Berth Allocation Planning Optimization in Container Terminals

Jim Dai * Wuqin Lin † Rajeeva Moorthy ‡ Chung-Piaw Teo§

Abstract

We study the problem of allocating berth space for vessels in container terminals, which is referred to as the *berth allocation planning problem.* We solve the static berth allocation planning problem as a rectangle packing problem with arrival time constraints, using a local search algorithm that employs the concept of *sequence pair* to define the neighborhood structure. We embed this approach in a real time scheduling system to address the berth allocation planning problem in a dynamic environment. We address the issues of vessel allocation to the terminal (thus affecting the overall berth utilization), choice of planning time window (how long to plan ahead in the dynamic environment), and the choice of objective used in the berthing algorithm (e.g., should we focus on minimizing vessels' waiting time or maximizing berth utilization?). In a moderate load setting, extensive simulation results show that the proposed berthing system is able to allocate space to most of the calling vessels upon arrival, with the majority of them allocated the preferred berthing location. In a heavy load setting, we need to balance the concerns of throughput with acceptable waiting time experienced by vessels. We show that, surprisingly, these can be handled by deliberately delaying berthing of vessels in order to achieve higher throughput in the berthing system.

1 Introduction

Competition among container ports continues to increase as the differentiation of hub ports and feeder ports progresses. Managers in many container terminals are trying to attract carriers by automating handling equipment, providing and speeding up various services, and furnishing the most current information on the flow of containers. At the same time, however, they are trying to reduce costs by utilizing resources efficiently, including human resources, berths, container yards, quay cranes, and various yard equipment.

Containers come into the terminals via ships. The majority of the containers are 20 feet and 40 feet in length. The quay cranes pick up the containers and load them into prime movers

*School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. Email: dai@isye.gatech.edu

†School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA. Email: linwq@isye.gatech.edu

‡Department of Decision Sciences, National University of Singapore, Singapore 119260. Email: dscrlmk@nus.edu.sg

§Department of Decision Sciences, National University of Singapore, Singapore 119260. Email: bizteocp@nus.edu.sg

(container trucks). The trucks then move them to the terminal yards. The yard cranes at the terminal yards then pick up the containers from the prime movers and stack them neatly in the yards according to a stacking pattern and schedule. Prime movers enter the terminals to pick up the containers for distribution to distriparks and customers. The procedure is reversed for cargo leaving the port. See Figure 1 for an example of a prime mover unloading containers from a vessel.

Figure 1: A prime mover unloading containers from a vessel

The problem studied in this paper is motivated by a berth allocation planning problem faced by a port operator. When planning berth usage, the berthing time and the exact position (i.e., wharf mark) of each vessel at the wharf, as well as various quay-side resources are usually determined. Several variables must be considered, including the length-overall and (expected) arrival time of each vessel, the number of containers for discharging and loading on each vessel, and the storage location of outbound/inbound containers to be loaded onto/discharged from the corresponding vessel. To minimize disruption and maximize efficiency, most of the customers (i.e., vessel owners) expect prompt berthing of their vessels upon arrival. This is particularly important for vessels from priority customers (called priority vessels hereon), who may have been guaranteed berth-on-arrival (i.e., within *two hours of arrival*) service in their contract with the terminal operator. On the other hand, the port operator is also measured by her ability to utilize available resources (berth space, cranes, prime-movers, etc.) in the most efficient manner,

with berth utilization,[1] berthing delays faced by customers and terminal planning[2] being prime concerns.

We assume that the terminal is divided into several wharfs, which in turn are divided into berths. Each wharf corresponds to a linear stretch of space in the terminal. Figure 2 shows the layout of 3 terminals in Singapore. For instance, Keppel Terminal is divided into 5 wharfs, with 5, 4, 3, 1 and 2 different berths in the 5 wharfs. Note that vessels can physically be berthed across different berths, but they cannot be berthed across different wharfs.

Figure 2: Terminal layout in 3 terminals in Singapore

For each vessel calling at the terminal, the vessel turn-around time at the port can normally be calculated by examining historical statistics (number of containers handled for the vessel, the crane intensity allocated to the vessel, historical crane rate, etc.). Vessel owners usually request a berthing time (called Berth-Time-Requested or BTR) in the terminal far in advance, and are usually allowed to revise the BTR when the vessel is close to calling at the terminal. Given a set of vessels calling at the terminal in a periodic schedule, our goal is to design a berthing system to allocate berthing space to the vessels, to ensure that most vessels, if not all, can be berthed on-arrival, and that most of the vessels will be allocated berthing space close to their preferred locations within the terminal. Note that the constraints in berthing space allocation give rise to non-convex domain, rendering the search for an optimal solution difficult. Furthermore, we

[1]This is the ratio of berth availability (hours of operations $\times$ total berth length) to berth occupancy (vessel time at berth $\times$ length occupied). The utilization rate is affected by the number and types of vessels allocated to the terminal. However, service performance (i.e., delays) will normally deteriorate with higher berth utilization, since more vessels will be competing for the use of the terminal.

[2]Containers to be unloaded from or loaded onto the vessels are normally stored at particular storage locations within the yard. To minimize vessel turn-around time, the berthing location should ideally be selected close to the storage location of the containers.

have to take into account the availability and accuracy of berthing time information, and must be able to determine an allocation dynamically over time.

In the berthing system, we define a *scheduling window* to be a time interval such that, at the beginning of the interval, the arrival time and processing information on all relevant ships are known. Here the relevant ships refer to all those ships that are currently at the terminal or will arrive at the terminal during the time interval. Of course, as time elapses, the scheduling window rolls forward. An optimal packing within a scheduling window is a schedule of all relevant ships that optimizes a certain objective. Examples of objectives include maximizing berth utilization within the window, maximizing throughput within the window, and minimizing berthing delay and space cost within the window.

Which objective should we use to obtain the packing within each scheduling window? How often should the packing schedules be produced? Do we update and change the vessel berthing plan in every scheduling window? How long should the scheduling window be? These decisions affect the size of the problem we need to solve in each scheduling window. The choice of the objective function, coupled with the non-convex packing constraints, often lead to extremely difficult combinatorial optimization problems. This leads us to the first issue related to the berth allocation planning problem:

> **Static Berth Allocation Problem**:
> How do we design an efficient berth planning system to allocate berthing space to (say) 100 vessels in each berthing window?

Other than the above combinatorial complexity associated with berth planning, we also need to embed the plan in a dynamic berth planning system to allocate berthing space to vessels over time. To do this, we need to address how the scheduling window is to be selected, and how often to update the berthing plan for those vessels considered in earlier scheduling windows.

This leads us to the second issue related to the berth allocation planning problem:

> **Dynamic Berth Allocation Problem**:
> How do we design a real time berth planning system to allocate berthing space to vessels which call at the terminal at regular intervals?

The berthing system is designed based on a rolling horizon framework. In a moderate load setting, we expect the berthing system to be able to accommodate most of the berthing constraints and allocate space to vessels, minimizing waiting time experienced by vessels and optimizing the utilization of resources available in the terminal. In a heavy load setting, however, the trade-off is especially crucial, since the impact of inefficient resource utilization translates into large drop in throughput. To understand the difficulty in handling these problems, we use a simple example to illustrate the challenges faced in the dynamic berth allocation planning problem.

Example:
Consider an example with three classes of vessels and a wharf with seven sections. Each class 1

vessel occupies 3 sections; each class 2 vessel occupies 4 sections; and each class 3 vessel occupies 5 sections. Suppose the arrival rates of both class 1 and class 2 vessels are 1 vessel per 16 hours, and the arrival rate of type 3 vessels is 1 per 640 hours. The n-th class 1 vessel arrives at time $16n - 5$, class 2 at time $16(n - 1)$, and class 3 at time $640n$. The processing times of the three classes of vessels are deterministic. They are 12 hours, 14 hours and 16 hours for class 1, class 2, and class 3 vessels, respectively. Notice that, at most, two vessels can be processed each time. If we ignore the class 3 vessels, it is obvious that we can pack all the class 1 and class 2 vessels immediately upon arrival. If we wish to minimize delays, the optimal packing is given by Figure 3.

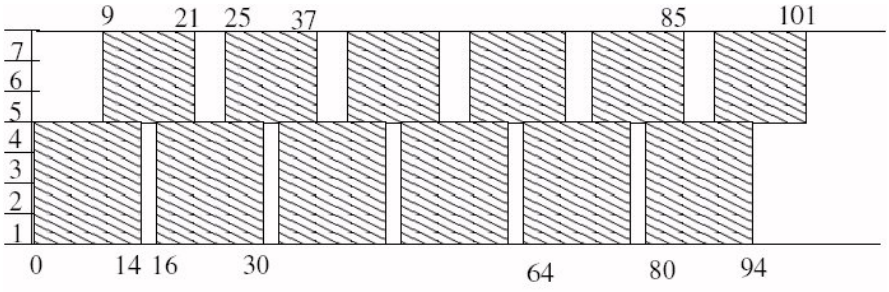

Figure 3: Optimal Packing to minimize delays for class 1 and 2 vessels, when class 3 vessels are ignored

However, with class 3 vessels being assigned to the terminal, the berthing system must now address the trade-off between delays for vessels in the 3 different classes. How would a good dynamic berth planning system handle the load situation presented by this example?

Case (a): Consider the situation where the most important objective in the scheduling window is to maximize *berth utilization* within the scheduling window, and say the scheduling window is chosen to be $W = 32$ hours.

Figure 4 shows the number of vessels waiting for service over time. It reveals a major flaw in the design of the above planning system: *by focusing on berth utilization within the scheduling window, class 3 vessels will never have a chance to be served at all!*

To understand why this is the case, note that the blank rectangles in Figure 3 are the capacity (measured by number of sections $\times$ time in hours) that can not be utilized. The space utilization is determined by the total area of blank rectangles: the larger the blank area, the smaller the utilization. Denote $U^1(\tau_1, \tau_2)$ to be the unutilized capacity (blank area) between time τ_1 and τ_2 under the packing policy shown in Figure 3. For any $\tau \geq 5$, we have:

$$U^1(\tau, \tau + 16) = 20. \tag{1}$$

This is because under this packing policy during any period of 16 hours, sections 5-7 are idle for 4 hours and sections 1-4 are idle for 2 hours. In fact, from Figure 3 one can observe that the

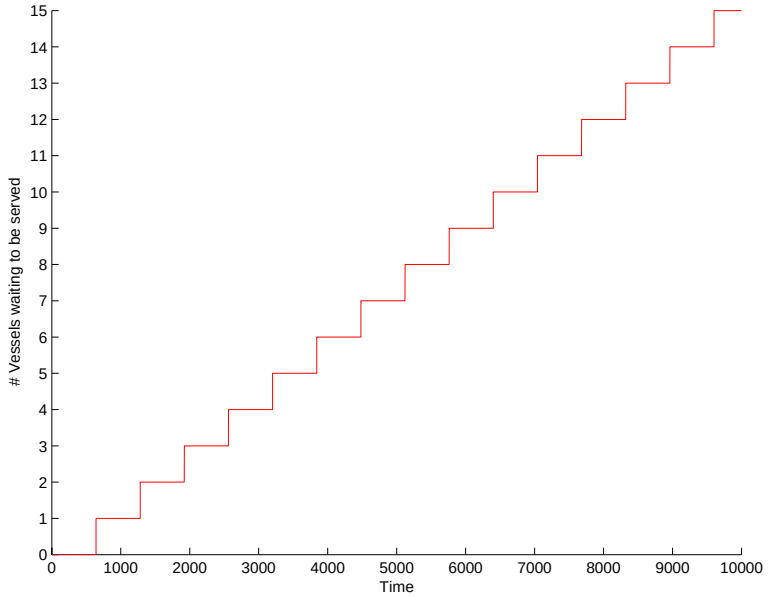

Figure 4: Identical buffer build up for class 3 vessels, under the objective of **(a):** maximizing berth utilization and **(b):** minimizing total waiting time

packing is periodic with period 16 and within each period there are two blank rectangles with respective areas 12 and 8.

Unfortunately, this is the best packing obtained if the objective is to maximize the berth utilization within each scheduling window. Hence the berthing system will not allocate berthing space to any class 3 vessels within any scheduling window. For a formal proof of this statement, we refer the readers to Appendix I.

Case (b): The flaw in the above objective lies in the fact that vessel waiting time has not been considered in the berth allocation objective. To rectify this problem, one obvious approach is to change the objective to minimizing total waiting time within each scheduling window. More formally, suppose the objective function is changed to

$$\min \sum_{i=1}^{N} (t_i - r_i)^+,$$

where r_i, t_i are the actual arrival time and planned berthing time of vessel i respectively, and N is the number of vessels considered in a scheduling window. Recall, for a real number x, $x^+ = \max(x, 0)$. Unfortunately, the same problem remains with this model: again, class 3 vessels will have to wait indefinitely for service in this terminal! The number of vessels waiting for service at the terminal as a function of time for this case is exactly the same as in the case of maximizing berth utilization and is shown in Figure 4.

This phenomenon can be explained by the fact that if class 3 vessels are berthed in any scheduling window, it will induce delays on one class 1 vessel and one class 2 vessel. Hence in each scheduling window, the class 3 vessels will be "sacrificed" and will have to wait for service indefinitely.

The above problem associated with dynamic berth allocation planning can be removed if we assign a higher priority status to vessels with larger length-overall (class 3 vessels) and assign a higher penalty for delaying vessels with higher priority, i.e., changing the objective to one of minimizing weighted waiting time. Figure 5 shows the effect of using differentiated priorities to prevent buffer build-up. Note that class 3 vessels will be berthed upon arrival, due to the higher priority status assigned. The queues built up can be cleared before the arrival of the next class 3 vessels.

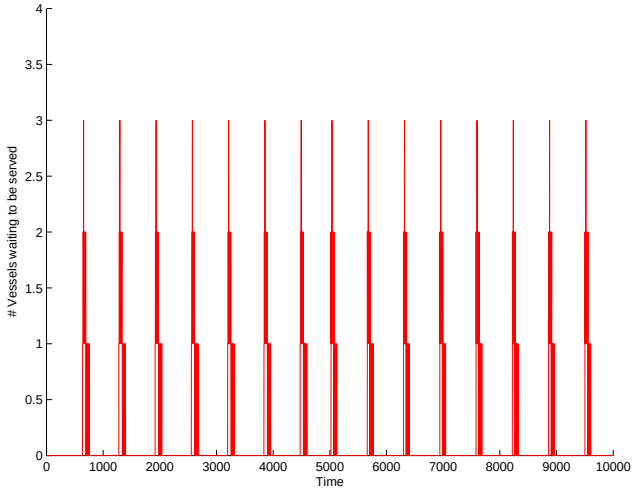

Figure 5: Buffer length over time when employing differentiated priority

This approach of assigning higher priority to vessels with larger length-overall, however, may lead to difficulties in other problem instances. For example, if the class 3 vessels occupy only 1 section, with a port stay of 6 hours, then this weighted priority approach will assign low priority to class 3 vessels. Class 1 and 2 vessels will be berthed immediately upon arrival, leaving only gaps of 2 hours and 4 hours each between successive berthing (cf. Figure 3). In this case, again, class 3 vessels will not be berthed by the berth allocation planning system, leading to a reduction in terminal throughput. We need a different approach to guarantee near optimal throughput by the terminal. In practice, the problem of assigning priority status to vessels is also complicated by the fact that the priority status for some vessels is determined by the contractual agreement signed between ports and customers, and cannot be arbitrarily assigned.

In the rest of this paper, we outline an approach to address the above issues associated with

the berth allocation planning problem. Our contributions can be summarized as follows:

- We solve the static berth allocation planning problem as a rectangle packing problem with arrival time constraints. The objective function chosen involves a combination of weighted waiting time and allocated berthing space cost function. The optimization approach builds on the "sequence-pair" concept that has been used extensively in solving VLSI layout design problems. While this method has been proven useful for addressing classical rectangle packing problem to obtain tight packing, our rectangle packing problem is different as the key concern here is to pack the vessel efficiently using the available berth space. Using the concepts of "virtual wharf marks", we show how the sequence-pair approach can be augmented to address the rectangle packing problem arising from the berth allocation planning model.

- We extend the static berth allocation model to address the case where certain regions in the time-space network cannot be used to pack arriving vessels. These forbidden regions correspond to instances where certain vessels have already been berthed and will leave the terminal some time later. In this case, the space allocated to these vessels cannot be used to berth other arriving vessels. The ability to address these side constraints allows our model to be embedded in a dynamic berth allocation planning model.

- In a heavy load setting, to ensure that throughput of the terminal will not be adversely affected due to the design of the berthing system, we propose a *discrete maximum pressure* policy to redistribute the load at the terminal at *periodic* intervals. We prove that this policy is throughput optimal even when processing times and inter-arrival times of vessels are random. Interestingly, this policy ensures that vessels do not have to wait indefinitely for service, although the policy may deliberately insert delays on certain vessels, even when the terminal has enough resources to berth the affected vessels on time. Within each scheduling window, however, the static allocation planning problem will be solved to assign berthing space to the vessels. Hence this approach ensures that both the service performance of the berthing system (BOA, preferred berthing space allocated, etc.) and terminal throughput performance are addressed by the berthing system.

- We conclude our study with an extensive simulation of the proposed approach, using a set of arrival patterns extracted and suitably modified from real data. As a side product, our simulation also illustrates the importance of the choice of scheduling windows, and the trade-off between frequent revision to berthing plans and berthing performance. Note that frequent revision of plans is undesirable from a port operation perspective, as frequent revision has an unintended impact on personnel and resource schedules.

2 Literature Review

To the best of our knowledge, none of the papers in the literature address the dynamic berth planning allocation problem proposed in this paper. Most of the existing papers focus mainly on

the static berth allocation problem, where the central issue is to obtain a good plan to pack the vessels waiting and arriving within the scheduling window. Brown et al. (1994) formulated an integer-programming model for assigning one possible berthing location to a vessel considering various practical constraints. However, they considered a berth as a collection of discrete berthing locations, and their model is more apt for berthing vessels in a naval port, where berth shifting of moored vessels is allowed. Lim (1998) addressed the berth planning problem by keeping the berthing time fixed while trying to decide the berthing locations. The berth was considered to be a continuous space rather than a collection of discrete locations. He proposed a heuristic method for determining berthing locations of vessels by utilizing a graphical representation for the problem. Chen and Hsieh (1999) proposed an alternative network-flow heuristic by considering the time-space network model. Tong, Lau, and Lim (1999) solved the ship berthing problem using the Ants Colony Optimization approach, but they focused on minimizing the wharf length required while assuming the berthing time as given. In the Berth Allocation Planning System (BAPS) (and its later incarnation iBAPS) developed by the Resource Allocation and Scheduling (RAS) group of NUS, a two-stage strategy is adopted to solve the BAP problem. In the first stage, vessels are partitioned into sections of the port without specifying the exact berth location. The second stage determines specific berthing locations for vessels and packs the vessels within their assigned sections. See Loh (1996) and Chen (1998). Moon (2000), in an unpublished thesis, formulated an integer linear program for the problem and solved using LINDO package. The computational time of LINDO increased rapidly when the number of vessels became higher than 7 and the length of the planning horizon exceeded 72 hours. Some properties of the optimal solution were investigated. Based on these properties, a heuristic algorithm for the berth planning problem was suggested. The performance of the heuristic algorithm was compared with that of the optimization technique. The heuristic works well on many randomly generated instances, but performs badly in several cases when the penalty for delay is substantial.

The static berth planning problem is related to a variant of the two-dimensional rectangle packing problem, where the objective is to place a set of rectangles in the plane without overlap so that a given cost function will be minimized. This problem arises frequently in VLSI design and resource-constrained project scheduling. Furthermore, in certain packing problems, the rectangles are allowed to rotate 90^o. In these problems, the objectives are normally to minimize the height or area used in the packing solution. Imahori et al. (2002), building on a long series of work by Murata et al. (1996), Tang et al. (2000), etc., propose a local search method for this problem, using an encoding scheme called sequence pair. Their approach is able to address the rectangle packing problem with spatial cost function of the type $g(\max_i p_i(x_i), \max_i q_i(t_i))$, where p_i, q_i are general cost functions and can be discontinuous or nonlinear, and g is nondecreasing in its parameters. Given a fixed sequence pair, Imahori et al. (2002) showed that the associated optimization problem can be solved efficiently using a Dynamic Programming framework. Unfortunately, the general cost function considered in their paper cannot be readily extended to incorporate the objective function considered in this paper.

The berth allocation planning problem is also related to a class of multiple machines stochastic scheduling problems on jobs with release dates, where each job needs to be processed on multiple

processors at the same time, i.e., there is a given pre-specified dedicated subset of processors which are required to process the task simultaneously. This is known as the multiprocessor task scheduling problem. Li, Cai, and Lee (1998) focused on the make-span objective and derived several approximation bounds for a variant of the first-fit decreasing heuristic. However, their model ignored the arrival time aspect of the ships and is not directly applicable to the berthing problem in practice. In a follow-up paper, Guan et al. (2002) addressed the case with weighted completion time objective function. They developed a heuristic and performed worst case analysis under the assumption that larger vessels have longer port stays. This assumption, while generally true, does not hold in all instances of container terminal berthing. Fishkin et al. (2001), using some sophisticated approximation techniques (combination of scaling, LP, and DP), derived the first polynomial time approximation scheme for this problem, for the minimum *weighted completion time* objective function. However, in the berth planning problem, since release time of each job is given, a more appropriate objective function is to consider the weighted *flow time* objective function. Unfortunately, there are very few approximation results known for scheduling problems with mean flow time objective function.

The closest literature to our line of work is arguably the series of papers by Imai, Nishimura, and Papadimitriou (2001, 2003). In the first paper, they proposed a planning model where vessel arrival time is modelled explicitly and they proposed a Lagrangian-based heuristic to solve the integer programming model. Their model is a simplified version of our static berth allocation planning problem, since they implicitly assume that each vessel occupies exactly one berth in their model. The issue of re-planning is also not addressed in that paper. In the second paper, they proposed an integer programming model and a genetic algorithm for solving the static berth planning model with differentiated priorities, but their model assumes deterministic data and does not take into consideration the arrival time information.

3 Static Berth Allocation Planning Problem

Given a scheduling window and information on the vessels that will be arriving within the window, we structured the associated berth planning problem as one of packing rectangles in a semi-infinite strip with general spatial cost structure. In this phase, the packing algorithm must minimize the delays faced by vessels, with higher priority vessels receiving the promised level of services. At the same time, the algorithm must also address the desirability, from a port operator's perspective, to berth the vessels on designated locations along the terminal and to minimize the movement and exchange of containers within the yards and between vessels.

The input to the problem are:

- Terminal specifics:
  - W: Number of wharfs in the terminal,
  - w_i: Length of wharf i, $i = 1, \ldots, W$,
  - M: Number of berths in the terminal,

  - L_i: Length of berth i, $i = 1, \ldots, M$.

- Vessel specifics:
  - N: Number of vessels that have arrived or will arrive within the scheduling window,
  - r_i: Arrival time of vessel i,
  - l_i: Length-overall of vessel i,
  - p_i: Length of port stay upon berthing by vessel i.

In this paper, we assume that vessels can be moored at any berth available within the terminal. In real terminal berth allocation planning, we also have to take into account the issues of draft constraints, equipment type availability along the berth, tidal information and crane availability; and we have to make sure that the berth allocation plan will not violate these additional considerations. Furthermore, we note that although we take the vessel-specific parameters as given constants for each scheduling window, in reality, some of the parameters (such as arrival time within the scheduling window) may not be as forecast. In particular, the port stay time p_i depends on crane intensity (average number of cranes working on the vessel per hour) assigned to vessel i, and also on the number of containers to be loaded and discharged. This information, however, maybe not be accurate or available prior to the scheduling decision.

The decision variables to the berth planning problem are:

- x_i: Berthing location for vessel i, measured with respect to the lower end of the vessel,
- t_i: Planned berthing time for vessel i.

Given a berthing plan with prescribed decisions x_i and t_i, we can evaluate the quality of the decision by two cost components:

- **Location Cost**: The quality of the berthing locations assigned is given by

$$\sum_{i=1}^{N} c_i(x_i),$$

  where the space cost function c_i is a *step function* in x_i. Let d_{il} be the penalty for berthing vessel i at berth l. Then,

$$c_i(x_i) = d_{i,l} \text{ if vessel } i \text{ is berthed in berth } l,$$

  i.e.,

$$x_i \leq \sum_{k=1}^{l} L_k, \qquad x_i > \sum_{k=1}^{l-1} L_k.$$

  Note that $d_{i,l}$ indicates the desirability of berthing vessel i in berth l. This depends on where the loading containers are stored in the terminal and also on the destination of the discharging containers.

- **Delay Cost**: The quality of the berthing times assigned is given by

$$\sum_{i=1}^{N} w_i \Big(t_i - r_i - 2 \Big)^+ .$$

A vessel is considered "berthed-on-arrival" (BOA) if it can be berthed within two hours of arrival (i.e., within r_i and $r_i + 2$). This is the service level promised to some key customers. The value w_i is a penalty factor attached to vessel i and indicates the "importance" of berthing the vessel i on arrival (i.e., within a two-hour window). The vessels are normally divided into several priority classes with different penalty factors.

The optimal packing problem in this phase can be formulated as the following Static Berth Planning problem (SBP):

$$\min \quad \sum_{i=1}^{N} (c_i(x_i)) + \sum_{i=1}^{N} w_i (t_i - r_i - 2)^+ \tag{2}$$

$$s.t. \quad t_i \geq r_i, \ \forall \ i, \tag{3}$$

$$x_i + l_i \leq \sum_i L_i, x_i \geq 0, \ \forall \ i, \tag{4}$$

$$x_i \text{ is not berthed across different wharfs for all vessel } i, \tag{5}$$

$$t_i + p_i \leq t_j \text{ or } t_j + p_j \leq t_i \text{ or } x_i + l_i \leq x_j \text{ or } x_j + l_j \leq x_i \ \forall \ i \neq j. \tag{6}$$

The last constraint models the condition that vessels i and j cannot occupy the same location in the terminal at the same time. Note that this is a nonlinear optimization problem with non-convex feasible region and is hence a difficult optimization problem.

3.1 Sequence Pair Concept

We first show that every berthing plan can be encoded by a pair of permutations of all vessels (H, V). Consider the berthing plan of two vessels as shown in Figure 6.

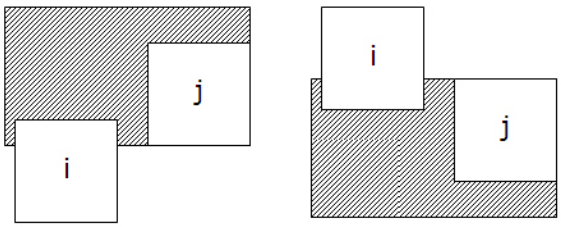

Figure 6: The figure on the left shows the LEFT-UP view of vessel j, whereas the figure on the right shows the LEFT-DOWN view of vessel j; the views seen by vessel j are shaded regions plus the parts blocked by vessel i.

The permutations H and V associated with the berthing plan are constructed with the following properties:

- If vessel i is on the right of vessel j in H, then vessel j *does not* "see" vessel i on its LEFT-UP view.
- Similarly, if vessel i is on the right of vessel j in V, then vessel j *does not* "see" vessel i on its LEFT-DOWN view.

It is clear that, given any berthing plan, we can construct a pair (H, V) (need not be unique) satisfying the above properties. For any two vessels a and b, the ordering of a, b in H, V essentially determines the relative placement of vessels in the packing. For the rest of the report, we write $a <_H b$ (and $a <_V b$) if a is placed on the left of b in H (resp. in V).

- If $a <_H b, a <_V b$, then a does not see b in LEFT-DOWN or LEFT-UP, i.e., vessel b is to the right of vessel a. In other words, vessel b can only be berthed after vessel a leaves the terminal.
- If $a <_H b, b <_V a$, then a does not see b in LEFT-UP and b does not see a in the LEFT-DOWN view, i.e., vessel b is berthed below vessel a in the terminal.

For any H and V, either one of the above holds, i.e., either vessel a and vessel b do not overlap in time (one is to the right of the other) or do not overlap in space (one is on top of the other).

Note that every sequence pair (H, V) corresponds to a *class* of berthing plans satisfying the above properties. The constraints imposed by the sequence pairs split into two classes: constraints of the type $x_i + l_i \leq x_j$ (in the space variables) or of the type $t_i + p_i \leq t_j$ (in the time variables). In this way, finding the optimal packing in this class, given a fixed sequence pair, decomposes into two subproblems: *space* and *time* (cf. Figure 7).

Given a fixed sequence pair, it will be ideal if the optimal berthing plan can be obtained in a LEFT-DOWN fashion, i.e., berthing each vessel at the *earliest time* and *lowest possible position* available, subject to the sequence-pair condition. This method has various other advantages, as the berthing plan for vessels can actually be constructed greedily in an iterative manner, allowing us to handle a host of other side constraints in real terminal berthing operations. Unfortunately, a LEFT-DOWN packing obtained with a fixed sequence pair may not always gives rise to the optimal packing, due to the step-wise feature of the berthing space cost.

3.2 Time Cost Minimization With Fixed Sequence Pair

Given (H, V), let G_T be the directed graph associated with constraints involving the berthing time variables t_i. The time-cost problem can be formulated as:

$$\min_{t_i} \sum_{i=1}^{N} w_i (t_i - r_i - 2)^+$$

subject to

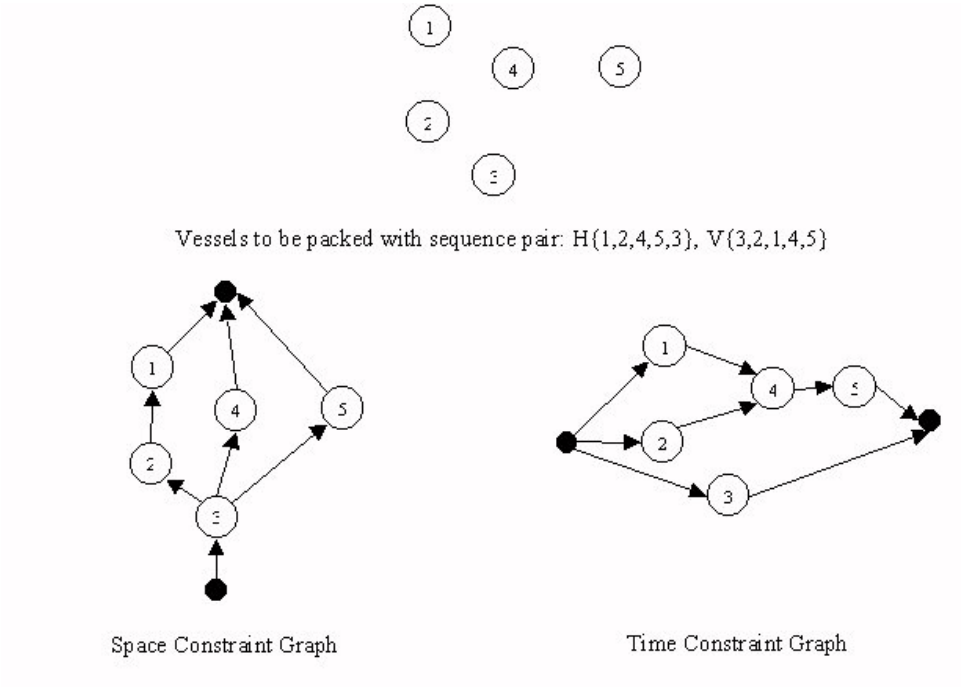

Figure 7: Directed Graphs on the space and time variables arising from the Sequence Pair

- Berthing time must be as large as arrival time: $t_i \geq r_i$ for all vessels.
- $t_i + p_i \leq t_j$ if $(i, j) \in G_T$.

Let

$$f_i(t_i) = w_i(t_i - r_i - 2)^+.$$

Since $f_i(\cdot)$ is a non-decreasing function in t_i, the optimal solution to the above is clearly to set t_i as small as possible, subject to satisfying all the constraints. This is the classical longest path problem in an acyclic digraph that can be solved easily using a simple dynamic-program. One can refer to Imahori et. al. (2003) for a dynamic programming based algorithm and Ahuja et. al. (1993) for a general discussion on longest path problems. In fact, each t_i is selected to be the smallest possible value satisfying the constraints and can be constructed in a greedy manner.

3.3 Space Cost Minimization With Fixed Sequence Pair

Given (H, V), let G_S be the directed graph associated with constraints involving the berthing location variables x_i. The space-cost problem can be formulated as:

$$\min_{x_i} \sum_{i=1}^{N} \sum_{i=1}^{N} c_i(x_i)$$

subject to

- Berthing location cannot extend beyond the terminal: $x_i + l_i \leq L$, $x_i \geq 0$.
- Berthing location cannot cross wharf: $x_i + l_i \leq \sum_{i=1}^{K} L_i$ or $x_i \geq \sum_{i=1}^{K} L_i$ if the berth K and berth $K+1$ belong to different wharfs.
- $x_i + l_i \leq x_j$ if $(i, j) \in G_S$.

Given arbitrary step function $c_i(\cdot)$, solving the above problem is exceedingly difficult. In the case when all $l_i = 0$ and G_S corresponds to a Hamiltonian Path, the above is just the classical nonlinear ordered set problem. In fact, berthing the vessel according to the lowest possible position satisfying the constraints may not always give rise to good packing. LEFT-DOWN packing will not produce a good solution in terms of space cost.

To address this problem, we outlined a method where the space cost minimization problem can be handled implicitly by extending the search space. We exploit the observation that the space cost function are essentially step functions that depend on the number of berths in the terminal. Note that the number of berths in a terminal is relatively small (compared to the number of vessels). Furthermore, the space objective cost-function is constant within a berth.

In the special case where $c_i(\cdot)$ is a constant function, the space cost minimization problem can be solved efficiently as a longest path problem in G_S, as in the previous case. In this instance, each vessel is berthed at the lowest possible position satisfying the precedence constraints and wharf-crossing constraints. For general $c_i(\cdot)$, unfortunately, we need to consider all possible berthing locations for vessel i in search of better berthing cost. This is the major bottleneck in the search for an optimal solution in this problem.

For each berth l, we introduce a *virtual wharf mark* $w(l)$ indicating a vessel with $r_{w(l)} = 0, l_{w(l)} = 0, p_{w(l)} = 0$, with additional constraint (lower-bound on berthing location)

$$x_{w(l)} \geq \sum_{i=1}^{l-1} L_i.$$

Note that the berthing plan on the left side of Figure 8 can be obtained from the sequence pair $(12w34, 431w2)$ by berthing the vessels (real and virtual) using the left-down berthing approach. The introduction of the virtual wharf marks in the sequence pair and the added constraints on the position of the wharf marks allow us to incorporate gaps into the berthing position of the vessels, and has virtually no impact on the berthing time. Note that it is not known, before hand, how many wharf marks will be needed to prop up the vessels to the desired locations in the terminal. But we note that for a sequence pair, with the right number of wharf marks distributed appropriately, we can obtain the optimal packing by using the LEFT-DOWN packing strategy. We record the observation in the following theorem.

Theorem 1. *Suppose $\mathcal{P}^*$ is the optimal berthing plan to the problem, minimizing the total berthing time and space cost. Then there exists a set of virtual wharf marks $\{w_1, \ldots, w_K\}$, for some K, such that $\mathcal{P}^*$ is equivalent to a berthing plan $\mathcal{Q}^*$, involving all the vessels and virtual wharf marks, such that $\mathcal{Q}^*$ is obtained from a corresponding LEFT-DOWN packing using some sequence pair H^{**} and V^{**}.*

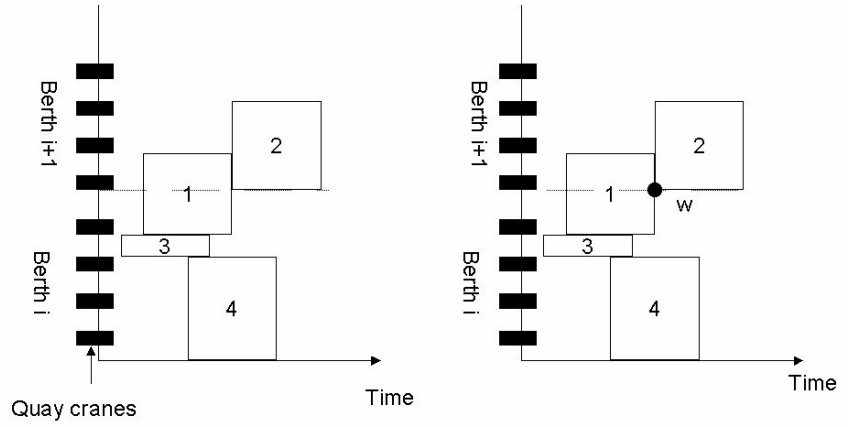

Figure 8: Berthing plan with virtual wharf mark as shown on the right. The sequence pair changes from (1234, 4312) to $(12w34, 431w2)$

Proof: See Appendix II.

In general, finding the optimal number of wharf marks and their associations with the vessels is extremely difficult. It also does not pay to find the optimal packing at every stage because the sequence pair may not correspond to the optimal solution. The main advantage of this approach is that given *any* feasible packing, we can, through introduction of virtual wharf marks, encode the packing as one obtainable from LEFT-DOWN from an associated sequence pair in an enlarged space. This allows us to explore more complicated neighborhoods in the simulated annealing procedure and its advantage will be evident in the next section.

3.4 Neighborhood Search Using Simulated Annealing

The approach outlined in the earlier section gives rise to an effective method to obtain a good packing, given a fixed sequence pair. We next use a simulated annealing algorithm to search through the space of all possible (H, V) sequence pairs. There are definitive advantages in using simulated annealing here, because now all the permutations of both the H and V sequence can be explored in a systematic manner.

The critical aspect for getting good solutions is to define a sufficiently large neighborhood that can be explored efficiently. To this end, we use the following neighborhood structure:

(a) Single Swap: This is obtained by selecting two vessels and swapping them in the sequence by interchanging their position. Single swap is defined when the swap operation is performed in either H or V sequence.

(b) Double Swap: Double swap neighborhood is obtained by selecting two vessels and swapping them in both H and V sequences.

(c) Single Shift: This neighborhood is obtained by selecting 2 vessels and sliding one vessel along the sequence until the relative positions are changed; i.e., if $i, j, \ldots, k, l$ is a subsequence, a shift operation involving i and l could transform the subsequence to $j, \ldots, k, l, i$. There are many variants of this operation depending on whether vessel i (or l) is shifted to the left or right of vessel l (or i). We define single shift as a shift operation along one of the sequences.

(d) Double Shift: This defines the neighborhood obtained by shifting along both H and V sequences.

Figure 9 shows examples of the above operations and their impact on the packing. The above neighborhoods described are simple perturbations of the sequence pair, but they result in remarkably different packing when compared visually. On the other hand, it is pretty difficult to modify the sequence pair to obtain visually simple neighborhoods. For example, shifting arcs from space graph to time graph and vice versa between two vessels i and j without affecting the constraints between the rest of the vessels and i and j is hard to obtain, though, visually, it may be as simple as sliding vessel i from the top to the bottom of vessel j. Due to the non-linear space cost, we observe that such neighborhoods would allow for shifting of vessels between different berths and hence provide improvement in the solution. This motivated us to define the next class of neighborhood structure.

(e) Greedy Neighborhood: Given a sequence pair H and V and the associated packing $\mathcal{P}$, we evaluate all possible locations that vessel i can take, with the rest of the vessels fixed in their respective positions. If there is a better location for the vessel, then we set the berth location of i to its new location. Note that since the time is kept constant, it is easy to check whether there is an overlap along the space dimension. Once the vessel is placed in the new location, we repeat the procedure for the rest of the vessels, until no improvement is possible.

Figure 10 shows the packing and the corresponding sequence pair obtained from a simple greedy neighbor.

The greedy neighborhood artificially modifies the position of the vessel along space. Hence we may have cases where some vessels are placed on top of other vessels but none of the arcs in the space graph result in binding constraints. Fortunately, since the berthing space cost is constant within every berth, we have the following conditions for the berthing location for each vessel after the greedy modification: (i) the vessel i physically sits on top of another vessel, i.e., $x_i = x_j + l_j$ where j is a vessel that overlaps with i along time; or (ii) $x_i = \sum_{i=1}^{k-1} L_i$ for some berth k.

In case (ii) above, we introduce a virtual wharf-mark to the sequence pair to create the constraint that the vessel should be berthed above berth edge k.

Cooling schedule for simulated annealing: It is well known that the choice of the cooling schedule affects the results in simulated annealing since it is critical in the systems ability to move away from local optima. For example, refer to Lin (1999). We evaluate the objective for the packing as the sum of the time and space costs where *time cost* and *space cost* are defined as in sections 3.2 and 3.3. We did some tests using different cooling schedules and empirically determined that the following schedule works well in our scenario:

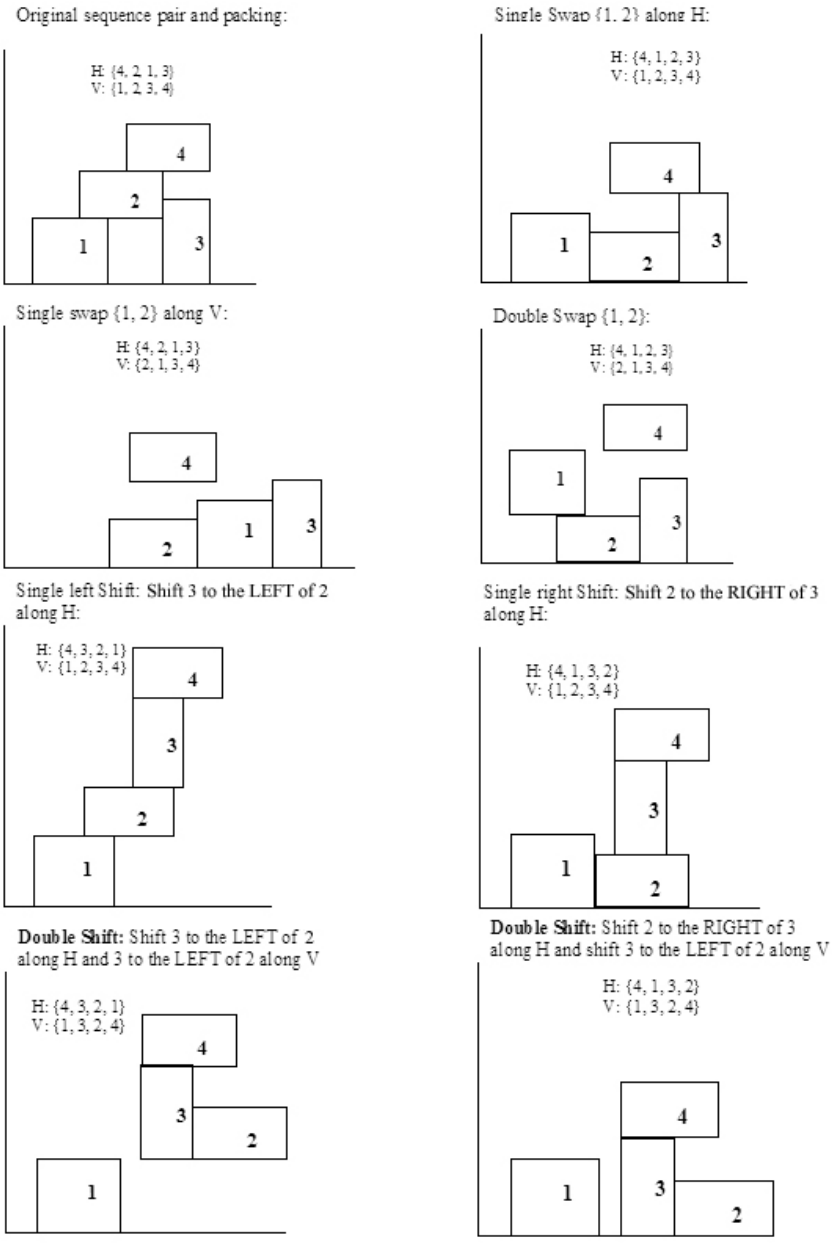

Figure 9: Examples of swap and shift neighborhoods

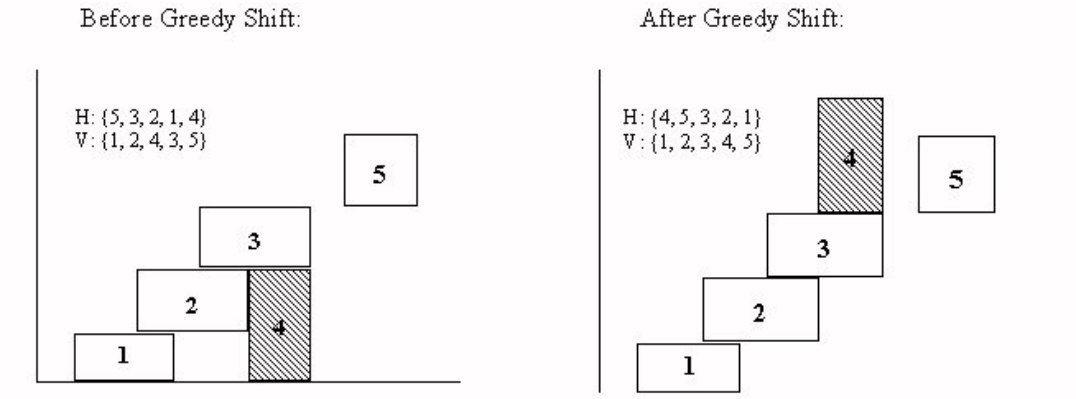

Figure 10: Examples of greedy neighborhood

$$T_0 = \frac{-Z_0}{\ln 0.95},$$

$$T_i = T_{i-1} e^{-Ci}.$$

The initial temperature T_0 is determined based on the objective (Z_0) of a greedily generated initial packing. The constant, 0.95, is an arbitrary choice and represents the probability of moving to inferior solutions during the first iteration of simulated annealing. After searching the neighborhood, we update the temperature T_i as shown. C is a constant and in our computation, we choose $C = 0.99$.

While searching at a specific temperature T_i, we denote the objective value at the beginning of the search as $\hat{Z}$ and the objective value for the current neighbor as Z_j. The algorithm then moves to the new neighbor with the following probability:

- Probability of 1 if $Z_j < \hat{Z}$.
- Probability of $e^{(\hat{Z}-Z_j)/T_i}$ if $Z_j \geq \hat{Z}$.

The algorithm is terminated when it reaches steady state (i.e., no further improvement can be achieved in a few iterations) or when $T = 0$.

Implementation incorporating virtual wharf mark: The problem in adding virtual wharf marks as additional vessels in the search space is that it increases the problem size and hence the computation time. Here, we propose a cost-effective way of implementing the approach by employing dynamic lower bounds.

Note that the vessels need to be propped after we employ the greedy neighborhood. Instead of adding virtual wharf marks, the idea is to (dynamically) set lower bounds for the berthing location for those vessels that need to be propped by a virtual wharf mark in the packing. We

retain these lower bounds while exploring the neighborhood using operators (a)-(d), and change the lower bounds only when operator (e) changes the packing and introduces new virtual wharf marks.

The dynamic lower bounding technique described above is equivalent to adding a virtual wharf mark w to i (with w coming immediately after i in the H sequence, and w immediately before i in the V sequence), and performing all neighborhood searches treating iw in H, and wi in V, as "virtual" vessel. Note that swapping or shifting iw with j in H, or swapping or shifting wi with j in V, has the same effect of swapping or shifting i with j in the original H sequence, but maintaining a lower bound (determined by virtual wharf mark w) on vessel i in the neighborhood.

3.5 Lower Bound For Static Berth Planning Problem

To evaluate the performance of the proposed approach, we need to compare the performance against a suitable lower bound. To this end, we use tools from mathematical programming to obtain a lower bound to our berth planning problem.

The berth planning problem can be structured as a set packing problem with side constraints. Let $\mathcal{C}_i$ denote the set of allowed positions that vessel i can be berthed in the schedule. The set $\mathcal{C}_i$ takes care of all the constraints in berthing positions (eg., no berthing across wharfs) and berthing times (eg., not earlier than arrival) of vessel i. The side constraints ensure that the vessels will not overlap.

Using standard Lagrangian relaxation approach, we can structure the problem as a set packing problem to construct a lower bound to the problem. The bound is obtained by solving the following:

$$\max_{\pi(x,t)} \min_{(x_i,t_i)\in\mathcal{C}_i} \sum_{i=1}^{N} \left(c_i(x_i) + w_i f_i(t_i) + \int_{x_i}^{x_i+l_i} \int_{t_i}^{t_i+p_i} \pi(x,t)dxdt \right) - \int_0^L \int_0^\infty \pi(x,t)dxdt.$$

The variables $\pi(x,t)$ are the dual prices associated with the non-overlapping constraints (for the position (x,t) in space). We use a version of subgradient algorithm, known as the volume algorithm in the literature (cf. Barahona and Anbil (2000)), to solve the above problem.

To make the problem manageable so that the lower bound can be obtained within reasonable time, the space-time network is discretized to moderate sizes using the following scaling technique:

- Let $l = \alpha L$, where L is the length of the terminal, α is a scaling constant.
- For x in $(0, L)$, define

$$u(x) = \begin{cases} x & \text{if } \frac{x}{L}(1+\frac{L}{l}) \text{ is an integer,} \\ \lfloor (\frac{L}{l}+1)\frac{x}{L} \rfloor l & \text{otherwise.} \end{cases}$$

  recall, for any real number x, $\lfloor x \rfloor$ is the largest integer smaller than x.

Note that for the problem considered, and since all vessel lengths are integers, we do not expect to encounter instances where the vessel length x is such that $\frac{x}{L}(1 + \frac{L}{l})$ is an integer. We may thus assume all vessel lengths are scaled to multiples of l. This helps to reduce the size of the problem considerably.

Proposition 1 ([10]). *If* $\sum_{i \in S} x_i \leq L$, *then* $\sum_{i \in S} u(x_i) \leq L$.

Note that the scaled problem will still provide us with a lower bound for the static berth planning problem.

3.6 Computational Results

In this section, we compare the performance of the virtual wharf mark approach with that of the lower bound. We also show the difference in computational time taken to solve the problem. We report computational results on problem sizes including 30, 50, and 70 vessels. The instances have varying resource utilization (RU) levels. Average resource utilization is a measure of traffic load at the terminal and the max resource utilization measures the peak demand for berthing space within the scheduling window. For each problem size, we compare the computational performances under two different scenarios:

- Congested Scenario: In this situation, all vessels available for packing arrive at time zero, thus creating intense competition for usage of the terminal. The instances in this scenario are created with high average RU.
- Light Load Scenario: In this case, all vessels can be berthed immediately upon arrival and at the most preferred berth. We do this working backward, i.e., starting with a packing, we devise cost parameters to the problem so that the packing is indeed the optimal solution. This is constructed by choosing appropriate values for vessel arrival time and preferred berthing locations. The average RU in these instances is typically lower and the max RU is always less than 1.

Figure 11 shows a typical plot of the convergence of the solution for simulated annealing and the volume algorithm over consecutive iterations for a congested scenario. For sake of plots, only the final stages of simulated annealing iterations are plotted.

The computational performance for the experiments (averaged over 10 random instances) is summarized in Tables 1 and 2. Note that the proposed lower bound is understandably weak, since the Lagrangian relaxation model relaxes the non-overlapping constraints in the problem. The simulated annealing (SA) algorithm, on the other hand, is able to return a solution close to 25%-36% of this lower bound (LB), depending on whether space cost is included in the model or not. We believe that the quality of the solution obtained by the algorithm is much better. This is confirmed by the light load cases, where the simulated annealing algorithm is able to consistently return close to optimal solution in all 10 random instances ($< 0.33\%$ vessels delayed and $< 6.3\%$ vessels placed in inferior locations).

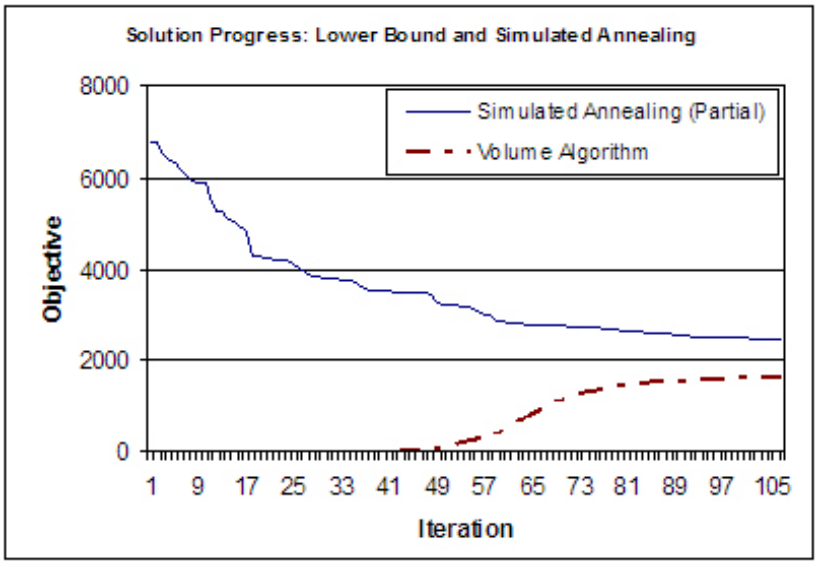

Figure 11: Convergence of objective values for simulated annealing and lower bound for a congested scenario

# Vessels	Avg RU %	Max RU %	Space Cost	Lower bound (LB)	Running Time (Sec)	Heuristic Solution (SA)	Running Time (Sec)	(SA - LB) / LB
30	60	250	No	613.8	52	794.3	15	0.32
			Yes	1645.6	54	2186.4	16	0.36
50	80	420	No	2119.6	69	2769.1	148	0.31
			Yes	2645.4	69	3468.9	155	0.33
70	75	580	No	5588.6	165	7260.3	643	0.31
			Yes	6503.1	165	8078.8	661	0.25

Table 1: Comparison with lower bound: congested scenario

# Vessels	Avg RU %	Max RU %	Vessels Delayed	Vessels in Inferior Berth
30	25	60	0.33 %	2.67 %
50	28	65	0.00 %	3.40 %
70	37	82	0.29 %	6.29 %

Table 2: Performance of simulated annealing: light load scenario

4 Dynamic Berth Allocation Planning Model

In the dynamic berth allocation planning model, we have the additional challenge of rolling the plan forward every time we move forward one period. This raises an important question: How do we handle the situation when certain vessels are already berthed in the terminal? Note that the space allocated to these vessels cannot be used to berth other vessels.

We model these constraints as "forbidden zones" along the left edge of the berthing network. In the presence of infeasible regions in the packing area, to obtain one-to-one correspondence between the sequence pair and the packing, we use a variant of the lower bounding strategy, which can be viewed as an extension of the virtual wharf mark approach. In the earlier results by Murata, Fujiyoshi, and Kaneko (1998), infeasible regions were modeled as pre-placed vessels and a greedy strategy was used to modify a packing to ensure that pre-placed vessels are restored to their original locations. But with a complexity of $O(N^4)$ for instances with N vessels, it is computationally expensive. Herein we develop a strategy to specifically cater to the issue of packing during dynamic deployment. Figure 12 shows a typical scenario.

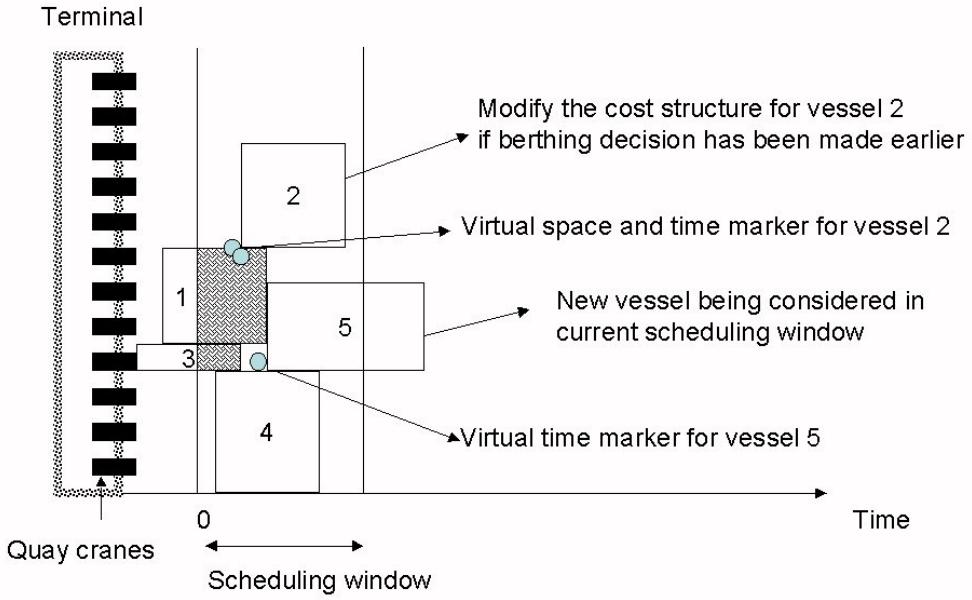

Figure 12: Handling pre-placed vessels

Along space, if a vessel is not supported below by another vessel (through the sequence-pair conditions), then we set an appropriate lower bound for the berthing location decision of this vessel. Along time, if the vessel is adjacent to an infeasible region and there are no binding arcs in the time-graph, we can also set an appropriate lower bound to the berthing time decision of these vessels.

An initial packing is obtained using a greedy insertion strategy. The neighborhoods are then explored using the previous neighborhood structures, with the additional lower bound constraints

on berthing time and location decisions. We omit the details of the implementation here.

Revisions to berthing plan: The ability to build-in the latest arrival information and to revise plans is definitely an advantage to the terminal operator in a dynamic setting; but frequent revision of the berthing plan is not desirable from a resource planning perspective. This is due to the fact that, based on the berthing plan, the port will normally have to plan the yard load and activate needed resources within the terminal to support loading/discharging activities from the vessels. In a container port, the following major resource-hogging events are affected while servicing a particular vessel: Quay Cranes, Yard Cranes, Prime Movers, Operator-Crew, Container Movement, etc. Typically, the container and resource management in the yard is a complex problem and is done beforehand based on the berth plan. Hence, any major deviation from the plan in a vessel's location or service time, results in a major re-shuffle of crew and prime mover allocations. It also results in re-deployment of cranes. Further, last minute changes to the berthing plan and re-deployment of personnel and equipment lead to confusion. Hence it is preferable that a proposed berth plan can be executed with minimal changes.

To ensure that the berthing plan will not be revised too frequently, the terminal operator can opt to freeze the berthing location decision made during earlier scheduling windows. Another alternative is to penalize deviation from an earlier plan by imposing a large penalty for changing the berthing location or time for a vessel. We opt for the second strategy, where the space and time cost function for vessel i is adjusted to

$$c_i'(x) = \begin{cases} 0 & \text{if } x = x_i, \\ A & \text{otherwise,} \end{cases}$$

$$f_i'(t) = A(t - t_i)^+,$$

where x_i, t_i are the earlier berthing decision, and A is a huge penalty term. The choice of $c_i'(\cdot)$ and $f_i'(\cdot)$ essentially forces vessel i to be berthed at the original berthing location and time.

5 Berth Planning With Throughput Consideration

The previous sections outlined a method to allocate berthing space to vessels in a dynamic environment. However, as shown in the earlier example, it does not preclude the possibility that certain classes of vessels will be delayed indefinitely and hence will not be berthed at all in the assigned terminal. This indirectly reduces the berth utilization and throughput of the terminal, which is a primary concern in berth planning. In this section, we propose a strategy to redistribute load within the terminal at periodic intervals, and we show that in this way, the throughput of the terminal will be maximized.

To study the issue of throughput in the berth allocation planning problem, we use the framework of stochastic processing networks. We first discretize the berth space into integer units, indexed by $k = 1, 2, \ldots, K$. Each unit of the space is called a section. Each section can accommodate one vessel each time, and each vessel occupies an integer number of consecutive sections

when it is moored. For ease of exposition, we group the vessels into categories or *classes* by the number of sections they require[3]. All the vessels of the same class require the same number of sections. The classes of vessels are indexed by $i = 1, 2, \ldots, I$. Let $K(i)$ be the number of sections required by class i vessel. When no berth space is available, the vessel has to wait for mooring. For each class of vessels, we create a (virtual) queue to buffer waiting and mooring vessels. This is a special class of the general stochastic processing networks advanced by Harrison (2000). An *activity* here accommodates a certain class of vessels and uses certain consecutive sections. Each activity is determined by the vessel class and the first section assigned to the activity, and the activities are indexed by $j = (i, k), i = 1, 2, \ldots, I, k = 1, 2, \ldots, K - K(i) + 1$. If activity $j = (i, k)$ is taken, then the sections $k, k + 1, \ldots, k + K(i) - 1$ are assigned to accommodate (process) a class k vessel. Denote J to be the total number of activities. We define the capacity consumption matrix A such that $A_{kj} = 1$ if activity j requires section k and $A_{kj} = 0$ otherwise. Each activity takes care of one vessel each time. When several vessels are moored in the berth, it is possible that several activities are active at the same time. An *allocation* is a possible set of active activities, which is defined by a binary J-vector a such that a_j equals 1 if activity j is active in the allocation and 0 otherwise. We define the set of all possible allocations $\mathcal{A} = \{a : Aa \leq e, a_j = 0, 1 \text{ for each activity } j\}$, where e is the K-dimensional vector of ones. A scheduling policy dictates which allocation to use at any given time.

We use the following example to illustrate our model. Consider a wharf of length 200m. We divide it into 4 equal length sections (servers). If we assume all the vessel lengths are between 75m and 125m, then the vessels are grouped into 2 classes. The vessels with length less than 100m occupy two sections and belong to class 1; the vessels with length between 100m and 125m require three sections and belong to class 2. The stochastic processing network model for this example is depicted in Figure 13. Open rectangles represent buffers, circles represent sections, and activities are labeled on lines connecting buffers with sections. For terminology simplicity, the stochastic processing network model for the berth allocation problem is referred to as the *berth allocation network*.

Let $E_i(t)$ be the cumulative number of class i vessels that have arrived by time t. Let $D_i(t)$ be the cumulative number of class i vessels that have departed by time t. For the stochastic processing network representation of the berth allocation problem, we assume the arrival rates (λ_i) are well defined. Namely, with probability one,

$$\lim_{t\to\infty} \frac{1}{t} E_i(t) = \lambda_i \quad \text{for each vessel class } i.$$

The berth allocation network is said to be *rate stable* if for every initial configuration, with probability one,

$$\lim_{t\to\infty} \frac{1}{t} D_i(t) = \lambda_i \quad \text{for each vessel class } i.$$

In short, the berth allocation network is rate stable if the departure rate is equal to the arrival

[3]In practice, these vessels may differ in preferred berth allocation. However, as we will focus on the throughput attained in this section, we do not distinguish between these vessels.

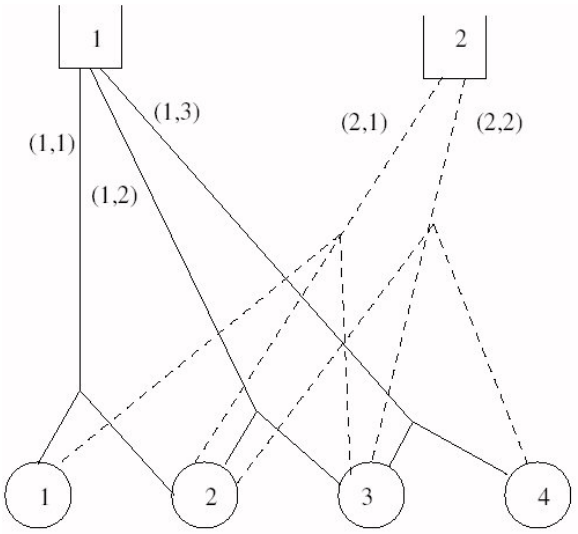

Figure 13: A stochastic processing network model for berth allocation problem

rate for each vessel class. Obviously, if the offered load λ_i is too high, the berth allocation network will not be rate stable no matter which scheduling policy is employed.

5.1 A Static Allocation Problem

Let $\eta_j(n)$ ($j = (i,k)$) be the length of port stay upon berthing for the nth class i vessel moored in sections k to $k + K(i) - 1$. The port stay length is allowed to be random. We assume that the processing rate μ_j is well defined. That is, with probability one,

$$\lim_{n\to\infty} n^{-1} \sum_{\ell=1}^{n} \eta_j(\ell) = m_j,$$

where $m_j = 1/\mu_j$ is the average length of port stay for vessels that are processed by activity j. Let $R_{ij} = \mu_j$ if activity j takes care of class i vessels and $R_{ij} = 0$ otherwise. R_{ij} is the average number of class i vessels that can be accommodated by a unit of activity j. We define the allocation $a = 0$ as *idle allocation* since all the servers (sections) are idle under this allocation. We characterize the stability condition by the following linear program:

$$\begin{aligned}
\min \quad & \rho && (7)\\
s.t. \quad & Rx = \lambda, && (8)\\
& x = \sum_{a\in\mathcal{A}\setminus\{0\}} a\pi_a, && (9)\\
& \sum_{a\in\mathcal{A}\setminus\{0\}} \pi_a = \rho, && (10)\\
& \pi_a \geq 0 \text{ for each } a \in \mathcal{A} \setminus \{0\}. && (11)
\end{aligned}$$

This linear program is closely related to LP (47)–(50) that was first introduced in Dai and Lin (2003) when preemption of jobs is not allowed. For each allocation $a \in \mathcal{A} \setminus \{0\}$, π_a is interpreted as the long-run fraction of time that allocation a is employed. Then for each activity j, x_j can be interpreted as the long-run fraction of time that activity j is active. With this interpretation, the left side of (8) is interpreted as the long-run net flow rate from the buffers. Equality (8) demands that exogenously inputs are processed to completion without other inventory being generated. ρ is interpreted as the long-run fraction of time that at least one server is busy. In other words, ρ is the utilization of the busiest server. This is different from the overall server utilization, which averages the utilization over all servers. It follows from Theorem 5 of Dai and Lin (2003) that the following theorem holds.

Theorem 2. *If the berth allocation network is stabilizable, then the linear constraints (8)-(11) have a feasible solution with* $\rho \leq 1$.

5.2 Discrete Maximum Pressure Policies

We show that the converse of Theorem 2 is also true when processing times are bounded. That is, as long as ρ defined by LP (7)–(11) is strictly less than 1 and the processing times are bounded, then there is a scheduling policy under which the system will be rate stable.

We assume all the processing times are bounded by a constant U. For each class i, we define $Z_i(t)$ to be the number of class i vessels that have arrived at the terminal either waiting to be served or being served at time t. We now define a family of policies, called *discrete maximum pressure* policies. They are adapted from Stolyar's MaxWeight policy. See Andrew et al. (2001) and Stolyar (2002). The presentation here follows closely the definition of maximum pressure policies in Dai and Lin (2003). Define the pressure of a berth allocation network with buffer level z under allocation a to be

$$p(a, z) = \sum_j a_j \sum_i R_{ij} z_i = z \cdot Ra.$$

The discrete maximum pressure policy with length $L \geq U$ is defined as follows.

Discrete Maximum Pressure

- The time horizon is divided into cycles of length L.
- At the beginning of each cycle, an allocation with *maximum pressure* is selected, and each section is assigned to the vessel class given by the allocation until

  (i) the corresponding buffer is empty or

  (ii) it finishes a job and can not finish the next one within the cycle.

  In both cases, the section is released and is ready to be assigned to any feasible activity that can be finished within the cycle.
- One has flexibility to assign the released sections. We can allocate space from the released section according to the objective outlined in the previous section, or we can do so such that the number of the remaining idle sections is minimized.

Theorem 3. *Consider the berth allocation networks under any type of discrete maximum pressure policy with length L. If all the processing times are bounded by U, and the linear program (8)-(11) has a feasible solution with $\rho \leq \rho_0 \equiv (L-U)/L$, then the system is rate stable.*

Variants of maximum pressure policies were first advanced by Tassiulas and Ephremides (1992) under different names for scheduling a multihop radio network. Their work was further studied by various authors for systems in different applications; see Tassiulas and Ephremides (1992), Tassiulas (1995), McKeown et al. (1999), Dai and Prabhakar (2000), and Tassiulas and Bhattacharya (2000).

To illustrate how the discrete maximum pressure policy works, consider the application of this policy in the example in Section 1. Note that the objective there is to maximize berth utilization within every scheduling window of 32 hours. It is easy to calculate $\rho = 0.9$ for the example. We choose $L = U/(1-\rho) = 160$ hours. Again, we only consider the activities $(1,1),(1,5),(2,1),(2,4),(3,1)$. Let allocation $a^1 = (0,0,1,0,0)'$ be the allocation such that only activity $(2,1)$ is active, allocation $a^2 = (0,1,1,0,0)'$ be the allocation such that only activities $(2,1)$ and $(1,5)$ are active, allocation $a^3 = (0,0,0,0,1)'$ be the allocation such that only activity $(3,1)$ is active, allocation $a^4 = (0,0,0,1,0)'$ be the allocation such that only activity $(2,4)$ is active, and allocation $a^5 = (1,0,0,1,0)'$ be the allocation such that only activities $(1,1)$ and $(2,4)$ are active.

No class 3 vessel arrives in the first cycle $(0,160)$, and it follows the argument in Section 1 that only activities $(2,1)$ and $(1,5)$ can be active by time $L = 160$ under our policy. Moreover, the allocation of the servers are the same as shown in Figure 3 during most of the cycle. However, some activities may be forced to be idle when they are close to the end of the cycle in order to make all sections available at the beginning of the next cycle. In particular, activity $(1,5)$ stays idle from time 149 because another class 1 job will arrive at 153 and it can not be finished until time 167, while activity $(2,1)$ can finish the last job in the cycle at time 158. It is easy to see that the buffer levels are $Z_1(160) = Z_2(160) = 1$ and $Z_3(160) = 0$.

Then, at time $t = 160$, the beginning of the second cycle, allocations a^2 and a^5 are the maximum pressure allocations. Again without loss of generality, we choose allocation a^2, and activities $(2,1)$ and $(1,5)$ are employed. Similar to the first cycle, under maximum utilization packing scheme only activities $(2,1)$ and $(1,5)$ can be active and all except the first two vessels are processed upon their arrival. At the end of the cycle, we have $Z_1(320) = Z_2(320) = 1$ and $Z_3(320) = 0$. The third and fourth cycles will repeat the second cycle.

At time 640, the beginning of the fifth cycle, there is a class 3 arrival, so $Z_1(640) = Z_2(640) = Z_3(640) = 1$. Allocations a^2 and a^5 are still the maximum pressure allocations as at the beginning of the previous 3 cycles. And under OUP packing, all vessels except the first two in the cycle will be processed upon their arrival, so $Z_1(t) = Z_2(t) = Z_3(t) = 1$ at time $t = 800$. The same goes for cycle 6, 7, and 8, so at time $t = 960, 1120$, $Z_1(t) = Z_2(t) = Z_3(t) = 1$, and at time $t = 1280$, $Z_1(t) = Z_2(t) = 1$ and $Z_3(t) = 2$ because the second class 3 vessel arrives at time $t = 1280$, the beginning of the ninth cycle.

Following the same argument, one can easily check that a^2 and a^5 are still the maximum

pressure allocations at the beginning of cycle 9 to cycle 11, and at time $t = 1440, 1600, 1760$, $Z_1(t) = Z_2(t) = 1$ and $Z_3(t) = 2$. At time $t = 1920$, the beginning of the thirteenth cycle, the third class 3 vessel arrives, so $Z_1(1920) = Z_2(1920) = 1$ and $Z_3(1920) = 3$.

Now, the allocation a^3 becomes the maximum pressure allocation. Activity $(3, 1)$ will be employed from time 1920. Note that buffer 3 has 3 vessels to be served and each requires 16 hours processing time, so at time $t = 1968$, buffer 3 is empty and all the sections are released, and $Z_1(t) = Z_2(t) = 4$ and $Z_3(t) = 0$ because there are 3 arrivals for each of class 1 and 2. Under OUP packing, the sections are assigned to one class 1 and one class 2 vessel during the rest of the cycle from time 1968 to time 2080. During this time interval, there will be 7 arrivals for each class 1 and 2. On the other hand 9 class 1 and 8 class 2 vessels are served during the period, so $Z_1(2080) = 2$, $Z_2(2080) = 3$ and $Z_3(2080) = 0$.

Therefore, at the beginning of the fourteenth cycle, all the class 3 vessels left while two class 1 and three class 2 (including the one arriving at time 2080) vessels are waiting for mooring. The allocations a^2 and a^5 become the maximum pressure allocations again at time 2080. It is easy to check that during the fourteenth cycle $(2080, 2240)$, 10 vessels arrive for each class 1 and 2 and 11 vessels of each class 1 and 2 are served during the cycle, so $Z_1(2240) = 1$, $Z_2(2240) = 2$ and $Z_3(2240) = 0$. Similarly, we have at time 2400, the beginning of the sixteenth cycle, $Z_1(2400) = 1$, $Z_2(2400) = 1$ and $Z_3(2400) = 0$. Since one class 3 vessel arrives at time 2560, the beginning of the seventeenth cycle, $Z_1(2560) = 1$, $Z_2(2560) = 1$ and $Z_3(2560) = 1$. Hence $Z(2560) = Z(640)$. That is, the buffer size at the beginning of the seventeenth cycle is exactly the same as that of the fifth cycle. And due to the periodicity of the arrivals, one can easily verify that the dynamics repeat every 1920 hours (12 cycles). Therefore, the system is rate stable.

5.3 Proof of Theorem 3

For a given buffer level z, define

$$\mathcal{M}(z) = \{\operatorname*{argmax}_{a \in \mathcal{A}} p(a, z)\}.$$

$\mathcal{M}(z)$ is the set of allocations maximizing the network pressure given the buffer level z. For any allocation a, we define $\bar{a}(z)$ such that

$$\bar{a}_j(z) = \begin{cases} a_j, & z_{i(j)} \neq 0, \\ 0, & z_{i(j)} = 0, \end{cases}$$

where $i(j)$ denotes the constituent buffer associated with activity j. We call $\bar{a}(z)$ the *support allocation* of allocation a given the buffer level z. The buffer level of the constituent buffer of any activity in a support allocation is positive.

Lemma 1. *If $a \in \mathcal{M}(z)$, then $\bar{a}(z) \in \mathcal{M}(z)$.*

Proof.

$$p(a,z) = \sum_j a_j R_{i(j),j} z_{i(j)} = \sum_{j, z_{i(j)}>0} a_j R_{i(j),j} z_{i(j)} = \sum_j \bar{a}_j(z) R_{i(j),j} z_{i(j)} = p(\bar{a}(z), z).$$

■

Lemma 1 says that for any buffer level z, we can always choose a support allocation with maximum pressure. That is, there exists a maximum pressure allocation a^* such that $z_{i(j)} > 0$ if $a^*_j > 0$. Define $\bar{\mathcal{M}}(z)$ to be the set of maximum pressure support allocations. Let $\mathcal{I}(z)$ be the set of buffers that have potential positive output flows under any maximum pressure support allocation. Namely,

$$\mathcal{I}(z) = \Big\{ i(j) : a_j > 0, a \in \bar{\mathcal{M}}(z) \Big\}.$$

Notice that $z_i > 0$ for any $i \in \mathcal{I}(z)$ because any $i \in \mathcal{I}(z)$ is a constituent buffer of some activity in a support allocation.

Lemma 2. *For any allocations a^1, a^2, if $a^1 \geq a^2$ then $p(a^1, z) \geq p(a^2, z)$ for any $z \geq 0$. Here $a^1 \geq a^2$ means $a^1_j \geq a^2_j$ for any j.*

Proof. Notice that $R_{i(j),j} \geq 0$ for all j. Therefore,

$$p(a^1, z) = \sum_j a^1_j R_{i(j),j} z_{i(j)} \geq \sum_j a^2_j R_{i(j),j} z_{i(j)} = p(a^2, z).$$

■

Recall that $Z_i(t)$ counts the number of class i vessels at the terminal at time t. We use $Z(t)$ to denote the corresponding vector. For each allocation a, we use $T_a(t)$ to denote the cumulative amount of time that allocation a has been used by time t, and $T(t)$ to denote the corresponding vector $T_a(t)$, $a \in \mathcal{A}$. In the following lemma and the rest of this section, fluid limits of the berth allocation network are used. They are obtained as limits of T and Z under a law-of-large-numbers type scaling. Equations that are satisfied by fluid limits are said to be fluid model equations. These equations define a fluid model of the corresponding berth allocation network. Fluid limits and fluid model are now quite standard objects in the stochastic processing network literature; see, for example, Section 5.3 of Dai and Lin (2003). In the following, $(\bar{Z}, \bar{T})$ denotes a fluid limit of the berth allocation network operating under a discrete maximum pressure policy.

Lemma 3. *Under discrete maximum pressure policy, each fluid limit $(\bar{Z}, \bar{T})$ satisfies the fluid model equation:*

$$\sum_{a \in \mathcal{M}(\bar{Z}(t))} \dot{\bar{T}}_a(t) \geq (L-U)/L, \quad t \geq 0. \tag{12}$$

Proof. Let $(\bar{Z}, \bar{T})$ be a fluid limit. Let allocation $a \in \mathcal{A} \setminus \mathcal{M}(\bar{Z}(t))$ be any non-maximum-pressure allocation, and allocation $a^* \in \bar{\mathcal{M}}(\bar{Z}(t))$ be any maximum pressure support allocation. Then $p(a, \bar{Z}(t)) < p(a^*, \bar{Z}(t))$. On the other hand, we know $\bar{Z}_i(t) > 0$ for any $i \in \mathcal{I}(\bar{Z}(t))$. By the continuity of $\bar{Z}(\cdot)$, there exist $\epsilon > 0$ and $\delta > 0$ such that for each $\tau \in [t-\epsilon, t+\epsilon]$ and $i \in \mathcal{I}(\bar{Z}(t))$,

$$p(a, \bar{Z}(\tau)) + \delta \le p(a^*, \bar{Z}(\tau)) \quad \text{and} \quad \bar{Z}_i(\tau) \ge \delta.$$

Thus, when n is sufficiently large, $p(a, Z(n\tau)) + n\delta/2 \le p(a^*, Z(n\tau))$ and $Z_i(n\tau) \ge n\delta/2$ for each $a^* \in \bar{\mathcal{M}}(\bar{Z}(t))$, each $i \in \mathcal{I}(\bar{Z}(t))$ and each $\tau \in [t-\epsilon, t+\epsilon]$. Choosing $n > 2J/\delta$, then for each $\tau \in [n(t-\epsilon), n(t+\epsilon)]$ we have

$$p(a^*, Z(\tau)) - p(a, Z(\tau)) \ge J, \quad \text{for each } a^* \in \bar{\mathcal{M}}(\bar{Z}(t)) \tag{13}$$

$$Z_i(\tau) \ge J \quad \text{for each } i \in \mathcal{I}(\bar{Z}(t)). \tag{14}$$

Condition (14) implies that any allocation $a^* \in \bar{\mathcal{M}}(\bar{Z}(t))$ is a feasible allocation at the beginning of any cycle in $[n(t-\epsilon), n(t+\epsilon)]$ because a^* only processes jobs from buffers in $\mathcal{I}(\bar{Z}(t))$. Following (13) and the definition of the discrete maximum pressure policy, the allocation a will not be selected at the beginning of any cycle during time interval $[n(t-\epsilon), n(t+\epsilon)]$.

Then we claim if the non-maximum-pressure allocation a is not employed at the beginning of a cycle in the interval $[n(t-\epsilon), n(t+\epsilon)]$, then it will not be employed at the first $L-U$ time units in that cycle. Let t_0 be the beginning of any cycle in $[n(t-\epsilon), n(t+\epsilon)]$, and suppose allocation $\hat{a}$ is employed at time t_0. From the previous argument, $\hat{a}$ must be a maximum pressure allocation. That is, $\hat{a} \in \mathcal{M}(\bar{Z}(t))$. From Lemma 1, the corresponding support allocation must be maximum pressure support allocation, or $\bar{\hat{a}}(\bar{Z}(t)) \in \bar{\mathcal{M}}(\bar{Z}(t))$. Then $i(j) \in \mathcal{I}(\bar{Z}(t))$ for any $j \in \bar{\hat{a}}(\bar{Z}(t))$ by the definition of $\mathcal{I}(\bar{Z}(t))$. (For an allocation a, we adopt the convention that $j \in a$ if and only if $a_j > 0$.) Consequently, any activity $j \in \bar{\hat{a}}(\bar{Z}(t))$ will be employed during $[t_0, t_0 + L - U]$ because $j \in \hat{a}$ and $Z_{i(j)}(\tau) > J$ for $\tau \in [n(t-\epsilon), n(t+\epsilon)]$. If a is employed at any time during $[t_0, t_0 + L - U]$, then $a \ge \bar{\hat{a}}(\bar{Z}(t))$. And hence $p(a, \bar{Z}(t)) \ge p(\bar{\hat{a}}(\bar{Z}(t)), \bar{Z}(t)) = p(\hat{a}, \bar{Z}(t))$, which implies a is a maximum pressure allocation given $\bar{Z}(t)$. We have a contradiction with the fact that a is not a maximum pressure allocation. Therefore, only the allocations in $\mathcal{M}$ can be deployed in the first $(L-U)$ time period of any cycle in the time interval $[n(t-\epsilon), n(t+\epsilon)]$. That is,

$$\sum_{a \in \mathcal{M}} T_a(t_0 + L) - T_a(t_0) \ge (L - U). \tag{15}$$

There are more than $(2n\epsilon - 2L)/L$ whole cycles during the interval $[n(t-\epsilon), n(t+\epsilon)]$. This implies

$$\sum_{a \in \mathcal{M}} T_a(n(t+\epsilon)) - T_a(n(t-\epsilon)) \ge ((2n\epsilon - 2L)/L)(L - U). \tag{16}$$

So $\sum_{a \in \mathcal{M}} \bar{T}_a(t+\epsilon) - \bar{T}_a(t-\epsilon) \ge 2\epsilon(L-U)/L$, and hence $\sum_{a \in \mathcal{M}} \dot{\bar{T}}_a(t) \ge (L-U)/L$. ∎

Lemma 4. *Under a discrete maximum pressure policy with length L,*

$$\sum_{a\in\mathcal{A}} \dot{\bar{T}}_a(t)p(a,\bar{Z}(t)) \geq \rho_0 p(a',\bar{Z}(t)) \text{ for all } a' \in \mathcal{A}, \tag{17}$$

where, as before, $\rho_0 = (L-U)/L$.

Proof.

$$\begin{aligned}
\sum_{a\in\mathcal{A}} \dot{\bar{T}}_a(t)p(a,\bar{Z}(t)) &\geq \sum_{a\in\mathcal{M}(\bar{Z}(t))} \dot{\bar{T}}_a(t)p(a,\bar{Z}(t)) &(18)\\
&\geq \sum_{a\in\mathcal{M}(\bar{Z}(t))} \dot{\bar{T}}_a(t)p(a',\bar{Z}(t)) &(19)\\
&\geq \rho_0 p(a',\bar{Z}(t)). &(20)
\end{aligned}$$

■

For any allocation $a \in \mathcal{A}$, we define the *potential net flow rate* out of buffer $i \in \mathcal{I}$ under the allocation as

$$v_i(a) = \sum_j R_{ij}a_j - \lambda_i.$$

Let $v_i(t) = -\dot{\bar{Z}}_i(t)$ be the *actual net flow rate* out of buffer $i \in \mathcal{I}$ at time t. It follows that

$$v_i(t) = \sum_j R_{ij}\dot{\bar{T}}_j(t) - \lambda_i.$$

Lemma 5. *Under a discrete maximum pressure policy with length L,*

$$v(t)\cdot\bar{Z}(t) \geq v(\rho_0 a)\cdot\bar{Z}(t) \text{ for all } a \in \mathcal{A}. \tag{21}$$

Proof. First, notice that

$$\begin{aligned}
v(t)\cdot\bar{Z}(t) &= \sum_i\sum_j R_{ij}\dot{\bar{T}}_j(t)\bar{Z}_i(t) - \lambda\cdot\bar{Z}(t)\\
&= \sum_{a'\in\mathcal{A}} \dot{\bar{T}}_a(t)\sum_i\sum_j R_{ij}a'_j\bar{Z}_i(t) - \lambda\cdot\bar{Z}(t)\\
&= \sum_{a'\in\mathcal{A}} \dot{\bar{T}}_a(t)p(a',\bar{Z}(t)) - \lambda\cdot\bar{Z}(t),
\end{aligned}$$

and

$$\begin{aligned}
v(a)\cdot\bar{Z}(t) &= \sum_i\sum_j R_{ij}a_j\bar{Z}_i(t) - \lambda\cdot\bar{Z}(t)\\
&= p(a,\bar{Z}(t)) - \lambda\cdot\bar{Z}(t).
\end{aligned}$$

It follows that

$$\begin{aligned} v(t)\bar{Z}(t) - v(\rho_0 a)\bar{Z}(t) &= \sum_{a' \in \mathcal{A}} \dot{\bar{T}}_a(t) p(a', \bar{Z}(t)) - p(\rho_0 a, \bar{Z}(t)) && (22) \\ &= \sum_{a' \in \mathcal{A}} \dot{\bar{T}}_a p(a', \bar{Z}(t)) - \rho_0 p(a, \bar{Z}(t)) && (23) \\ &\geq 0 && (24) \end{aligned}$$

■

Proof of Theorem 3. Following Theorem 3 of Dai and Lin (2003), it is enough to prove that for each fluid limit $(\bar{Z}, \bar{T})$, $\bar{Z}(t) = 0$ for $t \geq 0$.

Let (ρ, x, π) be a solution to (8)-(11), and $(\bar{Z}, \bar{T})$ be a fluid limit. Then $\sum_{a \in \mathcal{A} \setminus \{0\}} (\pi_a / \rho) = 1$, and

$$\sum_{a \in \mathcal{A} \setminus \{0\}} (\pi_a / \rho) v(\rho a) = \sum_{a \in \mathcal{A} \setminus \{0\}} Rx - \lambda = 0.$$

Because $\rho_0 \geq \rho$, $v(t) \cdot \bar{Z}(t) \geq v(\rho_0 a) \cdot \bar{Z}(t) \geq v(\rho a) \cdot \bar{Z}(t)$ for all $a \in \mathcal{A}$. Then

$$v(t) \cdot \bar{Z}(t) \geq \sum_{a \in \mathcal{A} \setminus \{0\}} (\pi_a / \rho) v(\rho a) \cdot \bar{Z}(t) = 0.$$

Consider the following quadratic Lyapunov function:

$$f(t) = \sum_i \bar{Z}_i^2(t). \tag{25}$$

Assume that t is a time at which $(\bar{Z}, \bar{T})$ is differential and $f(t) > 0$.

$$\dot{f}(t) = 2\dot{\bar{Z}}(t) \cdot \bar{Z}(t) = -2v(t) \cdot \bar{Z}(t) \leq 0. \tag{26}$$

Since $\bar{Z}(0) = 0$, it follows that $\bar{Z}(t) = 0$ for all $t \geq 0$. ■

6 Simulation Results

There are many variables that affect the dynamic deployment scenario and the impact can only be studied by using a simulation model.

Arrival pattern: We generate arrival to the terminal using an arrival pattern representative of a typical port (cf. Figure 14). There are 48 vessels scheduled to call at the terminal on a weekly basis and the figure shows the expected time of call during a week for each vessel. There are several other vessels shown below the chart, and these correspond to vessels which do not call weekly at the terminal, but may call every 10 days.

Design of experiment: The following describes the events regarding vessel arrival times and the subsequent planning and execution that is done prior to berthing a vessel. The experimental setup for the simulations also follows these events.

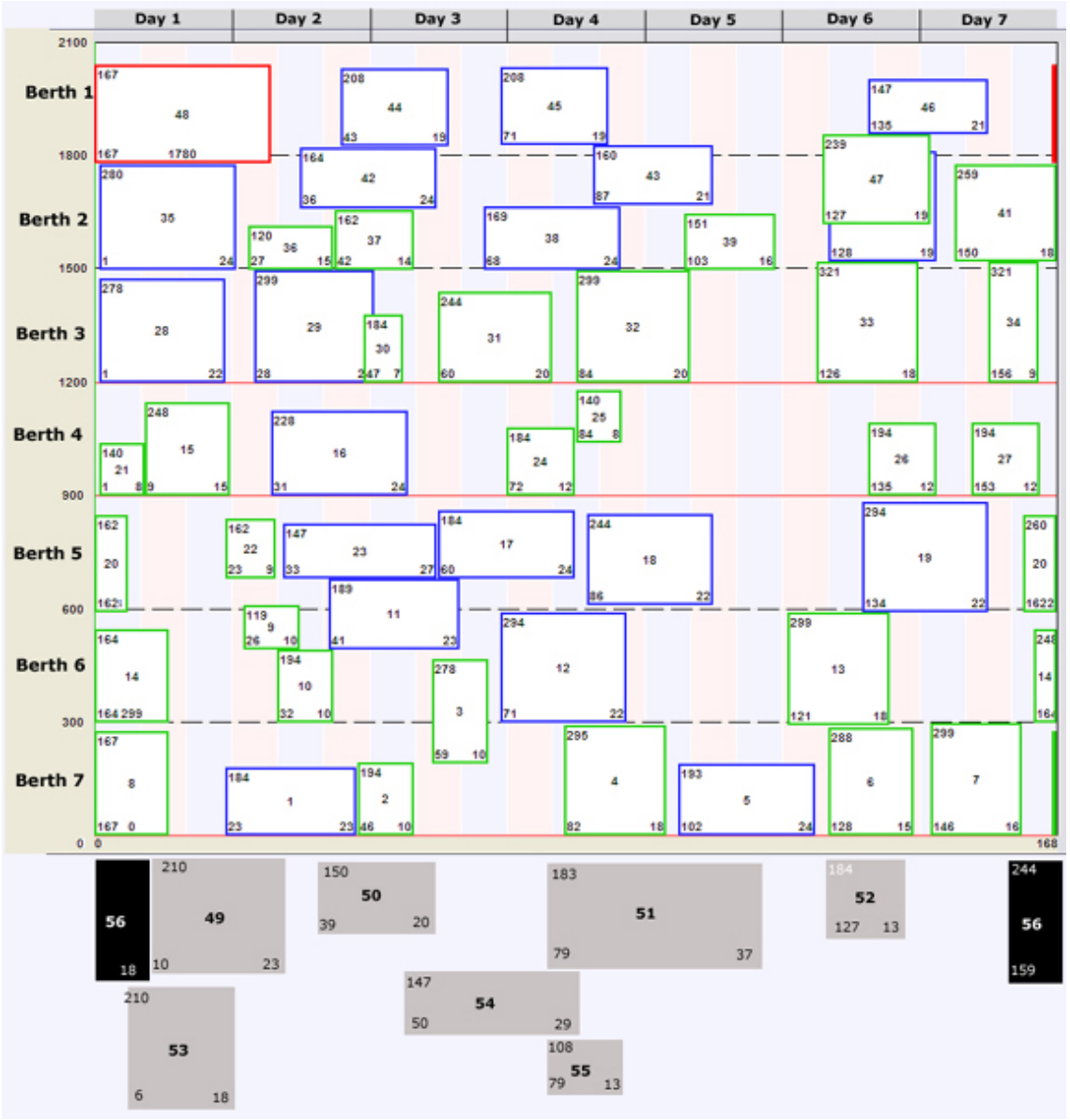

Figure 14: Template for the simulation

- The vessels follow a cyclic arrival with a period of 7 (regular calls) or 10 (irregular calls) days.
- The arrival time according to the template is the *Initial Berth Time Requested* (IBTR) by the vessels. The actual arrival time is stochastically distributed with the *Initial Berth Time Requested* as the mean arrival time and a large variance.
- In practice, the vessel captain is allowed to update the arrival time twice before actually reaching the port. Firstly, a few days before arrival (in simulation, we assume 48 hours), the captain updates the port with an *Updated Berth Time Requested* (UBTR). This is a more accurate estimate of the actual arrival time and in the simulation, we obtain the arrival time from a stochastic distribution with the *Updated Berth Time Requested* as the mean and a smaller variance. Furthermore, the exact time at which the different vessels update this information will be different. In the simulation, we randomly generate the exact time that every vessel reports the *Updated Berth Time Requested*.
- At another time fence, very close to the current time, usually in the order of 1-2 shifts, the *exact arrival time* is reported. In the case of our simulation, we assume that vessels will update their exact arrival time when the *Updated Berth Time Requested* is around 16 hours away from current time. As before, we randomly generate the time when each vessel reports the exact arrival time.
- The experiments are performed for 15 weeks and the results are reported only for the first 14 weeks to eliminate boundary effects.
- The vessels are randomly partitioned into seven priority classes, and the preferred berth location is generated according to the berthing location within the template.
- Typically, the berth locations are fixed (or minimum changes made) for the vessels set to arrive in the nearest shifts. This is done to prevent last minute resource re-allocation, which may lead to inefficient resource management. Based on this, we define a notion of *steadiness*. Steadiness is measured based on changes in the plan and the execution within the next 2 shifts (16 hrs) from the current time. In the simulation, to ensure 100% steadiness, we will also impose upper and lower bound constraints on the berthing location of the vessels.
- Since the exact arrival time of the vessel is not known, and due to the fact that a vessel may update the exact arrival information at any time, we measure steadiness based only on the changes in vessel location. In the simulation, we allow changes in the planned and realized berthing times. But we ensure that the variation is kept to a minimum by restricting the earliest and latest update time.

The results from Section 5 show that the port operator should definitely load the vessels in the terminal so that $\rho < 1$ in the LP defined by (8) - (11). In this case, to ensure rate stability,

the port should set a planning window L and apply the discrete maximum pressure policy to redistribute the load in the terminal every L period. This, however, is costly since it requires the port to clear/delay all vessels prior to the beginning of each planning cycle. In reality, however, the load in the terminal should be selected in such a way that no vessel will experience substantial delays while waiting for berthing, i.e., the heavy traffic situation should ideally be avoided. In a moderate load scenario, by looking ahead of the scheduling window, we hope to be able to plan for future load in an adequate manner so that current decisions will not produce undesirable effects in the future. We will avoid the costly exercise of rebalancing the load in the terminal by using the discrete maximum pressure policy.

We specifically run the following test scenarios and compare the performance of the dynamic berth allocation planning model in a rolling horizon setting. For comparison, we generate the *Initial Berth Time Requested* and the *Updated Berth Time Requested* beforehand and run the following experiments. There are some managerial issues that are interesting to study in the berth planning problem. The developed simulation package allows us to study the impact of these on the overall performance of the system. For instance, we test the effect of the *planning time window*, i.e., how far to look ahead in the scheduling window at each period. There is a trade-off between myopic planning (small time window) and inaccuracy in data (long time window). We also test the trade-off between the objective of attaining high BOA for vessels, and the desire to minimize replanning of activities.

Experiment A: Forecast is accurate.

In this experiment, the actual arrival time of vessels follows the data as given, i.e., the IBTR corresponds to the actual arrival time. This test represents the current arrival scenario and provides a bench mark for the performance of the proposed system. Ideally, if we use the 48 vessels scenario in Figure 14, and if the arrival time is accurately reflected, we expect all vessels to attain BOA status and to be berthed at their preferred location. Table 3 shows the results from a run of the simulation package, with a planning time window of 48 hours. Furthermore, berthing plan for vessels due to arrive over the next 16 hours (where arrival time is known exactly) is frozen.

96% out of 672 vessels served in the 14 week simulation environment received BOA treatment, and close to 85% were berthed at their designated preferred berths.

The above results indicate that the proposed dynamic berth allocation planning model performs extremely well in this environment.

Experiment B: Template-based data with uncertainty in arrival time.

In this setting, we assume that the actual arrival time has variance 30 and 10 with respect to IBTR and UBTR respectively. We examine the impact of the planning window on the performance of the algorithm. Again, the berthing plan for vessels due to arrive over the next 16 hours (where arrival time is known exactly) in each planning time window will be frozen.

We summarize the simulation results in Table 4. Interestingly, the results show the advantage

Class	# of vessels served	% BOA	Average Delays (Hr)	% Allocated Preferred Berth
0	126	90	0.38	79
1	56	75	0.50	50
2	56	100	0.00	100
3	84	100	0.00	82
4	70	100	0.00	97
5	210	100	0.00	87
6	56	100	0.00	96
7	14	100	0.00	93

Table 3: Simulation results obtained for Experiment A

of incorporating the longer planning time window of 72 hours over the shorter planning time window of 16 hours. Advance information on vessel arrival time, despite the uncertainty in the information, is still beneficial for the terminal in terms of berth planning. This is mainly because the plan for the first 16 hours will be frozen, and hence advance arrival information beyond the first 16 hours helps the terminal to obtain better berthing plan.

Windows	% BOA	Average Delays (hr)	% Allocated Preferred Berth
16	88	0.81	70
24	89	0.53	75
48	90	0.72	78
72	93	0.33	76

Table 4: Simulation results obtained for Experiment B

Experiment C: Effect of freezing berthing plan. In this setting, we compare the performance of the model under the same setting as experiment B, except that the port will replan and possibly change berthing positions every time there is an update on vessel arrival data, i.e., we do not freeze the berthing plan in every planning window.

Windows	% BOA	Average Delay (Hr)	% Allocated Preferred Berth	% Steadiness
16	94	0.25	78	23
24	94	0.23	75	29
48	94	0.26	73	31
72	93	0.29	75	38

Table 5: Simulation results obtained for Experiment C

In this case, the ability to replan allows the model to achieve better performance in terms of BOA and preferred berth allocation statistics. Furthermore, the advantage of a longer planning

time window is no longer apparent. By planning using only the accurate arrival information (within first 16 hours), and with the capability to update the berthing plan every hour, the terminal can achieve an equivalent performance to the case when longer planning time windows are used. However, this comes at a price of frequent berthing plan revisions, which is undesirable. In the table, steadiness is a measure of the number of vessels that remain at the same location within the 16 hour period. In Experiment B, steadiness is 100%.

Experiment D: Effect of differentiated priorities. In this experiment, we study the effect of differentiated priorities on the performance of the system, using a planning time window of 48 hours. We compare average BOA and preferred berth allocation between the current system (with differentiated priorities), and the case when all vessels are accorded high or low priority status. Note that in the equal-high scenario, the overriding concern is to berth all vessels on arrival. In the equal-low scenario, the focus is on maximizing the benefits of preferred berth allocation. As in experiment C, we do not freeze the berthing plan in this case. Table 6 summarizes the results observed.

Priority	% BOA	Average Delay (Hr)	% Allocated Preferred Berth	% Steadiness
Differentiated	94	0.26	73	31
Equal-High	97	0.14	67	30
Equal-Low	90	0.58	86	46

Table 6: Simulation results obtained for Experiment D

The results indicate a good balance between the desire to obtain high BOA with a high percentage allocated to preferred berth.

Experiment E: Irregular arrivals. Here, we repeat the experiment set up in A, but with a loading profile according to Figure 14, including the vessels with a cycle time of 10 days. Table 7 shows the results obtained with the added classes of vessels. Note that in this case, berth utilization of the terminal increases at a price: while class 6 and 7 vessels continue to enjoy BOA berthing, the service experienced by class 0-5 vessels suffers. This is especially apparent for class 0-2 vessels, where the service standard drops by close to 20%.

Experiment F: Irregular arrivals and uncertainty in arrival time. Here, we repeat the experiment set up in B, but with an irregular vessel arrival pattern along with uncertainty in arrival time. Table 8 shows the results obtained, which conform to the results in experiment B.

7 Conclusion

In this paper, we proposed an efficient heuristic to construct a berthing plan for vessels in a dynamic berth allocation planning problem. Our method builds on a well-known encoding scheme used for VLSI design and rectangle packing problems. We show that the same approach

Class	# of vessels served	% BOA	Average Delays (Hr)	% Allocated Preferred Berth
0	126	86	2.20	83
1	56	61	1.57	52
2	56	77	1.88	88
3	104	98	0.13	86
4	100	98	0.04	88
5	220	98	0.04	82
6	56	100	0.00	91
7	34	100	0.00	68

Table 7: Simulation results obtained for Experiment E

Windows	% BOA	Average Delays (Hr)	% Allocated Preferred Berth
16	83	1.47	73
24	86	0.91	80
48	88	0.78	77
72	86	0.93	75

Table 8: Simulation results obtained for Experiment F

can be used effectively for the berth allocation planning problem, with the introduction of virtual wharf marks. Extensive simulation results based on a set of vessel arrival data shows the promise of this approach. For a moderate load scenario, this approach is able to allocate space to over 90% of vessels upon arrival, with more than 80% of them being assigned to the preferred berthing location. Our simulation results also show that the performance varies according to the various policy parameters adopted by the terminal operator. For instance, frequent replanning and revisions to the berthing plan, with the resulting undesirable impact on terminal resource deployment, yields close to 1%-6% improvement in BOA statistics.

For a heavy load scenario, we need to ensure that the throughput of the terminal will not be adversely affected by the design of the berthing algorithm. We use the discrete maximum pressure policy to redistribute load at every L interval, where L is suitably selected based on the solution to an associated LP. In this way, throughput at a contain terminal is maximized.

To the best of our knowledge, this is arguably the first paper to examine issues in the dynamic berth allocation planning problem. There are, however, several limitations to our approach. Future work will try to address some of these limitations. For instance, we have ignored several important berthing constraints while allocating space to vessels. Some of the constraints ignored include the availability and scheduling of cranes, and gaps between berthing vessels. We need to address these concerns in the static berth allocation planning problem so that the model can be applied in practice. Furthermore, we have also assumed historical crane intensity rates while working out the port stay time. In practice, the port stay time of each vessel can be erratic. We

need to enrich the berthing algorithm to handle situations with uncertain port stay times.

Acknowledgments:
Research supported in part by National Science Foundation grant DMI-0300599, National University of Singapore Academic Research Grant R-314-000-030-112 and R-314-000-048-112, TLI-AP, a partnership between National University of Singapore and Georgia Institute of Technology, and by the Singapore-MIT Alliance Program in High Performance Computation for Engineered Systems.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin (1993). Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Englewood Clffs, NJ.

[2] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting (2000). Scheduling in a queueing system with asynchronously varying service rates. Preprint.

[3] F. Barahona and R. Anbil (2000). "The volume algorithm: producing primal solutions using a subgradient method," *Mathematical Programming*, 87, pp 385-399.

[4] G. G. Brown, S. Lawphongpanich, and K. P. Thurman (1994). "Optimizing ship berthing," *Naval Research Logistics*, 41, 1-15.

[5] C. Y. Chen and T. W. Hsieh (1999). "A timespace network model for the berth allocation problem," Presented at the *19th IFIP TC7 Conference on System Modelling and Optimization.*

[6] L. W. Chen (1998). "Optimisation problem in a container port," M.Sc Research Report, SoC, NUS.

[7] J. T. Chia, H. C. Lau, and Andrew Lim (1999). Ant Colony Optimization for the Ship Berthing Problem, in P.S. Thiagarajan, R. Yap (Eds.): *ASIAN'99*, LNCS 1742, pp. 359 - 370.

[8] J. G. Dai and Wuqin Lin (2003). Maximum Pressure Policies in Stochastic Processing Networks. Preprint.

[9] J. G. Dai and B. Prabhakar (2000). "The throughput of data switches with and without speedup," *IEEE INFOCOM*, pages 556–564.

[10] S. P. Fekete and Jorg Schepers (1997). "A new exact algorithm for general orthogonal d-dimensional knapsack problems," *Lecture Notes in Computer Science Volume 1284*, Number 267, pages 144–156.

[11] A. V. Fishkin, Klaus Jansen and Lorant Porkolab (2001). "On minimizing average weighted completion time: A PTAS for scheduling general multiprocessor tasks," in *Proceedings 13th International Symposium on Fundamentals of Computation Theory (FCT'01)*, Rusins Freivalds (Ed.), Riga, LNCS 2138, Springer Verlag, 495-507.

[12] Y. P. Guan , W. Q. Xiao, Raymond K. Cheung and CL Li (2002). "A multiprocessor task scheduling model for berth allocation: heuristic and worst case analysis," *Operations Research Letters* **30**, 343-350.

[13] J. M. Harrison (2000). "Brownian models of open processing networks: canonical representation of workload," *Annals of Applied Probability*, 10:75–103.

[14] S. Imahori, M. Yagiura, and T. Ibaraki (2003). "Local Search Algorithms for the Rectangle Packing Problem with General Spatial Costs," *Mathematical Programming* Series B, 97, 543-569.

[15] A. Imai, Etsuko Nishimur, and Stratos Papadimitriou (2001). "The dynamic berth allocation problem for a container port," *Transportation Research* Part B, 35, 401-417.

[16] A. Imai, Etsuko Nishimura and Stratos Papadimitriou (2003). "Berth allocation with service priority," *Transportation Research* Part B, 37, 437-457.

[17] K. K. Lai, and K. Shih (1992). "A study of container berth allocation," *Journal of Advanced Transportation*, 26(1), 45-60.

[18] C. L. Li, X. Cai, and C. Y. Lee (1998). "Scheduling with multiple-job-on-one-processor pattern," *IIE Transactions*.

[19] Andrew Lim (1998). "On the ship berthing problem," *Operations Research Letters*, Vol 22, Number 2-3, 105-110.

[20] C. Lin (1999). "A more efficient sequence pair perturbation scheme," IEEE/ProRISC99, ISBN 90-73461-18-9, pp. 295-300.

[21] S. N. Loh (1996), "The Quadratic assignment problem and its generalization to the berth allocation problem," Honours Years Project Report, DISCS, NUS.

[22] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand (1999). "Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, 47:1260–1267.

[23] K.C. Moon (2000). "A Mathematical Model and a Heuristic Algorithm for Berth Planning," Unpublished thesis. Pusan National University.

[24] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani (1996), VLSI module placement based on rectangle packing by the sequence pair, *IEEE Transanctions on Computer Aided Design of Integrated Circuits and Systems*, 15-12, 1518-1524.

[25] H. Murata, K. Fujiyoshi, and M. Kaneko (1998). "VLSI/PCB Placement with Obstacles Based on Sequence Pair," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 17, 60-68.

[26] A. L. Stolyar (2001). MaxWeight scheduling in a generalized switch: State space collapse and equivalent workload minimization under complete resource pooling. Preprint.

[27] X. Tang, R. Tian, and D.F. Wong (2000). "Fast evaluation of sequence pair in block placement by longest common subsequence computation," in *Proc. Des. Autom. Test Eur. Conf.*, 106-111.

[28] L. Tassiulas (1995). "Adaptive back-pressure congestion control based on local information," *IEEE Transactions on Automatic Control*, 40:236–250.

[29] L. Tassiulas and P. B. Bhattacharya (2000). "Allocation of interdependent resources for maximal throughput," *Stochastic Models*, 16:27–48.

[30] L. Tassiulas and A. Ephremides (1992). "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, 37(12):1936–1948.

[31] L. Tassiulas and A. Ephremides (1993). "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Transactions on Information Theory*, 39:466–478.

Appendix I

Consider the berthing example as shown in Section 1. To maximize the space utilization, one would like to berth vessels along the boundaries. For example, when a class 1 vessel comes in, one would like to moor it in sections 1-3 or in sections 5-7, so that there are 4 consecutive sections left for the processing of another vessel of either class 1 or 2. Moreover, if a class 3 vessel is processed at a given time, then no other vessel can be processed. So we can assume all class 3 vessels are processed along the lower boundary.

Since the length of the scheduling window is 32 hours, at time 0, the optimal berth utilization algorithm (denoted by OUP policy) is to always assign the sections to class 1 or class 2 vessels or both. This is obviously true when there is no vessel from class 3 and there is no class 3 arrival in the scheduling window, since all vessels get processed upon their arrival. The vessels are thus packed, according to OUP, as shown in Figure 3.

Note that the first class 3 arrival occurs at time 640 and the scheduling window is 32, so time 608 is the first time a class 3 arrival is observed in the scheduling window.

Claim 1. *Even if there are vessels in buffer 3 or some class 3 vessels are arriving in the scheduling window, only class 1 and 2 vessels are going to be packed by algorithm OUP.*

Proof. We show this by contradiction. Suppose time t_0 is the beginning of the first scheduling window when the vessel in class 3 is selected to be employed at some time t_1. Let t_2 be the completion time of the last class 1 or 2 vessel before or at time t_1.

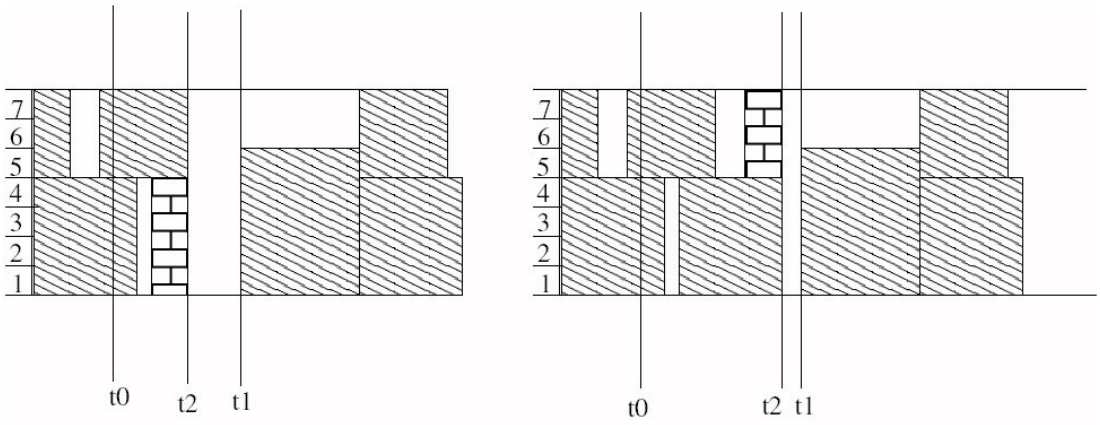

Figure 15: OUP for an example when class 3 vessel is selected

Denote $U^2(\tau_1, \tau_2)$ to be the unutilized capacity between time τ_1 and τ_2 under the new packing policy. We want to show

$$40 = U^1(t_0, t_0 + 32) < U^2(t_0, t_0 + 32), \tag{27}$$

which implies the new packing policy has smaller utilization than the first packing policy in the scheduling window starting from time t_0. This will contradict OUP.

Note that immediately before time t_2, if a class 1 vessel is the last to be processed before t_1, there will be at least 28 units of unused space. If a class 2 vessel is the last to be processed before t_1, there will be at least 27 units of unused space. If $t_1 \leq t_0 + 25$, then a class 3 vessel will finish at least 7 units of its processing time within the scheduling window. Since this leads to unused space of 14 unit-space, we have

$$U^2(t_0, t_0 + 32) \geq 14 + \min(28, 27) = 41 > U^1(t_0, t_0 + 32).$$

When $t_1 \geq t_0 + 26$, however, at least one class 1 and one class 2 vessel can be processed entirely within the scheduling window, giving rise to additional unused space with the new policy. It can be shown via tedious analysis that the total unused space under the new policy will always be greater than 40, contradicting the assertion that OUP optimizes the berth utilization within the scheduling window. Therefore under the optimal utilization policy with scheduling window length equal to 32 hours, the sections will always be assigned to class 1 and 2 vessels, and class 3 vessels get no chance to be served. ■

Appendix II

Proof of Theorem 1: Let (H^*, V^*) denote the sequence pair corresponding to an optimal packing $\mathcal{P}^* = \{(x_i^*, t_i^*)\}$. Note that $\mathcal{P}^*$ may not be obtainable from (H^*, V^*) using LEFT-DOWN packing. By reducing the value of x_i^*, t_i^* as much as possible, while satisfying the precedence constraints defined from G_S and G_T, we can obtain an equivalent optimal packing (EP) such that for every vessel i, either (i) $x_i^* = x_j^* + l_j$ for some j, or (ii) $x_i^* = \sum_{i=1}^{k-1} L_i$ for some k. The second condition corresponds to the vessel berthing on an edge of the berth. We introduce an appropriate number of virtual wharf-marks such that for every vessel (say, vessel i) without binding constraints in the space graph (i.e., case (1) does not hold, but case (2) holds), we modify the sequence pair as follows:

$$H^* : (\dots, i, \dots) \rightarrow H^{**} : (\dots, i, w_k, \dots), \quad V^* : (\dots, i, \dots) \rightarrow V^{**} : (\dots, w_k, i, \dots)$$

where w_k is a wharf-mark with lower bound constraint $x_{w_k} \geq \sum_{j=1}^{k-1} L_j$.

We next show that the LEFT-DOWN strategy, using the sequence pair (H^{**}, V^{**}), gives rise to a packing where the vessels in the original problem are packed exactly as in $\mathcal{P}^*$. Note that by construction, the ordering of w_k in the sequence pair (H^{**}, V^{**}), with respect to every other vessel in the original problem, is identical to the ordering of vessel i with that vessel in (H^*, V^*). Let (x_i^{**}, t_i^{**}) be the optimal solution corresponding to (H^{**}, V^{**}). For vessel i such that case (ii) holds in the packing in $\mathcal{P}^*$ for vessel i, we have $x_i^{**} = x_{w(k)}^{**} = x_i^*$. Vessel i is now supported by the virtual wharf mark $w(k)$ and hence the optimal packing corresponding to (H^{**}, V^{**}) can be obtained by setting the berthing position and time as small as possible, while satisfying all the lower bound and precedence constraints. ■