatrix}u\\ v\end{pmatrix}\geq\mathbb{B}\\ \sum_{r\in\mathcal{R}}(\sum_{i\in\mathcal{V}\wedge u^*_{ir}=1}(u^*_{ir}-u_{ir})+\\ \sum_{i\in\mathcal{V}\wedge\sum_{i\in\mathcal{V}}u^*_{ir}=0}(u_{ir}))\leq 1\\ u_{ir}=u^*_{ir}\quad \forall i\in\mathcal{V},r\in\mathcal{R}\wedge(\sum_{t\in\mathcal{T}}k^*_{rt}=0)\end{array}\right] \tag{5.24}$$

The additional constraint $u_{ir}=u^*_{ir}$ applies only for r such that $\sum_{t\in\mathcal{T}}k^*_{rt}=0$, i.e those ranges r that do not have any violations in step 1 will not quality to be changed in Stage 2. Therefore, this limits the Stage 2 solution such that only those ranges r satisfying $\sum_{t\in\mathcal{T}}k^*_{rt}>0$ are allowed to change their assignments, otherwise, they have

to be the same as that in the Stage 1 solution. The rationale behind this constraint is that if a range does not have activity concentration violations, then the Stage 1 assignment for this range is probably not vulnerable to the uncertainty ω, and hence the assignment should be retained. We call this proposed recovery algorithm for one change policy **RA-1**.

Stage 1 - nimble solution assuming RA-1

We evaluate the possibility of using SGLS to solve Stage 1. We find that each run of RA-1 can take 30-60 seconds. The Evaluate*(u) function will in turn run RA-1 50 times for the 50 randomly generated ω. Using RA-1 in the Evaluate*(u) function makes the evaluation of the neighbourhood $N(u)$ in the local search extremely slow. In order to speed up the local search, we modified the SGLS to only allow some neighbours to be generated.

We recall that a neighbour of u is generated by swapping the assignment of any two ranges as follows. We let $(a_1, a_2, \ldots, a_{|\mathcal{R}|})$ represent the assignment of u and a neighbour to the solution u is defined to have a pair of assignment swapped, i.e. the values of a_r and $a_{r'}$ are exchanged, for some r and r' in $\mathcal{R}$. Instead of considering all possible neighbours of u to be in $N(u)$, we choose neighbours into $N(u)$ as follows. We first evaluate u by generating 50 random ω and solve each case with RA-1. The case that gives the maximum objective value for RA-1 is chosen as the worst solution. We let k^*_{rt} be the values of k_{rt} of this solution. A neighbour u' (with values of a_r and $a_{r'}$ interchanged) is chosen into $N(u)$ only if either $\sum_{t\in\mathcal{T}} k_{rt} > 0$ or $\sum_{t\in\mathcal{T}} k_{r't} > 0$. The rationale is that exchanging a_r and $a_{r'}$ is not likely to improve the solution if both

$\sum_{t\in\mathcal{T}} k_{rt} = 0$ and $\sum_{t\in\mathcal{T}} k_{r't} >= 0$, as both ranges do not have the problem of violating the activity concentration thresholds. We call this algorithm Selective-SGLS as we are selective in the choice of neighbours in $N(u)$. Selective-SGLS algorithm is almost exactly like the SGLS as shown in Figure 5.3 (flowchart) and Figure 5.4 (pseudo code), except that, instead of RA, RA-1 is used, and the neighbourhood set $N(u)$ is chosen selectively based on k^*_{rt} .

Experimental results

The experiment set up is exactly the same as that used for the experiments for SGLS-R, as described in Section 5.2.3. We run Selective-SGLS for Stage 1 and RA-1 for Stage 2. We label this approach as **SGLS-R1**. As SGLS-R1 solves for Stage 1 non-deterministically using simulations in the local search, the experiment set for SGLS-R1 is replicated 10 times, and the means and standard deviations of the 10 replications are computed. The details of the average and worst case objective values for each of the 10 runs are tabulated in Table 5.6 and 5.7, respectively. Table 5.5 tabulates the results for SGLS-R1, and for convenience of comparison with SGLS-R, the results for SGLS-R are also tabulated.

We compare the quality of the solutions of a recovery policy where we allow more changes in the recovery stage (SGLS-R1), with one that has more restriction (SGLS-R). The average and worst case violation of SGLS-R and SGLS-R1 are compared using the corresponding objective values (violations) with the same δ.

For the case with $\delta = 8$, experiments show that SGLS-R1 is better since it is able to find solutions with zero violations with the recovery algorithm RA-1. With $\delta = 10$, the

Table 5.5: Experimental results comparing SGLS-R versus SGLS-R1 : average and worst case violation

δ	Average Violation				Worst Case Violation			
	SGLS-R		SGLS-R1		SGLS-R		SGLS-R1	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	0.42	0.50	0.00	0.00	13.60	16.02	0.00	0.00
9	0.45	0.56	0.72	0.25	15.60	16.24	24.10	0.32
10	0.76	0.78	0.58	0.13	29.00	24.17	24.50	0.53

Table 5.6: Detailed results of 10 replications of SGLS-R1 experiments - average violation

δ	Experiments sets											
	run 1	run 2	run 3	run 4	run 5	run 6	run 7	run 8	run 9	run 10	Mean	Std Dev
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	1	1	1	0	1	0	1	1	1	1	0.72	0.25
10	1	1	0	1	1	1	0	0	1	1	0.58	0.13

average and worst case results also show that SGLS-R1 is better with lower violations.

However, for $\delta = 9$, SGLS-R1 gives solutions with higher violations in both the average

Table 5.7: Detailed results of 10 replications of SGLS-R1 experiments - worst case violation

	Experiments sets											
δ	run 1	run 2	run 3	run 4	run 5	run 6	run 7	run 8	run 9	run 10	Mean	Std Dev
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	25	24	24	24	24	24	24	24	24	24	24.10	0.32
10	24	25	24	24	25	25	25	25	24	24	24.50	0.53

and worst case. Although we expect the one change policy to be more superior than strict assignment policy (with less violations), as we allow one of the assignments to be changed, we note that in our implementation, we compromise the optimality of the solutions for better run time as follows. Firstly, we have used the RA-1 recovery algorithm, which disallows some of the changes in u to be considered in Stage 2, and that could have given us sub-optimal solutions for these cases. Secondly, we have used Selective-SGLS, where some of the neighbours that could have helped to find better solutions are not evaluated. Hence, the algorithm could have missed out the better solutions due to these reasons. Nonetheless, as the differences between SGLS-R and SGLS-R1 are not large and with high standard deviation, we are not able to show, using Student's t-test, any significant difference between SGLS-R and SGLS-R1 for the cases with $\delta = 9$ and $\delta = 10$.

In conclusion, the experiments show that allowing one of the assignments to change,

as in the one change policy, gives a better solution than that with strict assignment policy. However, due to run time issues, we compromise on the solution and we are not able to show that the one change policy is better in one of the cases. We also note that the problem gets harder to solve when we allow more changes to the solution in the recovery stage, i,e. increase flexibility in the recovery stage. This will be discussed further in the next section.

5.2.5 Case 3: generic nimble yard plan template problem

In Section 5.2.3, we consider the case where u cannot be changed in the Stage 2 recovery algorithm. In Section 5.2.4, we consider the case where one of the assignments in u can be changed in the Stage 2 recovery algorithm, and there is no cost associated to this change. In this section, we consider the most generic case of the nimble yard template problem, where there is no limit to the changes in u in Stage 2. However, we measure the cost of the change and that has to be minimised in the objective.

Problem definition

We let the function $\triangle\left((u^*\ v^*)^T,(u\ v)^T\right)$ denote the cost of changing from a Stage 1 solution $(u^*\ v^*)^T$ to a Stage 2 solution $(u\ v)^T$. A formulation of the general Stage 2

recovery algorithm could be

$$\mathbb{F}(u, v|\ u^*, v^*, \omega) \ = \ \left[\begin{array}{l} \mathbb{C}^T v + \triangle \left((u^*\ v^*)^T, (u\ v)^T \right) \\ \\ \text{subject to} \\ \\ \mathbb{A}(\omega) \begin{pmatrix} u \\ \\ v \end{pmatrix} \geq \mathbb{B} \end{array} \right] \tag{5.25}$$

Hence, the corresponding nimble yard plan template problem will be defined as follows.

$$\underset{(u^*\ v^*)^T}{\text{Minimise}} \ \max_{\omega} \ \min_{(u\ v)^T} \ \mathbb{F}(u, v|\ u^*, v^*, \omega) \tag{Model V.VI}$$

Note that this formulation is in the form of the generic nimble optimisation.

Complexity of the problem

We attempt to give a intuitive explanation to the complexity of the problem. We can see that the formulation is a generalisation of the strict assignment policy and the one change policy. For the case of strict assignment policy, $\triangle((u^*\ v^*)^T, (u\ v)^T)$ is zero when $u^* = u$, and infinity otherwise. For the case of one change policy, $\triangle((u^*\ v^*)^T, (u\ v)^T)$ is zero when u^* and u differs by one of the assignment.

However, the above two cases consider only feasibility, and hence in the solution approach, they are modelled into the constraints of the recovery algorithm. For the case of strict assignment policy, the optimum solution for RA can be solved in reasonable time.

However, the mathematical programme that models the exact one change policy is not able to terminate within the time-out limit of eight hours, and hence a restricted version RA-1 is used in the experiments, where some possible solutions are not considered. Hence RA-1 may not give the optimum solution based on the exact one change policy. Note also that even with this compromise in the RA-1, we have to also compromise the local search with a Selective-SGLS where some neighbours are not evaluated.

In this most generalised case, $\triangle((u^* \ v^*)^T, (u \ v)^T)$ may not be just zero or infinity, but can be a complex non-linear relationship between $(u^* \ v^*)^T$ and $(u \ v)^T$. As we have seen in solution approaches for the two cases, it gets harder to solve for the recovery algorithm when the changes in Stage 2 are more flexible (the model gets more complex). Hence we think that the generic nimble yard plan template problem is a very challenging problem to tackle, and we leave this for future research work. Nonetheless, we think that the simulation guided local search architecture on a distributed computing environment is likely to be the direction to pursue.

5.3 Summary

In this chapter, we consider a container terminal that adopts a consolidated and dedicated yard plan strategy. With vessel arrival schedule being uncertain, we would like to have a yard plan template that can be changed easily to an optimal solution after the uncertainty is revealed. We call this the nimble yard plan template problem.

We first introduce a generic nimble optimisation formulation. It is interesting to note that this problem has never been addressed in the literature, except in some related

works where we show that our formulation is a generalisation of some of their models. We also propose a general method for solving this class of problems, and also propose that we could use distributed architecture (such as cloud computing) to implement the solution architecture.

As there can be many variations of the nimble yard plan template problem, we first focus on two cases, where the recovery stage adopts a strict assignment policy, and a one change policy, respectively. We present solution approaches for both cases. We show via experiments that a nimble yard plan template with a recovery policy in Stage 2 gives very much lower violations compared to an approach that is not nimble and does not recover the solution after the uncertainty is revealed. We also show that a one change policy generally gives lower violations compared to the strict assignment policy, but in one instance, our solution method fails to find the optimum solution. Lastly, we define a generic nimble yard plan template problem as the third case, and provide some intuition of the complexity of the problem.

Chapter 6

Conclusion

Container terminals must be efficient in order to meet the shipping community's demand. For container terminals to be efficient, relying on quay-side technological advancements like better and faster cranes is not enough. Movement of containers into and out of the storage yard must be as efficient as possible, and the most important factor in determining yard storage efficiency is the yard plan. We focus most part of the thesis on solving a yard plan template - a plan that repeats on a weekly basis. We first focus on a static problem, and then move on to a problem where input could be uncertain. We find that the yard plan template solution has to be "nimble", i.e. agile to changes when the uncertainty is revealed. Motivated by the nimble yard plan template problem, we define a generic nimble optimisation formulation, and some preliminary research work is presented.

6.1 Thesis achievements and contributions

The following points summarise what we have achieved and our contribution in this thesis:

1. We have provided good insight into why container terminals need to have an optimum yard plan in order to improve their efficiency at the quay side, despite new quay cranes technology is becoming available for quay cranes to lift multiple containers at the same time. It is found that the bottleneck could be transferred to the storage and retrieval efficiency in the yard. Storage and retrieval are slowed down due to violations of the thresholds for activity concentration and yard crane contention. These violations can be minimised with a optimised yard plan.

2. We present a very comprehensive literature review of various areas of research pertaining to container terminal operations, ranging from yard planning, trucks and yard cranes routing and scheduling, berth planning, quay cranes scheduling and stowage planning. We also review works on container terminal simulations, and also a couple of works on terminal handling capacity, cost of operating the terminal, and manpower planning issues.

3. We have described how shipping lines operate their vessels by grouping them into services. Each service calls at a terminal on a weekly cycle, and on a fixed day of the week. This allows terminal to generate a yard plan that repeats on a weekly basis - a yard plan template. Terminals usually generate yard plan templates following a yard plan strategy. To help terminals to discover the best strategy for their scenario, we will require the yard plan template problem to be defined generically so that solutions that apply different strategies can be generated and compared. Realising that there is this need, we formulate a generic model that defines the yard plan template problem that is applicable to a wide variety of yard

plan strategies. To the best of our knowledge and from the papers reviewed, this has never been formulated before.

4. Search engines can be designed to discover the best yard plan strategy, and that requires the generic yard plan template problem to be solved in a reasonable amount of time. As the generic problem has different complexity when applied to different strategies, we propose to approach the problem by having different solution approaches for different strategies. We present solution approaches for two strategies, namely, consolidated and dedicated strategy, and consolidated and non-dedicated strategy, respectively. The first approach solves the model as a mathematical programme on CPLEX, and we show that re-modelling some of the constraints reduces the run time tremendously. The second approach is a heuristic local search algorithm, and we show that the run time is extremely fast compared to the mathematical programme. However, the experiments show that it does not give the optimum solution in one case.

5. We note that uncertainty of events prevails in reality. We learn from a global terminal operator that changes to the vessels arrival schedule have the greatest impact to operation efficiency for a terminal that adopts the consolidated strategy. We have defined a nimble yard plan template problem, where the objective is not to find a yard template that is robust against vessel arrival uncertainty, but a nimble yard template that is agile to change to a new solution “easily” when the uncertainty is revealed. We explore three variations of the nimble yard plan template problem. Heuristic algorithms based on local search are presented to solve

the first two cases. Experiments show that nimble solutions are better for situations with vessel arrival uncertainty. Experiments also show that a more flexible recovery policy gives a better solution in most of the cases, but it gets harder to solve. The third case is the most generic form of the nimble yard plan template problem, and we give some intuition of the complexity of solving such a problem.

6. In many real world problems, often we are forced to make an initial decision, but at a later time, we can change the decision when new information is available, but this change incurs additional cost, either wasted money or effort of rework. Hence, the initial decision is deemed to be nimble, and nimble yard plan template problem is an example. Motivated by the nimble yard plan template problem, we have defined a generic formulation for Nimble Optimisation. We find that there are no research papers that have formulated the nimble optimisation problem. By defining the generic problem formulation, it allows the non-container terminal operation community to also explore if any other problems can be classified into this class of problems, hence allowing a bigger research community to be exposed to this problem. A distributed solution architecture to solve this problem is presented. We also show that it is a generalisation of some of the works we have reviewed.

6.2 Major limitations of the model

While we have defined the scope of the study for the purpose of this dissertation, various assumptions are made, and hence the model presented is not without limitations. The following summarises the major limitations of the model.

1. The yard plan template problem model, presented in this thesis, only applies to terminals with RMG fixed to the blocks. Allowing dynamic yard crane movement across blocks can be modelled, but we find that the problem is too complex to be solved.

2. The model also assumes that the terminal adopts immediate final grounding. Allowing containers to be moved to another location later means that we have to augment the yard planning problem with a scheduling problem of deciding when each container will be moved from one yard location to another. This is again too complex to be solved.

3. In the nimble optimisation (Nimo) problem definition, we limit the decision making to be done in two stages, but in real life, decisions in the face of uncertainty are done in multiple event triggered stages. However multi-stage problems are highly intractable, and hence we limit out model to two stages.

4. In the nimble yard plan template problem, we limit the uncertainty to the service arrival schedule. We reasoned that this is the major factor causing yard activity violation. However, a more in depth study with the combination of other uncertainty factors can be explored.

6.3 Future research direction

We are still faced with a few un-solved problems. In the static yard plan template problem, we have not explored solution approaches for other strategies such as consolidated

strategy with variable size clusters. For strategies with clusters, we do not think a mathematical programme is able to solve it, and hence heuristic algorithms are needed. For the nimble yard plan template problem, more work can be done to explore different search methods, like the genetic algorithm or simulated annealing. Distributed computing can be explored to distribute the runs for the evaluation of the solutions with simulation. For the nimble optimisation (Nimo) model, it may lead to many interesting research work on solving the problem, either special cases, or solving the general problem with a general framework. Lastly, Nimo can be applied to almost every real life problem, as uncertainty prevails in real life, and we are always constantly adapting to new changes.

Bibliography

D. Ambrosino, et al. (2004). 'Stowing a containership: the master bay plan problem'. *Transportation Research Part A* **38A**:81–99.

A. Ben-Tal, et al. (2006). 'Extending scope of robust optimization: comprehensive robust counterparts of uncertain problems'. *Mathematical Programming Series B* **107**(1):63–89.

A. Ben-Tal, et al. (2004). 'Adjustable Robust Solutions of Uncertain Linear Programs'. *Mathematical Programming Series A* **99**:351–376.

A. Ben-Tal & A. Nemirovski (1997). 'Robust truss topology design via semidefinite programming'. *SIAM Journal on Optimization* **7**(4):991–1016.

A. Ben-Tal & A. Nemirovski (1998). 'Robust convex optimization'. *Mathematics of Operations Research* **23**(4):769–805.

A. Ben-Tal & A. Nemirovski (1999). 'Robust solutions of uncertain linear programs'. *Operations Research Letters* **25**(1):1–13.

A. Ben-Tal & A. Nemirovski (2000). 'Robust solutions of Linear Programming problems contaminated with uncertain data'. *Mathematical Programming Series A* **88**(3):411–424.

M. Benacchio, et al. (2001). 'Cost, Benefits and Pricing of Dedicated Container Terminals'. *The 9th world Conference on Transportation Research, Seoul, Korea* .

D. Bertsimas & C. Caramanis (2010). 'Finite Adaptability in Multistage Linear Optimization'. *IEEE Transactions on Automatic Control* **55**(12):2751–2766.

D. Bertsimas & M. Sim (2004). 'The Price of Robustness'. *Operations Research* **52**(1):35–53.

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

E. K. Bish, et al. (2007). 'Dispatching Vehicles in a Mega Container Terminal'. *Container Terminals and Cargo Systems, Design, Operations Management and Logistics Control Issues* pp. 179–194.

G. Bontempi, et al. (1997). 'Simulation and Optimization for Management of Intermodal Terminals'. *European Simulation Multi-conference* .

A. Bruzzone & R. Signorile (1998). 'Simulation and Genetic Algorithms for ship planning and shipyard layout'. *Simulation* **71**(2):74–83.

B. Casey & E. Kozan (2006). 'A container storage handling model for high tech automated multimodal terminals'. *Proceedings of the second International Intelligent Logistics Systems Conference* pp. 10.1 – 10.19.

C. Chen (2003). 'Simulation and Optimization of Container Yard Operations: a survey'. *Maritime Tehnology* **10-12 Sep Singapore**.

C. Y. Chen, et al. (2000). 'A time-space network model for the space resource allocation problem in container marine transportation'. *17th international symposium on mathematical programming* .

T. Chen (1999). 'Yard operations in the container terminal - a study in the "unproductive moves"'. *Maritime Policy and Management* **26**(1):27–38.

X. Chen & Y. Zhang (2009). 'Uncertain linear programs: extended affinely adjustable robust counterparts'. *Operations Research* **57**(6):1469–1482.

R. K. Cheung, et al. (2002). 'Inter block crane deployment in container terminals'. *Transportation Science INFORMS* **36**(1):79–93.

S. Choo (2006a). 'The Crane Split and Sequencing Problem with Clearance and Yard Congestion Constraints in Container Terminal Ports'. *NUS Master Thesis, Singapore-MIT Alliance* .

S. Choo (2006b). 'A Multi-agent Framework Applied to the Quay Crane Scheduling Problem in Container Terminal Ports'. *MIT Master Thesis, Singapore-MIT Alliance* .

Y.-C. Chung, et al. (1988). 'A simulation analysis for a transtainer-based container handling facility'. *Computers and Industrial Engineering* **14**:113–115.

J. Dai, et al. (2007). 'Berth Allocation Planning Optimization in Container Terminals' **119**:69–104.

S. K. Das & L. Spasovic (2003). 'Scheduling material handling vehicles in a container terminal'. *Production Planning and Control* **14**(7):623–633.

B. de Castilho & C. F. Daganzo (1993). 'Handling strategies for import containers at marine terminals'. *Transportation Research Part B* **27B**:151–166.

R. Dekker, et al. (2006). 'Advanced methods for container stacking'. *OR Spectrum* **26**(4):563–586.

Y. Du, et al. (2010). 'A feedback procedure for robust berth allocation with stochastic vessel delays'. *Proceedings of the 8th World Congress on Intelligent Control and Automation (WCICA)* pp. 2210–2215.

M. K. Fung, et al. (2003). 'The impact of terminal handling charges on overall shipping charges: an empirical study'. *Transportation Research Part A* **37A**:703–716.

L. Gambardella, et al. (1998). 'Simulation and planning of an intermodal terminal'. *Special issue of Simulation Journal in Harbor and Maritime Simulation* **21 Aug**(2):107–116.

L. M. Gambardella, et al. (2001). 'An optimization methodology for intermodal terminal management'. *Journal of Intelligent Manufacturing* **12**:521–534.

L. M. Gambardella & A. E. Rizzoli (2000). 'The role of simulation and optimization in intermodal container terminals'. *ESS2000, European Simulation Symposium* **Sep**.

H.-O. Guenther, et al. (2006). 'Simulation of transportation activities in automated seaport container terminals'. *Proceedings of the second International Intelligent Logistics Systems Conference* pp. 3.1–3.15.

X. Guo, et al. (2009). 'A simulation based hybrid algorithm for yard crane dispatching in container terminals'. *Proceedings of the 2009 Winter Simulation Conference* pp. 2320–231.

X. Guo, et al. (2008). 'Yard crane dispatching based on real time data driven simulation for container terminals'. *Proceedings of the 2008 Winter Simulation Conference* pp. 2468–2655.

X. Han, et al. (2010). 'A proactive approach for simultaneous berth and quay crane scheduling problem with stochastic arrival and handling time'. *European Journal of Operational Research* **207**(3):1327–1340.

M. Hendriks, et al. (2010). 'Robust cyclic berth planning of container vessels'. *OR Spectrum* **32**(3):501–517.

L. Henessey, et al. (2004). 'Using simulation in evaluating berth allocation at a container terminal'. *3rd International Conference on Computer Applications and Information Technology in Maritime Industries* .

C.-I. Hsu & Y.-P. Hsieh (2005). 'Shipping economic analysis for ultra large containership'. *Journal of the Eastern Asia Society for Transportation Studies* **6**:936–951.

W.-C. Huang & C.-Y. Chu (2004). 'A selection model for in-terminal container handling systems'. *Journal of Marine Science and Technology* **12**(3):159–170.

M. Itmi, et al. (2000). 'A Simulation Methodology description of container placement using a unique system to control the logistics'. *Summer Computer Simulation Conference* .

M. Itmi, et al. (2001). 'Simulation operations policies in a container terminal'. *Summer Computer Simulation Conference* .

P. Jarjoui, et al. (2001). 'Simulation of a Multi-agent system for container terminal activities co-operation of handling teams'. *Summer Computer Simulation Conference* .

J. Kang, et al. (2006). 'Deriving stacking strategies for export containers with uncertain weight information'. *Journal of Intelligent Manufacturing* **17**(4):399–410.

M. Kia, et al. (2002). 'Investigation of port capacity under a new approach by computer simulation'. *Computers and Industrial Engineering* **42**:533–540.

K. H. Kim (1997). 'Evaluation of the number of rehandles in container yards'. *Computers Industrial Engineering* **32**(4):701–711.

K. H. Kim & J. W. Bae (1998). 'Re-marshalling export containers in port container terminals'. *Computers Industrial Engineering* **35**(3-4):655–658.

K. H. Kim & H. B. Kim (1999a). 'Segregating space allocation models for container inventories in port container terminals'. *International Journal of Production Economics* **59**:415–423.

K. H. Kim & H. B. Kim (2002). 'The optimal sizing of the storage space and handling facilities for import containers'. *Transportation Research Part B* **36B**:821–835.

K. H. Kim, et al. (2004). 'Operator-scheduling using a constraint satisfaction technique in port container terminals'. *Computers and Industrial Engineering* **46**:373–381.

K. H. Kim & K. Y. Kim (1999b). 'Routing straddle carriers for the loading operation of containers using a beam search algorithm'. *Computers and Industrial Engineering* **36**:109–136.

K. H. Kim, et al. (2003). 'Sequencing delivery and receiving operations for yard cranes in port container terminals'. *International Journal of Production Economics* **84**:283–292.

K. H. Kim & K. C. Moon (2003). 'Berth scheduling by simulated annealing'. *Transportation Research Part B* **37B**:541–560.

K. H. Kim & K. T. Park (2003). 'A note on a dynamic space-allocation method for outbound containers'. *European Journal of Operations Research* **148**:92–101.

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

K. H. Kim, et al. (2000). 'Deriving decision rules to locate export containers in container terminal'. *European Journal of Operations Research* **124**:89–101.

K. Y. Kim & K. H. Kim (1997). 'A routing algorithm for a single transfer crane to load export containers onto a containership'. *Computers Industrial Engineering* **33**(3-4):673–676.

L. P. Ku, et al. (2010). 'An optimisation framework for yard planning in a container terminal: case with automated rail-mounted gantry cranes'. *OR Spectrum* **32**:519–541.

B. C. Kulick & J. T. Sawyer (1999). 'Flexible interface and architechture for container and intermodal freight simulations'. *Proceedings of the 1999 Winter Simulation Conference* .

L. H. Lee, et al. (2006). 'An optimization model for storage yard management in transhipment hubs'. *OR Spectrum* **28**(4):539–561.

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

P. Legato & M. F. Monaco (2004). 'Human resources management at a marine container terminal'. *European Journal of Operational Research* **156**:769–781.

M. Levinson (2006). *The Box - How the Shipping Container Made the World Smaller and the World Economy Bigger*. Princeton University Press.

C.-L. Li & G. L. Vairaktakakis (2004). 'Loading and unloading operations in container terminals'. *IIE Transactions* **36**:287–297.

C. Liebchen, et al. (2009). 'Robust and Online Large-Scale Optimization'. chap. The Concept of Recoverable Robustness, Linear Programming Recovery, and Railway Applications, pp. 1–27. Springer-Verlag, Berlin, Heidelberg.

A. Lim, et al. (2002). 'Crane Scheduling using Tabu search'. *IEEE* pp. 146–153.

R. J. Linn & C. Q. Zhang (2003). 'A heuristic for dynamic yard crane deployment in a container terminal'. *IEE Transactions* **35**:161–174.

P. J. Meersmans & R. Dekker (2001). 'Operations Research suppports container handling'. Tech. Rep. EI 2001-22, Econometric Institute, Erasmus University Rotterdam.

P. J. Meersmans & A. P. Wagelmans (2001). 'Dynamic scheduling of handling equipment at automated container terminals'. Tech. Rep. EI 2001-33, Econometric Institute, Erasmus University Rotterdam.

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

K. G. Murty, et al. (2005a). 'A decision support system for operations in a container terminal'. *Decision Support Systems* **39**(3):309–332.

K. G. Murty, et al. (2005b). 'Hong Kong International Terminal gains Elastic Capacity using a Data-intensive Decision Support System'. *Interfaces* **35**(1):61–75.

A. Narasimhan & U. S. Palekar (2002). 'Analysis and Algorithms for the transtainer routing problem in container port operations'. *Transportation Science* **36**(1):63–78.

W. Ng (2005). 'Crane scheduling in container yards with inter-crane interference'. *European Journal of Operational Research* **164**(1):64–78.

J. Ottjes, et al. (1994). 'A simulation model of a sailing container terminal service in the port of Rotterdam'. *Proceedings of the Conference on Modelling and Simulation* **Jun**.

J. A. Ottjes, et al. (1996). 'Robotised inter terminal transport of containers: a simulation study at the Rotterdam port area'. *Proceedings of the 8th European Simulation Symposium ESS* **Oct**.

M. E. Petering & K. G. Murty (2006). 'Simulation analysis of algorithms for container storage and yard crane scheduling at a container terminal'. *Proceedings of the second International Intelligent Logistics Systems Conference, Brisbane, Australia* pp. 19.1–19.15.

M. Rebollo, et al. (2000). 'A MAS approach for port container terminal management'. *Proceedings of the 3rd IBEROAMERICAN Workshop on DAI - MAS* pp. 83–94.

M. Rida, et al. (2003). 'Object oriented approach and java-based distributed simulation for container terminal operation management'. *International Conference on Information Systems and Engineering* .

A. E. Rizzoli, et al. (1999). 'Simulation for the evaluation of optimized operations policies in a container terminal'. *HMS99, Maritime and Industrial Logistic Modeling and Simulation* **Sep**.

C. D. Rudolft (2007). 'Ship-to-Shore productivity: an it keep up with mega-ship size increase? Part 1'. *Port Technology International* **Apr**:83.

Y. A. Saanen & R. Dekker (2007a). 'Intelligent stacking as way out of congested yards? Part 1'. *Port Technology International* **31st**:87–92.

Y. A. Saanen & R. Dekker (2007b). 'Intelligent stacking as way out of congested yards? Part 2'. *Port Technology International* **32nd**:80–85.

B. Schroer, et al. (2004). 'Using simulation to evaluate the expansion of terminal and intermodal operations at a deepwater port'. *Summer Computer Simulation Conference* .

A. Shabayek & W. Yeung (2002). 'A simulation model for the Kwai Chung container terminals in Hong Kong'. *European Journal of Operational Research* **140**:1–11.

A. Soyster (1973). 'Convex Programming with set-inclusive constraints and applications to inexact linear programming'. *Operations Research* **21**(5):1154–1157.

R. Stahlbock & S. Voβ (2008). 'Operations Research at Container Terminal: a Literature Update'. *OR Spectrum* **30**:1–52.

D. Steenken, et al. (2004). 'Container terminal operation and operations research - a classsification and literarture review'. *OR spectrum* **26**:3–49.

D. Steenken, et al. (2001). 'Stowage and Transport Optimization in Ship Planning'. *Online Optmization of Large Scale Systems* pp. 731–745.

N. N. Taleb (2007). 'The Black Swan: The Impact of the Highly Improbable' .

M. Taleb-Ibrahimi, et al. (1993). 'Storage space vs handling work in container terminals'. *Transportation Research Part B* pp. 13–32.

C. J. Tong, et al. (1999). 'Ant colony Optimization for the Ship Berthing Problem'. *ASIAN'99* pp. 359–370.

J. Tongzon (2001). 'Efficiency measurement of selected Australian and other international ports using data envelopment analysis'. *Transportation Research Part A* **35A**:107–122.

I. Vis, et al. (2001). 'Determination of the number of automated guided vehicles required at a semi-automated container terminal'. *Journal of the Operational Research Society* **52**:409–417.

I. F. Vis & R. de Koster (2003). 'Transhipment of containers at a container terminals: An overview'. *European Journal of Operational Research* **147**:1–16.

I. F. A. Vis, et al. (2005). 'Minimum Vehicle Fleet Size under time-window contraints at a container terminal'. *Transportation Science* **39**(2):249–260.

I. Wilson & P. Roach (2000). 'Container stowage planning: a methodology for generating computerised solutions'. *Journal of the Operational Research Society* **51**:1248–1255.

A. Wong & E. Kozan (2006). 'An integrated approach in optimising container process at seaport container terminals'. *Proceedings of the second International Intelligent Logistics Systems Conference* pp. 23.1–23.13.

H. Wren (2007). 'a solution to the Emma challenge'. *Cargo Systems* **Jul**/**Aug**:43–45.

Y. Xu, et al. (2010). 'A Robust Integrated Approach to Yard Space Allocation and Crane Scheduling in Container Terminals'. *7th International Conference on Service Systems and Service Management* pp. 1–6.

W. Yun & Y. Choi (1999). 'A simulation model for container-terminal operation analysis using an object oriented approach'. *International Journal of Production Economics* **59**:221–230.

C. Zhang, et al. (2003). 'Storage space allocation in container terminals'. *Transportation Research Part B* **37B**:883–903.

ZPMC (2007). 'ZPMC sets sights on quadruple-40 crane'. *Cargo Systems* **Jan**/**Feb**:19.

Appendix A

Nimble Optimisation – A Generalisation of Some Problems

To the best of our knowledge, we do not find any work that has defined their formulations in the same way as Nimo. However, some of them have similar problem structures as Nimo. We prove that Nimo is a generalisation of these problems by defining a transformation of an instance of that problem to an instance of Nimo problem, and the solution of Nimo is then transformed back to the solution of the original problem. This not only shows that Nimo is a generalisation of these problems, it also shows that it is equal or harder to solve than these problems (i.e. if there is a solution method for Nimo, then we have a solution method for those problems too). If those problems are proven to be NP hard, Nimo is also NP hard.

We first give an overview of the approach that we undertake, followed by the proofs.

A.1 Overview of the approach

All the following proofs follow the same structure, and we explain them here for better clarity. For each proof, we will first explain how a problem instance of the problem will be transformed to a problem instance of Nimo. The proof then proceeds as follows.

1. We will first cover the case where Nimo finds no feasible solution (objective value is ∞), and we show that no feasible solution can also be found in the original problem too.

2. We then cover the case where a feasible optimum solution is found in Nimo and we map this solution to the solution in the original problem.

   (a) We first show that this corresponding solution in the original problem is feasible.

   (b) We then show that this corresponding solution in the original problem is optimum. We show this by contradiction as follows. We first assume that it is not optimum and there exist a solution that has lower objective value in the original problem. We then map this solution back to Nimo and show that there will be a solution in Nimo that has lower objective value than the known optimum in Nimo, and hence this contracts the fact that Nimo has found a feasible optimum solution. We can then conclude that the assumption that there exists a solution that has lower objective value in the original problem cannot hold.

A.2 Ben-Tal's model (ARC)

Ben-Tal et al. (Ben-Tal et al. 2004) introduce the Adjustable Robust Counterpart (ARC), and they separate the decision variables to those that are made in stage 1 and those that are made in stage 2. It is not presented that stage 2 is a recovery of stage 1, and the decision variables of stage 1 and stage 2 may not have the same dimensions. The objective function only includes the stage 1 decision variables, and hence we deduce that they do not measure the differences between stage 1 and stage 2 decision variables. We prove that Nimo is a generalisation of this problem as follows.

Mathematically, ARC is defined as:

$$\underset{\forall x \in \Re^n}{\text{Minimise}} \;\; c^T x$$

$$s.t. \;\; \exists y(\omega) \;\; A(\omega)x + B(\omega)y(\omega) \geq b(\omega) \;\; \forall \omega \in \Omega$$

We transform ARC to Nimo with the following transformation. We let

$$x_0 = \begin{pmatrix} x \\ \\ 0 \end{pmatrix}, \text{ and } \; x_1 = \begin{pmatrix} x \\ \\ y \end{pmatrix}$$

We also define

$$\Delta\left(\begin{pmatrix} x \\ \\ z \end{pmatrix}, \begin{pmatrix} x' \\ \\ y \end{pmatrix}\right) = \begin{cases} 0 & \text{if } \; x = x' \text{ and } z = 0 \\ \\ \infty & \text{otherwise} \end{cases}$$

Then the vector x in the solution $x_0 = \begin{pmatrix} x \\ \\ 0 \end{pmatrix}$ of the following Nimo is the solution x of the original ARC problem.

$$\underset{\forall x_0 \in \Re^n}{\text{Minimise}} \quad \max_{\forall \omega \in \Omega} \quad \min_{\forall x_1(\omega) \in \Re^n} \left[\begin{array}{l} \left(c^T \quad 0 \right) x_1(\omega) + \Delta(x_0, x_1(\omega)) \\ \text{subject to} \\ \left(A(\omega) \quad B(\omega) \right) x_1(\omega) \geq b(\omega) \end{array} \right]$$

Proof:

Let $x_0^* = \begin{pmatrix} x^* \\ \\ z \end{pmatrix}$ be the optimum solution for Nimo. The transformation says that the optimum solution in problem ARC is x^*.

1. If the objective value of x_0^* is ∞, and z could be forced to zero without violating any constraints, it means that there does not exist x' such that $x_1(\omega) = \begin{pmatrix} x' \\ \\ y(\omega) \end{pmatrix} \forall \omega \in \Omega$, and the constraints $\left(A(\omega) \quad B(\omega) \right) x_1(\omega) \geq b(\omega)$ are satisfied. That is, there does not exist a x' and $y(\omega)$ such that $A(\omega)x' + B(\omega)y(\omega) \geq b(\omega)$. This will mean that there does not exist a feasible solution in ARC.

2. If the objective value is not ∞.

   (a) We first show that x^* is a feasible solution in ARC. Since the objective value is not ∞, $x_1(\omega)$ must take the form $\begin{pmatrix} x^* \\ \\ y(\omega) \end{pmatrix}$. Then, because the constraints in Nimo are satisfied, it can be re-expressed as $A(\omega)x^* + B(\omega)y(\omega) \geq b(\omega)$ and hence x^* is a feasible solution in ARC.

(b) Next we show that x^* is the optimum solution in ARC. We prove by contradiction by assuming that there exist another solution x' where the objective value of ARC with x' is smaller than that of x^*, i.e.,

$$c^T x' < c^T x^*$$

The objective value of Nimo with solution x_0^* is $\max\limits_{\forall \omega \in \Omega} \min\limits_{\forall x_1(\omega) \in \Re^n} \left(\begin{pmatrix} c^T & 0 \end{pmatrix} x_1(\omega) + \Delta\left(x_0^*, x_1(\omega)\right) \right)$. With the value of Δ as zero, and $x_1(\omega) = \begin{pmatrix} x^* \\ y(\omega) \end{pmatrix}$, the objective value simplifies to $c^T x^*$. Next, we let $x_0' = \begin{pmatrix} x' \\ 0 \end{pmatrix}$ and $x_1'(\omega) = \begin{pmatrix} x' \\ y'(\omega) \end{pmatrix}$ where $y'(\omega)$ take the same values as $y(\omega)$ in the feasible solution in ARC with solution x'. Since x' is a feasible solution in ARC, the constraints $A(\omega)x' + B(\omega)y'(\omega) \geq b(\omega)$ are satisfied, and it can be expressed as $\begin{pmatrix} A(\omega) & B(\omega) \end{pmatrix} x_1'(\omega) \geq b(\omega)$, and we get the feasible solution in Nimo. The objective value of Nimo with solution x_0' is $\max\limits_{\forall \omega \in \Omega} \min\limits_{\forall x_1(\omega) \in \Re^n} \left(\begin{pmatrix} c^T & 0 \end{pmatrix} x_1(\omega) + \Delta\left(x_0', x_1(\omega)\right) \right)$, which will be less than or equal to $\max\limits_{\forall \omega \in \Omega} \left(\begin{pmatrix} c^T & 0 \end{pmatrix} x_1'(\omega) + \Delta\left(x_0', x_1'(\omega)\right) \right)$. This can be simplified to $c^T x'$, and by our assumption, this is less than cx^*, the objective value of the optimum solution x_0^*. We have a contradiction where we have found a feasible solution of Nimo having a smaller objective value than the optimum, hence x^* is the optimum solution of ARC.

#

A.3 Liebchen's model (LRR)

Liebchen et al. (Liebchen et al. 2009) define the Linear Recovery Robust problem (LRR). They explicitly measure the cost of the difference between stage 1 and stage 2 decision variables, just like Nimo. They let $\hat{A}$ be a matrix called the recovery matrix, d be a vector called the recovery cost vector, and D be a non-negative number called the recovery budget. LRR is defined as follows.

$$\underset{x \in \Re^n}{\text{Minimise}} \quad c^T x$$

subject to

$$A(\omega_0)x \geq b(\omega_0)$$

$$\exists y(\omega) \in \Re^n \left[\begin{array}{l} A(\omega)x + \hat{A}y(\omega) \geq b(\omega) \\ d^T y(\omega) \leq D \end{array} \right] \quad \forall \omega \in \Omega$$

This is exactly the same model as Ben-Tal's ARC model with the following mapping. $\hat{A}$ is the matrix $B(\omega)$, where in this case $\hat{A}$ is static (not depending on ω). The constraints on $A(\omega_0)x \geq b(\omega_0)$ and $d^T y(\omega) \leq D$ and can be easily subsumed into the matrices $A(\omega), B(\omega)$ and vector $b(\omega)$. Since ARC and LRR are equivalent, having shown Nimo to be a generalisation of ARC, it follows that Nimo is also a generalisation of LRR.

#

A.4 Bertsimas's model (CARO and FARO)

Bertsimas and Caramanis (Bertsimas & Caramanis 2010) present the Adaptable Robust Optimisation problem that assumes a two stage decision making process, labelled as x and y respectively. Unlike ARC, both sets of decision variables are included in the constraints and objective function, and hence implicitly they can also measure the difference between the stage 1 and stage 2 decision variables. They define a Complete Adaptable Robust Optimisation (CARO) as:

$$\underset{\forall x \in \Re^n}{\text{Minimise}} \quad \max_{\forall \omega \in \Omega} \quad \min_{\forall y(\omega) \in \Re^n} \left[\begin{array}{l} c^T x + d^T y(\omega) \\ \text{subject to} \\ A(\omega)x + B(\omega)y(\omega) \geq b(\omega) \end{array} \right]$$

We show below that Nimo is a generalisation of CARO. It is also worth noting that since CARO has both stage 1 and 2 decision variables included in the constraints and objective, we can model a Nimo problem easily as CARO, and hence the proof that CARO is also a generalisation of Nimo is obvious. Therefore, CARO and Nimo are equivalent problems modelled differently.

They then propose a special case where there is a finite number of possible solutions of y. The problem then is to solve x and a set of $\{y_1, y_2, \ldots, y_k\}$. The problem is termed as Finite Adaptable Robust Optimisation (FARO). We also show below that Nimo is a generalisation of FARO.

CARO

We first consider the CARO. We transform CARO to Nimo with the following trans-

formation. We let

$$x_0 = \begin{pmatrix} x \\ \\ 0 \end{pmatrix}, \text{ and } x_1 = \begin{pmatrix} x \\ \\ y \end{pmatrix}$$

We also define

$$\Delta\left(\begin{pmatrix} x \\ \\ z \end{pmatrix}, \begin{pmatrix} x' \\ \\ y \end{pmatrix}\right) = \begin{cases} 0 & \text{if } x = x' \text{ and } z = 0 \\ \\ \infty & \text{otherwise} \end{cases}$$

Then the vector x in the solution $x_0 = \begin{pmatrix} x \\ \\ 0 \end{pmatrix}$ of the following Nimo is the solution x of the original CARO problem.

$$\underset{\forall x_0 \in \Re^n}{\text{Minimise}} \quad \max_{\forall \omega \in \Omega} \quad \min_{\forall x_1(\omega) \in \Re^n} \begin{bmatrix} \begin{pmatrix} c^T & d^T \end{pmatrix} x_1(\omega) + \Delta(x_0, x_1(\omega)) \\ \text{subject to} \\ \begin{pmatrix} A(\omega) & B(\omega) \end{pmatrix} x_1(\omega) \geq b(\omega) \end{bmatrix}$$

Proof:

Let $x_0^* = \begin{pmatrix} x^* \\ \\ z \end{pmatrix}$ be the optimum solution for Nimo. The transformation says that the optimum solution in problem CARO is x^*.

1. If the objective value of x_0^* is ∞, and z could be forced to zero without violating any constraints, it means that there does not exist x' such that $x_1(\omega) = \begin{pmatrix} x' \\ \\ y(\omega) \end{pmatrix} \forall \omega \in \Omega$, and the constraints $\begin{pmatrix} A(\omega) & B(\omega) \end{pmatrix} x_1(\omega) \geq b(\omega)$ are satisfied. That is, there does not exist a x' and $y(\omega)$ such that $A(\omega)x' + B(\omega)y(\omega) \geq b(\omega)$. This will mean

that there does not exist feasible solutions in CARO.

2. If the objective value is not ∞.

   (a) We first show that x^* is a feasible solution in CARO. Since the objective value is not ∞, $x_1(\omega)$ must take the form $\begin{pmatrix} x^* \\ y(\omega) \end{pmatrix}$. Then, because the constraints in Nimo are satisfied, they can be re-expressed as $A(\omega)x^* + B(\omega)y(\omega) \geq b(\omega)$ and hence x^* is a feasible solution in CARO.

   (b) Next we show that x^* is the optimum solution in CARO. We prove by contradiction by assuming that there exists another solution x' where the objective value of CARO with x' is smaller than that of x^*, i.e.,

$$\max_{\forall \omega \in \Omega} \left(\min_{\forall y(\omega) \in \Re^n} [c^T x' + d^T y(\omega)] \right) < \max_{\forall \omega \in \Omega} \left(\min_{\forall y(\omega) \in \Re^n} [c^T x^* + d^T y(\omega)] \right) \quad \text{(A.1)}$$

   The objective value of Nimo with solution x_0^* is $\max_{\forall \omega \in \Omega} \min_{\forall x_1(\omega) \in \Re^n} \left(\begin{pmatrix} c^T & d^T \end{pmatrix} x_1(\omega) + \Delta\left(x_0^*, x_1(\omega)\right) \right)$. With the value of Δ as zero, and $x_1(\omega) = \begin{pmatrix} x^* \\ y(\omega) \end{pmatrix}$, the objective value simplifies to $\max_{\forall \omega \in \Omega} \min_{\forall x_1(\omega) \in \Re^n} \left(\begin{pmatrix} c^T & d^T \end{pmatrix} \begin{pmatrix} x^* \\ y(\omega) \end{pmatrix} \right)$, and further simplifies to $\max_{\forall \omega \in \Omega} \min_{\forall x_1(\omega) \in \Re^n} \left(c^T x^* + d^T y(\omega) \right)$, which is the R.H.S. of the expression (A.1).

   Next, we let $x_0' = \begin{pmatrix} x' \\ 0 \end{pmatrix}$ and $x_1'(\omega) = \begin{pmatrix} x' \\ y'(\omega) \end{pmatrix}$ where $y'(\omega)$ takes the same

values as $y(\omega)$ in the solution in CARO with solution x' at optimal (i.e. the $y(\omega)$ values that give $\min_{\forall y(\omega)\in\Re^n}\left(c^Tx' + d^Ty(\omega)\right)$). Then the L.H.S. of the expression (A.1) can be re-expressed as $\max_{\forall\omega\in\Omega}\left(c^Tx' + d^Ty'(\omega)\right)$. Since x' is a feasible solution in CARO, the constraints $A(\omega)x' + B(\omega)y'(\omega) \geq b(\omega)$ are satisfied, and can be expressed as $\begin{pmatrix} A(\omega) & B(\omega) \end{pmatrix} x'_1(\omega) \geq b(\omega)$, and we get a feasible solution in Nimo.

The objective value of Nimo with solution x'_0 is

$\max_{\forall\omega\in\Omega}\min_{\forall x_1(\omega)\in\Re^n}\left(\begin{pmatrix} c^T & d^T \end{pmatrix} x_1(\omega) + \Delta\left(x'_0, x_1(\omega)\right)\right)$, which will be less than or equal to $\max_{\forall\omega\in\Omega}\left(\begin{pmatrix} c^T & d^T \end{pmatrix} x'_1(\omega) + \Delta(x'_0, x'_1(\omega))\right)$, and can be re-expressed as $\max_{\forall\omega\in\Omega}\left(c^Tx' + d^Ty'(\omega)\right)$, which is the L.H.S. of the expression. We have now a feasible solution x'_0 in Nimo, whose objective value is equal to the L.H.S. of expression (A.1) and it is assumed to be smaller than the R.H.S., which is the objective value of the optimum solution x^*_0. We have a contradiction where we have found a feasible solution of Nimo having a smaller objective value than the optimum, hence x^* is the optimum solution of CARO.

#

FARO

Next we consider the Finite Adaptable RO. FARO model restricts the set $y(\omega)$ to a finite set with a fixed given cardinality k. Hence $y(\omega) \in \{y_1, y_2, \ldots, y_k\}$, i.e. there are k contingencies for all possible $\omega \in \Omega$, and hence for each possible ω, we just need to ensure that at least one of the elements in $\{y_1, y_2, \ldots, y_k\}$ satisfies the constraints.

Consider the following FARO model:

$$\text{Minimise}_{\forall x \in \Re^n} \quad \left(\max_{\forall y \in \{y_1, y_2, \ldots, y_k\}} (cx + dy) \right)$$

$$\begin{aligned} s.t. \quad & A(\omega)x + B(\omega)y_1 \geq b(\omega) \text{ or} \\ & A(\omega)x + B(\omega)y_2 \geq b(\omega) \text{ or} \\ & \vdots \\ & A(\omega)x + B(\omega)y_k \geq b(\omega) \ \ \forall \omega \in \Omega \end{aligned}$$

We transform FARO to Nimo with the following transformation.

We let

$$x_0 = \begin{pmatrix} x & y_1 & \ldots & y_i & \ldots & y_k \end{pmatrix}^T$$

We impose that x_1 takes the following form of

$$x_1 = \begin{pmatrix} x & 0 & \ldots & 0 & y_i & 0 & \ldots & 0 \end{pmatrix}^T \quad 1 \leq i \leq k$$

by defining the Δ function as follows:

$$\Delta \left(\begin{pmatrix} x \\ y_1 \\ y_2 \\ \vdots \\ y_k \end{pmatrix}, \begin{pmatrix} x' \\ y'_1 \\ y'_2 \\ \vdots \\ y'_k \end{pmatrix} \right) = \begin{cases} 0 & \text{if } x = x' \text{ and } \exists i \text{ s.t. } 1 \leq i \leq k \\ & \quad y_i = y'_i, \text{ and } \forall j, 1 \leq j \leq k \text{ and } j \neq i, y'_j = 0, \\ \infty & \text{otherwise} \end{cases}$$

Then $x, y_1, \ldots, y_k$ in the solution $x_0 = \begin{pmatrix} x & y_1 & y_2 & \ldots & y_k \end{pmatrix}^T$ of the following Nimo is the solution of the original problem FARO.

$$\underset{\forall x_0 \in \Re^n}{\text{Minimise}} \quad \max_{\forall \omega \in \Omega} \quad \min_{\forall x_1(\omega) \in \Re^n} \left[\begin{array}{l} \begin{pmatrix} c^T & d^T & \ldots & d^T \end{pmatrix} x_1(\omega) + \Delta(x_0, x_1(\omega)) \\ \text{subject to} \\ \begin{pmatrix} A(\omega) & B(\omega) & \ldots & B(\omega) \end{pmatrix} x_1(\omega) \geq b(\omega) \end{array} \right]$$

$$s.t. \quad \begin{pmatrix} A(\omega) & B(\omega) & \ldots & B(\omega) \end{pmatrix} x_1(\omega) \geq b(\omega) \quad \forall \omega \in \Omega$$

Proof:

Let $x_0^* = \begin{pmatrix} x^* & y_1^* & \ldots & y_k^* \end{pmatrix}^T$ be the optimum solution for Nimo. The transformation says that the optimum solution in the FARO problem is $x^*, y_1^*, \ldots, y_k^*$.

1. If the objective value of x_0^* is ∞, it means that there does not exist $x_1(\omega)$ such that the constraints $\begin{pmatrix} A(\omega) & B(\omega) & \ldots & B(\omega) \end{pmatrix} x_1(\omega) \geq b(\omega)$ are satisfied and

$$x_1(\omega) \in x_1(\Omega) = \left\{ \begin{pmatrix} x' \\ y_1^* \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} x' \\ 0 \\ y_2^* \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} x' \\ 0 \\ \vdots \\ 0 \\ y_i^* \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} x' \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ y_k^* \end{pmatrix} \right\}$$

Hence there does not exist x', $y'_1, \ldots, y'_k$ such that the conditions for FARO are satisfied, otherwise Nimo will have adopted them to avoid ∞ in objective value. This will mean that there does not exist a feasible solutions in FARO.

2. If the objective value is not ∞.

   (a) We first show that x^*, $y^*_1, \ldots, y^*_k$ is a feasible solution in FARO. Since the objective value is not ∞, $x_1(\omega)$ must take the form

$$x_1(\omega) \in x_1(\Omega) = \left\{ \begin{pmatrix} x' \\ y^*_1 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} x' \\ 0 \\ y^*_2 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} x' \\ 0 \\ \vdots \\ 0 \\ y^*_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} x' \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ y^*_k \end{pmatrix} \right\}$$

   Then, because the constraints in Nimo are satisfied, they can be re-expressed as $A(\omega)x^* + B(\omega)y^*_i \geq b(\omega)$ and hence x^*, $y^*_1, \ldots, y^*_k$ is a feasible solution in FARO.

   (b) Next we show that x^*, $y^*_1, \ldots, y^*_k$ is the optimum solution in FARO. We prove by contradiction by assuming that there exists another solution x', $y'_1, \ldots, y'_k$ where the objective value of FARO with x', $y'_1, \ldots, y'_k$ is smaller than that of

$x^*, y_1^*, \ldots, y_k^*$, i.e.,

$$\max_{\forall y \in \{y_1', y_2', \ldots, y_k'\}} \left[c^T x' + d^T y\right] < \max_{\forall y \in \{y_1^*, y_2^*, \ldots, y_k^*\}} \left[c^T x^* + d^T y\right] \qquad \text{(A.2)}$$

The objective value of Nimo with solution x_0^* is $\max_{\forall \omega \in \Omega} \min_{\forall x_1(\omega) \in \Re^n} \left(\begin{pmatrix} c^T & d^T & \ldots & d^T \end{pmatrix} x_1(\omega) + \Delta\left(x_0^*, x_1(\omega)\right) \right)$. With the value of Δ as zero, and $x_1(\omega)$ taking the form $\begin{pmatrix} x^* & 0 & \ldots & y_i^* & \ldots & 0 \end{pmatrix}^T$, the objective value simplifies to $\max_{\forall x_1(\omega) \in x_1(\Omega)} \left(\begin{pmatrix} c^T & d^T & \ldots & d^T \end{pmatrix} x_1(\omega) \right)$, and further simplifies to $\max_{\forall y \in \{y_1', y_2', \ldots, y_k'\}} \left(c^T x^* + d^T y\right)$, which is the R.H.S. of the expression (A.2).

Next, we let $x_0' = \begin{pmatrix} x' & y_1' & \ldots & y_k' \end{pmatrix}^T$ and $x_1'(\omega)$ take the form $\begin{pmatrix} x' & 0 & \ldots & y_i' & \ldots & 0 \end{pmatrix}^T$. Since $x', y_1', \ldots, y_k'$ is a feasible solution in FARO, the constraints $A(\omega)x' + B(\omega)y' \geq b(\omega)$ are satisfied for some ω, and can be expressed as $\begin{pmatrix} A(\omega) & B(\omega) & \ldots & B(\omega) \end{pmatrix} x_1'(\omega) \geq b(\omega)$, and we get the feasible solution in Nimo. The objective value of Nimo with solution x_0' is $\max_{\forall \omega \in \Omega} \min_{\forall x_1(\omega) \in \Re^n} \left(\begin{pmatrix} c^T & d^T & \ldots & d^T \end{pmatrix} x_1(\omega) + \Delta(x_0', x_1(\omega)) \right)$, which will be less than or equal to $\max_{\forall \omega \in \Omega} \left(\begin{pmatrix} c^T & d^T & \ldots & d^T \end{pmatrix} x_1'(\omega) + \Delta\left(x_0', x_1'(\omega)\right) \right)$, and can be re-expressed as $\max_{\forall y \in \{y_1', y_2', \ldots, y_k'\}} \left(c^T x^* + d^T y\right)$, which is the L.H.S. of the expression (A.2). We have now a feasible solution x_0' in Nimo, whose objective value is equal to the L.H.S. of (A.2) and smaller than the R.H.S. of (A.2), which is the objective value of the optimum solution x_0^*. We have a contradiction where we have found a feasible solution of Nimo having a smaller objective value than the optimum, hence $x^*, y_1^*, \ldots, y_k^*$ is the optimum solu-

tion of FARO.

#

A.5 Soyster's Model (SOY)

For completeness, we also consider Soyster's (Soyster 1973) model (SOY), the iconic work in the arena of robust optimisation, where all decisions are made before the uncertainty is revealed, and the decision remains feasible for all possible outcomes of ω. There is no stage 2 decision, or rather the stage 2 does not change the decision of stage 1. We show that Nimo is also a generalisation of SOY as follows.

SOY can be expressed as

$$\underset{\forall x \in \Re^n}{\text{Minimise}} \;\; c^T x$$

$$s.t. \;\; A(\omega)x \geq b(\omega) \;\; \forall \omega \in \Omega$$

The transformation is as follows:

Let $x_0 = x$. We impose an implicit constraint that $x_1 = x_0$, by having $\Delta(x_0, x_1)$ defined as

$$\Delta(x_0, x_1) = \begin{cases} 0 & \text{if } \; x_0 = x_1 \\ \\ \infty & \text{otherwise} \end{cases}$$

Then the solution x_0 of the following Nimo is the solution x of the original problem SOY.

$$\underset{\forall x_0 \in \Re^n}{\text{Minimise}} \quad \max_{\forall \omega \in \Omega} \min_{\forall x_1(\omega) \in \Re^n} \left[\begin{array}{l} c^T x_1(\omega) + \Delta(x_0, x_1(\omega)) \\ \text{subject to} \\ A(\omega) x_1(\omega) \geq b(\omega) \end{array} \right]$$

Proof:

Let x_0^* be the optimum solution for Nimo. The transformation says that the optimum solution in problem SOY is $x^* = x_0^*$.

1. If the objective value of x_0^* is ∞, it means that there does not exist a x_0' such that $x_1(\omega) = x_0' \; \forall \omega \in \Omega$, and the conditions $A(\omega)x_1(\omega) \geq b(\omega)$ are satisfied. This will mean that there does not exist a feasible solutions in SOY.

2. If the objective value is not ∞

   (a) We first show that x^* is a feasible solution in the original model. Since the objective value is not ∞, $x_1(\omega)$ must be equal to x_0^*, and because the constraints in Nimo are satisfied, it is obvious that x^* is a feasible solution in SOY.

   (b) Next we show that x^* is the optimum solution in SOY. We prove by contradiction by assuming that there exists another solution x' where the objective value of SOY with x' is smaller than that of x^*, i.e.,

   $$c^T x' < c^T x^*$$

   The objective value of Nimo with solution x_0^* is

$\max_{\forall\omega\in\Omega} \min_{\forall x_1(\omega)\in\Re^n} \left(c^T x_1(\omega) + \Delta\left(x_0^*, x_1(\omega)\right)\right)$. With the value of Δ as zero, and $x_1(\omega) = x^*$, the objective value simplifies to $c^T x^*$. Next, we let $x'_0 = x'$ and $x'_1(\omega) = x'_0\ \forall\omega \in \Omega$. Since x' is a feasible solution in SOY, the constraints $A(\omega)x'_1(\omega) \geq b(\omega)$ in Nimo are also satisfied, and we get a feasible solution in Nimo whose objective value is $\max_{\forall\omega\in\Omega} \min_{\forall x_1(\omega)\in\Re^n} \left(c^T x_1(\omega) + \Delta\left(x'_0, x_1(\omega)\right)\right)$. This can be simplified to $c^T x'$, and by our assumption, this is less than $c^T x^*$, the objective value of the optimum solution x_0^*. We have a contradiction where we have found a feasible solution of Nimo having a smaller objective value than the optimum, hence x^* is the optimum solution of SOY.

#

A.6 Regret optimisation (RegO)

While the classical regret optimization (RegO) problem is not exactly a robust optimisation problem, however, it does consider effects of uncertainty on the solution. The concept is to minimise the worst case regret defined by a regret function – the difference in objective value of the chosen solution from the optimum solution if the uncertain outcome is known in advance. We show that Nimo is also a generalisation of the RegO problem.

Mathematically, we consider the RegO as follows:

$$\underset{x\in\Re^n}{\text{Minimise}} \quad \max_{\omega\in\Omega} \quad \min_{y(\omega)\in\Re^n} \begin{bmatrix} c^T x - c^T y(\omega) \\ \text{subject to} \\ A(\omega)y(\omega) \geq b(\omega) \\ A(\omega_0)x \geq b(\omega_0) \end{bmatrix}$$

We transform RegO to Nimo with the following transformation. We let

$$x_0 = \begin{pmatrix} 0 \\ \\ x \end{pmatrix}, \text{ and } \; x_1 = \begin{pmatrix} x-y \\ \\ x \end{pmatrix}$$

We also define

$$\Delta\left(\begin{pmatrix} z \\ \\ x \end{pmatrix}, \begin{pmatrix} z' \\ \\ x' \end{pmatrix}\right) = \begin{cases} 0 & \text{if } \; x = x' \text{ and } z = 0 \\ \\ \infty & \text{otherwise} \end{cases}$$

Then the vector x in the solution $x_0 = \begin{pmatrix} 0 \\ \\ x \end{pmatrix}$ of the following Nimo is the solution x of the original RegO problem.

$$\underset{\forall x_0\in\Re^n}{\text{Minimise}} \quad \max_{\forall\omega\in\Omega} \quad \min_{\forall x_1(\omega)\in\Re^n} \begin{bmatrix} \begin{pmatrix} c^T & 0 \end{pmatrix} x_1(\omega) + \Delta(x_0, x_1(\omega)) \\ \text{subject to} \\ \begin{pmatrix} -A(\omega) & A(\omega) \end{pmatrix} x_1(\omega) \geq b(\omega) \\ \begin{pmatrix} -A(\omega_0) & A(\omega_0) \end{pmatrix} x_0 \geq b(\omega_0) \end{bmatrix}$$

Proof:

Let $x_0^* = \begin{pmatrix} 0 \\ \\ x^* \end{pmatrix}$ be the optimum solution for Nimo. The transformation says that the optimum solution in problem RegO is x^*.

1. If the objective value of x_0^* is ∞, it means that there does not exist x' such that $x_1(\omega) = \begin{pmatrix} x' - y(\omega) \\ \\ x' \end{pmatrix} \forall \omega \in \Omega$, and the constraints $\begin{pmatrix} -A(\omega) & A(\omega) \end{pmatrix} x_1(\omega) \geq b(\omega)$ are satisfied. That is, there does not exist a x' and $y(\omega)$ such that $-A(\omega)x' + A(\omega)y(\omega) + A(\omega)x' = A(\omega)y(\omega) \geq b(\omega)$. This will mean that there does not exist a feasible solution in RegO.

2. If the objective value is not ∞.

   (a) We first show that x^* is a feasible solution in RegO. Since the objective value is not ∞, $x_1(\omega)$ must take the form $\begin{pmatrix} x^* - y(\omega) \\ \\ x^* \end{pmatrix}$, and x_0 must take the form $\begin{pmatrix} 0 \\ \\ x^* \end{pmatrix}$. Then, because the constraints in Nimo are satisfied, they can be re-expressed as $-A(\omega)x^* + A(\omega)y(\omega) + A(\omega)x^* = A(\omega)y(\omega) \geq b(\omega)$ and $A(\omega_0)x* \geq b(\omega_0)$. Hence x^* is a feasible solution in RegO.

   (b) Next we show that x^* is the optimum solution in RegO. We prove by contradiction by assuming that there exists another solution x' where the objective value of RegO with x' is smaller than that of x^*, i.e.,

$$\max_{\forall \omega \in \Omega} \left(\min_{\forall y(\omega) \in \Re^n} [c^T x' - c^T y(\omega)] \right) < \max_{\forall \omega \in \Omega} \left(\min_{\forall y(\omega) \in \Re^n} [c^T x^* - c^T y(\omega)] \right) \quad \text{(A.3)}$$

The objective value of Nimo with solution x_0^* is

$\max_{\forall\omega\in\Omega} \min_{\forall x_1(\omega)\in\Re^n} \left(\begin{pmatrix} c^T & 0 \end{pmatrix} x_1(\omega) + \Delta\left(x_0^*, x_1(\omega)\right) \right)$. With the value of Δ as zero, and $x_1(\omega) = \begin{pmatrix} x^* - y(\omega) \\ x^* \end{pmatrix}$, the objective value simplifies to

$\max_{\forall\omega\in\Omega} \min_{\forall x_1(\omega)\in\Re^n} \left(\begin{pmatrix} c^T & 0 \end{pmatrix} \begin{pmatrix} x^* - y(\omega) \\ x^* \end{pmatrix} \right)$, and further simplifies to

$\max_{\forall\omega\in\Omega} \min_{\forall x_1(\omega)\in\Re^n} \left(c^T x^* - c^T y(\omega)\right)$, which is the R.H.S. of the expression (A.3).

Next, we let $x_0' = \begin{pmatrix} 0 \\ x' \end{pmatrix}$ and $x_1'(\omega) = \begin{pmatrix} x' - y'(\omega) \\ x' \end{pmatrix}$ where $y'(\omega)$ takes the same values as $y(\omega)$ in the solution in RegO with solution x' at optimal (i.e. the $y(\omega)$ values that gives $\min_{\forall y(\omega)\in\Re^n} \left(c^T x' - c^T y(\omega)\right)$). Then the L.H.S. of the expression (A.3) can be re-expressed as $\max_{\forall\omega\in\Omega} \left(c^T x' - c^T y'(\omega)\right)$. Since x' is a feasible solution in RegO, the constraints $A(\omega)y'(\omega) \geq b(\omega)$ and $A(\omega_0)x' \geq b(\omega_0)$ are satisfied, and they can be expressed as $\begin{pmatrix} -A(\omega) & A(\omega) \end{pmatrix} x_1'(\omega) \geq b(\omega)$ and $\begin{pmatrix} -A(\omega_0) & A(\omega_0) \end{pmatrix} x_0' \geq b(\omega_0)$. We get a feasible solution x_0' in Nimo.

The objective value of Nimo with solution x_0' is

$\max_{\forall\omega\in\Omega} \min_{\forall x_1(\omega)\in\Re^n} \left(\begin{pmatrix} c^T & 0 \end{pmatrix} x_1(\omega) + \Delta\left(x_0', x_1(\omega)\right) \right)$, which will be less than or equal to $\max_{\forall\omega\in\Omega} \left(\begin{pmatrix} c^T & 0 \end{pmatrix} x_1'(\omega) + \Delta(x_0', x_1'(\omega)) \right)$, and can be re-expressed as

$\max_{\forall\omega\in\Omega} \left(c^T x' - c^T y'(\omega)\right)$, which is the L.H.S. of the expression. We have now a feasible solution x_0' in Nimo, whose objective value is equal to the L.H.S. of expression (A.3) and it is assumed to be smaller than the R.H.S., which

is the objective value of the optimum solution x_0^*. We have a contradiction where we have found a feasible solution of Nimo having a smaller objective value than the optimum, hence x^* is the optimum solution of RegO.

#

A.7 Summary

In this appendix, we have shown that nimble optimisation (Nimo) is a generalisation of some known problem models, namely, Ben-Tal's ARC, LiebChen's LRR, Bertsimas's CARA and FARO, Soyster's robust model, and the classical regret optimisation model. This list is not likely to be exhaustive, and there could be other known problems that we have not included in this thesis.