Berth Allocation Planning Optimization in Container Terminal

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

August 2003. Revised June 2004

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 release 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.

*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

Containers come into the terminals via ships. The majority of the containers are 20 feet and 40 feet in length. The quay cranes load the containers into prime movers (container trucks). The trucks then move them to the terminal yards. The yard cranes at the terminal yards then unload 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.

The problem studied in this paper is motivated by a berth allocation planning problem faced by a port operator. 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, and historical crane rate). Vessel owners usually request a berthing time (called Berth-Time-Requested or BTR) in the terminal several days in advance, and are allowed to revise the BTR when the vessel is close to calling at the terminal. Note that the terminal operator allocates berth and quay side cranes according to the long term schedules and weekly ETA (Estimated-Time-of-Arrival) provided by the Lines. To minimize disruption and to facilitate planning, the terminal operator normally demands all containers bound for an incoming vessel to be ready in the terminal before the ETA. Similarly, 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 being prime concerns.

We assume that the terminal is divided into several wharfs. Each wharf consists of several berths, which in turn are divided into sections. Each wharf corresponds to a linear stretch of space in the terminal. Figure 1 shows the layout of the 4 terminals in Singapore. Note that vessels can physically be moored across different sections, but they cannot be moored across different wharfs.

Our goal in this study is to design a berthing system to allocate berthing space to the vessels in real time. The objectives are 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. The allocation must be determined dynamically over time. To achieve this objective, we need to solve the following problem:

Static Berth Allocation Problem:
How do we design an efficient berth planning system to allocate berthing space to vessels in each planning epoch?

[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., on time berthing) will normally deteriorate with higher berth utilization, since more vessels will be competing for the use of the terminal.

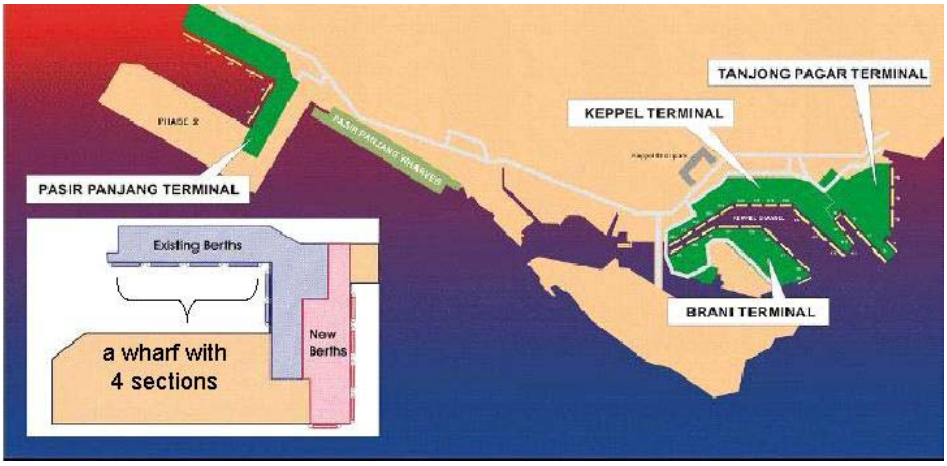

Figure 1: Layout of four terminals in Singapore

The above plan can then be embedded in a rolling horizon framework to allocate berthing space to vessels over time. However, while this may be able to meet the customers' request for high BOA service level, it may result in suboptimal use of the available terminal resources, in particular, low berth utilization. In a heavy load setting, this trade-off is especially crucial, since the impact of inefficient resource utilization translates into a drastic drop in throughput. This leads us to the second issue related to the berth allocation planning problem:

> **Dynamic Berth Allocation Problem**:
> How do we incorporate features into the real time berth planning system to allocate berthing space to vessels, so as to o ensure on time berthing at the preferred locations for most of the vessels calling at the terminal, without adversely affecting the long term throughput?

Example: To see that such features are necessary and important components of a real time vessel berthing system, consider an example with three classes of vessels and a berth with 7 sections.

- Each class 1 vessel occupies 3 sections. The arrival rate is 1 vessel every 16 hours. The vessel turnaround time is 12 hours.
- Each class 2 vessel occupies 4 sections. The arrival rate is also 1 vessel every 16 hours. The vessel turnaround time is 14 hours.
- Each class 3 vessel occupies 5 sections. The arrival rate is 1 vessel every 640 hours. The vessel turnaround time is 16 hours.

Suppose further that the n-th class 1 vessel arrives at time $16n-5$, class 2 at time $16(n-1)$, and class 3 at time $640n$. 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. The optimal packing is given by Figure 2.

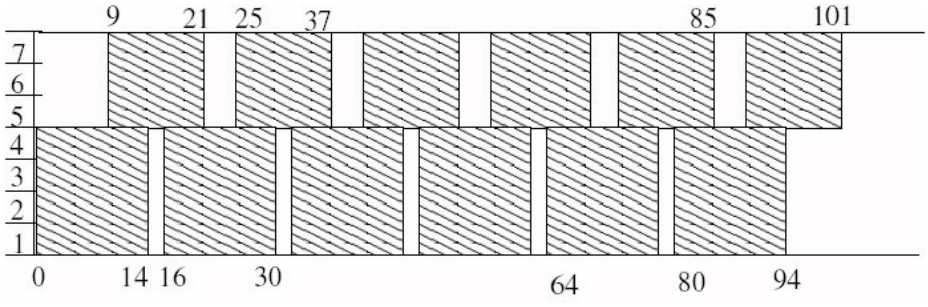

Figure 2: Optimal berthing schedule for class 1 and 2 vessels, when class 3 vessels are ignored

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?

If the focus is on minimizing total delays, then in each planning epoch, since each class 3 vessels will lead to delays on both class 1 and class 2 vessels, it is rational not to service class 3 vessels at all. This arises because class 3 vessels utilize 5 sections in the terminal (16 hours for each class 3 vessel), leaving the other 2 sections unusable. However, the berthing plan obtained by ignoring the class 3 vessels will result in many small unused gaps within the berthing chart (cf. Figure 2), resulting in lower utilization rate and lower throughput in the long haul (since class 3 vessels were ignored). Note that this low throughput scenario can be avoided if we use a berthing policy that ensures class 3 vessels can be moored immediately upon arrival, by reserving space in the terminal for class 3 vessels ahead of arrival. This, however, means that certain vessels from class 1 and 2 will have to be deliberately delayed even if the terminal has space available for them when they call at the terminal. This can be achieved with a priority differentiated berthing policy, or with a complicated penalty cost function to penalize against excessive delay experienced by the vessels. In general, how these can be incorporated in a dynamic berth allocation planning system and how these choices will complicate the static berth allocation planning problem, remains a challenging issue. In this paper, we borrow ideas and incorporate features from throughput optimal policies in the literature to address this issue. In this way, we ensure that the proposed dynamic berth allocation system will not lead to a situation where vessels will be delayed indefinitely in a heavy traffic environment.

In the rest of this paper, we outline an approach to address the issues associated with the static and dynamic 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 side constraints. The objective function chosen involves a delicate trade-off between the waiting time experienced and the deviation from preferred berthing space allocated.

The optimization approach builds on the "sequence-pair" concept (cf. Imahori et. al. (2003)) 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 moored and will leave the terminal some time later. In this case, the space allocated to these vessels cannot be used to moor other arriving vessels. The ability to address these side constraints allows our model to be embedded in a rolling horizon berth allocation planning model.

- In a moderate to 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 throughput rate attained is close to optimum, although the policy may deliberately insert delays on certain vessels, even when the terminal has enough resources to moor the affected vessels on time. In between the periods of load redistribution, however, the static allocation planning problem will be solved in each planning epoch to assign berthing space to the vessels. Hence this approach ensures that both the service performance of the berthing system (number of vessels berthed-on-arrival, preferred berthing space allocated, etc.) and terminal throughput performance are addressed by a single 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 planning horizon, 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, since it has an unintended impact on personnel and resource schedules.

2 Literature Review

There is by now a huge literature on the applications of operations research in container operations (cf. Steenken et al. (2004)). To the best of our knowledge, while there are sporadic papers that address the static berth allocation planning problem, none of the papers in the literature addresses specifically the issue of throughput loss in the dynamic berth planning allocation model. 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 the National University of Singapore (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 it 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. For a follow up paper see Kim et al. (2003). Park et al. (2003) use a Lagrangian relaxation based approach to solve the berth planning problem. Assuming a convex cost structure, they also describe a heuristic procedure for obtaining feasible solutions from the solutions to the relaxed problem. Though relevant, the approach is limited by the choice of the cost function. Further, they do not describe how the methodology can be incorporated in a dynamic framework.

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 makespan 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. Guan et al. (2004) consider two mathematical formulations for berth allocation with the weighted flow time objective and propose a heuristic procedure for solving large size problems. They group vessels into batches and assume that vessels within a batch arrive at the same time. Their approach is tactical in nature and cannot be adapted to schedule vessels in a dynamic setting. Fishkin et al. (2001), using some sophisticated approximation techniques (combination of scaling, LP, and DP), derived the first polynomial time approximation scheme for the multi-processor task scheduling 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 modeled 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

At each planning epoch, given information on the vessels that will be arriving within the chosen planning horizon (or scheduling window), we structured the associated berth planning problem as one of packing rectangles (without overlap) in a semi-infinite strip with general spatial and time 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 moor the vessels on designated locations along the terminal and to minimize the movement and exchange

of containers within the yards and between vessels. These preferred locations for the vessels are normally pre-determined based on the containers storage locations and transshipment cargo volume of the vessels.

The input to the problem are:

- Terminal specifics:
  - W: Number of wharfs in the terminal,
  - M: Number of berths in the terminal,
  - b_l: Length of berth l, $l = 1, \ldots, M$.
- Vessel specifics:
  - N: Number of vessels that have arrived or will arrive within the scheduling window,
  - r_i: Arrival time (ETA) of vessel i,
  - l_i: Length-overall of vessel i,
  - p_i: Length of port stay (turnaround time) upon berthing by vessel i.

The parameters l_i and p_i specifies the dimensions of the N rectangles that need to be packed within the semi-infinite strip, and r_i represents the release time constraint imposed on vessel i. For ease of exposition, we assume that vessels can be moored at any berth available within the terminal. In real terminal berth allocation planning, we normally have to take into account other issues like draft constraints, equipment availability along the berth, tidal information and crane availability etc., to determine the berth where the vessels will be moored. The approach proposed herein can be easily extended to address a host of such side constraints. We also 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 deviate from 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. Nevertheless, the impact of such inaccuracies can be minimized by re-optimization in a rolling horizon framework.

The decision variables to the berth planning problem are:

- x_i: Berthing location for vessel i, measured with respect to the lower left corner of the vessel, represented as rectangle in the time-space plane,
- 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 $c_i(\cdot)$ is a *step function* in x_i defined as,

$$c_i(x_i) = \begin{cases} d_{i,1} & \text{if} \quad x_i \leq b_1, \\ d_{i,2} & \text{if} \quad b_1 < x_i \leq b_1 + b_2, \\ \vdots & \vdots \quad \vdots \\ d_{i,M} & \text{if} \quad \sum_{j=1}^{M-1} b_j < x_i \leq \sum_{j=1}^{M} b_j. \end{cases}$$

Note that $d_{i,l}$ indicates the desirability of mooring 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

$$\sum_{i=1}^{N} \omega_i \times f_i(t_i),$$

where, the function $f_i(t_i)$ is a non decreasing function of t_i. Usually, a vessel is considered "berthed-on-arrival" (BOA) if it can be moored within two hours of arrival (i.e., within r_i and r_i+2) and hence, the delay cost function is typically defined to be $f_i(t_i) = (t_i - r_i - 2)^+$. The value ω_i is a penalty factor attached to vessel k and indicates the "importance" of berthing the vessel k on arrival (i.e., within a two-hour window). The vessels are normally divided into several priority classes with different penalty factors. Note that the approach proposed in this paper works for all non-decreasing cost function f.

The berthing decisions x_i and t_i must satisfy a host of other constraints. Most notably, the choice of x_i and t_i must not lead to any conflict in terminal space allocation. i.e., no two vessels can be assigned to the same stretch of space in the terminal at the same time. Furthermore, since the vessels cannot be moored across different wharfs, x_i is further restricted to belong to disjoint segments of space within the terminal. Coupled with the release time constraints, the static berth allocation planning problem belongs to a class of non-convex optimization problem which is exceedingly difficult to solve. In the rest of this section, we model this problem as rectangle packing problem and use a special encoding scheme to reduce the optimization problem to a search over pairs of permutations.

3.1 Sequence Pair Concept

We first show that every berthing plan can be encoded by a pair of permutations (H, V) of all vessels. Consider the berthing plan of two vessels as shown in Figure 3.

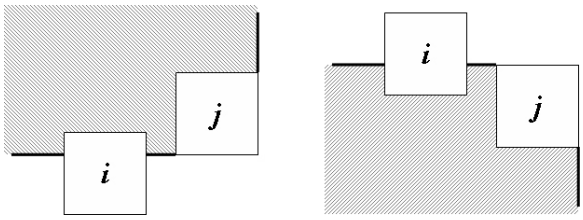

Figure 3: The figure on the left shows the LEFT-UP view of vessel j, whereas the figure on the right shows the LEFT-DOWN view of vessel j in the time-space plane; the views seen by vessel j are shaded regions plus the parts hidden by vessel i.

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

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

Figure 3 shows the LEFT-UP and LEFT-DOWN views of vessel j, with respect to a fixed berthing plan. 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 paper, we write $a <_H b$ (and $a <_V b$) if a is placed on the left of b in H (resp. in V).

- If $a <_H b$ and $a <_V b$, then a does not see b in LEFT-DOWN or LEFT-UP, i.e., vessel b is placed to the right of vessel a in the berthing plan. In other words, vessel b can only be berthed after vessel a leaves the terminal.
- If $a <_H b$ and $b <_V a$, 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 splits 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 4). In the space and time graphs, each vessel is represented by a node and the constraints imposed by the sequence pair are represented by directed arcs.

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 is advantageous since the berthing plan for vessels can actually be constructed greedily in an iterative manner, allowing us to handle a host of side constraints in real terminal berthing operations. Unfortunately, a LEFT-DOWN packing obtained with a fixed sequence pair may not always give rise to the optimal packing, due to the step-wise nature of the berthing space cost.

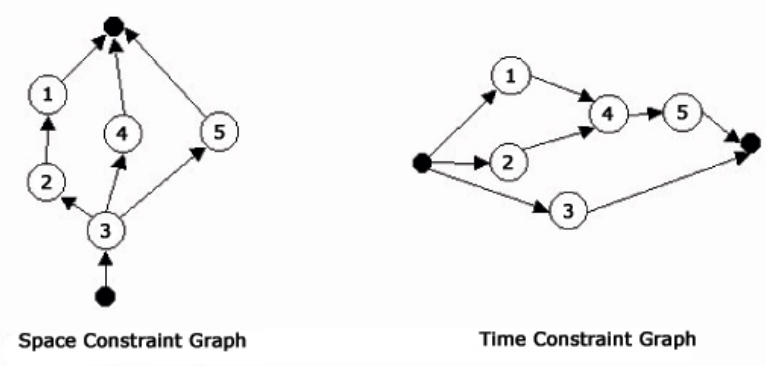

Figure 4: Directed graphs on the space and time variables arising from the sequence pair, H:{1,2,4,5,3} and V:{3,2,1,4,5}

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:

$$\begin{aligned} & \min_{t_i} \sum_{i=1}^{N} \omega_i \times f_i(t_i) \\ s.t. \quad & t_i \geq r_i \ \ \forall \ \ i = 1, \ldots, N \\ & t_i + p_i \leq t_j \ \ \text{if } (i,j) \in G_T. \end{aligned}$$

Since $f_i(t_i) = (t_i - r_i - 2)^+$ 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 dual version of the classical longest path problem in an acyclic digraph, and can be solved easily using a simple dynamic program. The t_i for the source node is set to the smallest possible value (i.e. r_i), and for a non-source node j, if S is the set of predecessors in G_T, then

$$t_j = \max\left(r_j, \max_{i \in S}(t_i + p_i)\right).$$

These values can be computed in a recursive manner (cf. Imahori et. al. (2003) for an efficient dynamic programming based algorithm and Ahuja et. al. (1993) for a general discussion on

the longest path problem). Each t_i is selected to be the *smallest* possible value satisfying the constraints and the solution can be constructed in a recursive 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. Let

$$L_l, U_l, \quad l = 1, \ldots, W,$$

denote the position of the lower and upper end of wharf l in the terminal. The space-cost problem can be formulated as:

$$\begin{aligned}
& \min \sum_{i=1}^{N} c_i(x_i) \\
s.t. \quad & x_i + l_i \leq \sum_{l=1}^{W} U_l y_{il} \quad \forall \quad i = 1, \ldots, N \\
& x_i \geq \sum_{l=1}^{W} L_l y_{il} \quad \forall \quad i = 1, \ldots, N \\
& \sum_{l=1}^{W} y_{il} = 1 \quad \forall \quad i = 1, \ldots, N \\
& x_i + l_i \leq x_j \quad \text{if } (i,j) \in G_S \\
& y_{il} \in \{0, 1\}, \quad l = 1, \ldots, W, \ i = 1, \ldots, N \\
& x_i \geq 0, \quad i = 1, \ldots, N.
\end{aligned}$$

In the special case where $c_i(\cdot)$ is a constant function, the space cost minimization problem can be solved efficiently as in the previous case, using a similar Dynamic Program on G_S. 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.

In fact, berthing the vessel according to the lowest possible position satisfying the constraints may not always give rise to good packing: i.e., LEFT-DOWN packing will not produce a good solution in terms of space cost.

To address this problem, we outline 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 for each vessel is essentially a step function that depends 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.

We introduce a set of *virtual wharf marks* $\mathcal{W}$, with $w \in \mathcal{W}$ representing a vessel with $r_w = 0, l_w = 0, p_w = 0$, with additional constraint of the type

$$x_w \geq \sum_{j=1}^{k(w)-1} b_j$$

for some suitably selected berth $k(w)$. Note that b_j denotes the length of berth j. The constraint imposes a natural lower bound on the berthing location of the virtual wharf mark and indicates the decision to berth the virtual wharf mark at a location in the terminal not lower than berth $k(w)$.

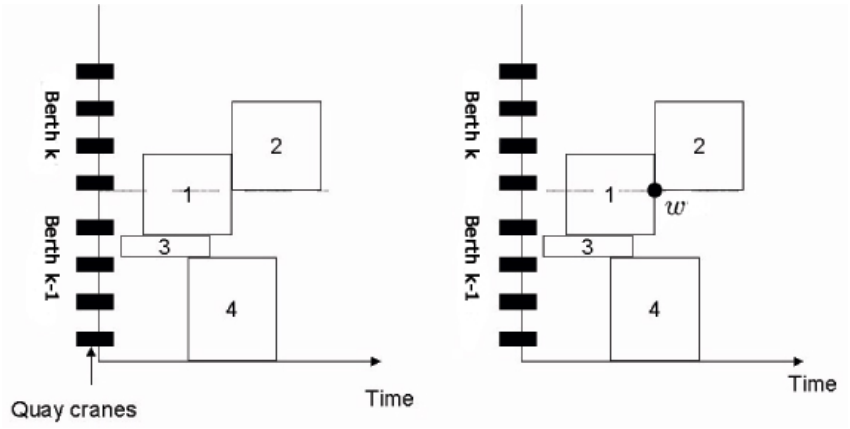

Figure 5: Berthing plan with virtual wharf mark as shown on the right. The sequence pair changes from (1234, 4312) to $(12w34, 431w2)$

Consider the berthing plan on the left side of Figure 5. Due to the space cost function, it may be optimal to allocate vessel 2 to berth k. Unfortunately, a LEFT-DOWN packing will allocate vessel 2 to a location immediately above vessel 4, putting vessel 2 in berth $k-1$. To rectify this situation, we introduce a virtual wharf mark and consider the modified sequence pair $(12w34, 431w2)$. The left-down packing approach will give rise to the desired packing on the right of Figure 5. The introduction of the virtual wharf marks in the sequence pair and the added constraints on the position of the wharf marks allows us to incorporate gaps into the berthing position of the vessels, to prop vessels up to certain berth. Note that the addition of the virtual wharf mark has no impact on the berthing time decisions.

It is not known, apriori, 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^{**}.

Proof: See Appendix I.

In general, finding the optimal number of wharf marks and their associations with the vessels is extremely difficult. It is unnecessary 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, by introducing virtual wharf marks, we can 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.

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.

The critical aspect for getting good solutions is to define an appropriate cooling schedule and a sufficiently large neighborhood that can be explored efficiently. In our empirical experiments, an exponential cooling schedule works well. With regard to the neighborhood, which is vital to ensure extensive exploration, we use the following structures:

(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 6 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. Due to the non-linear space cost, we observe that shifting of vessels between different berths may provide improvement in the solution. This motivated us to define the next class of neighborhood structure.

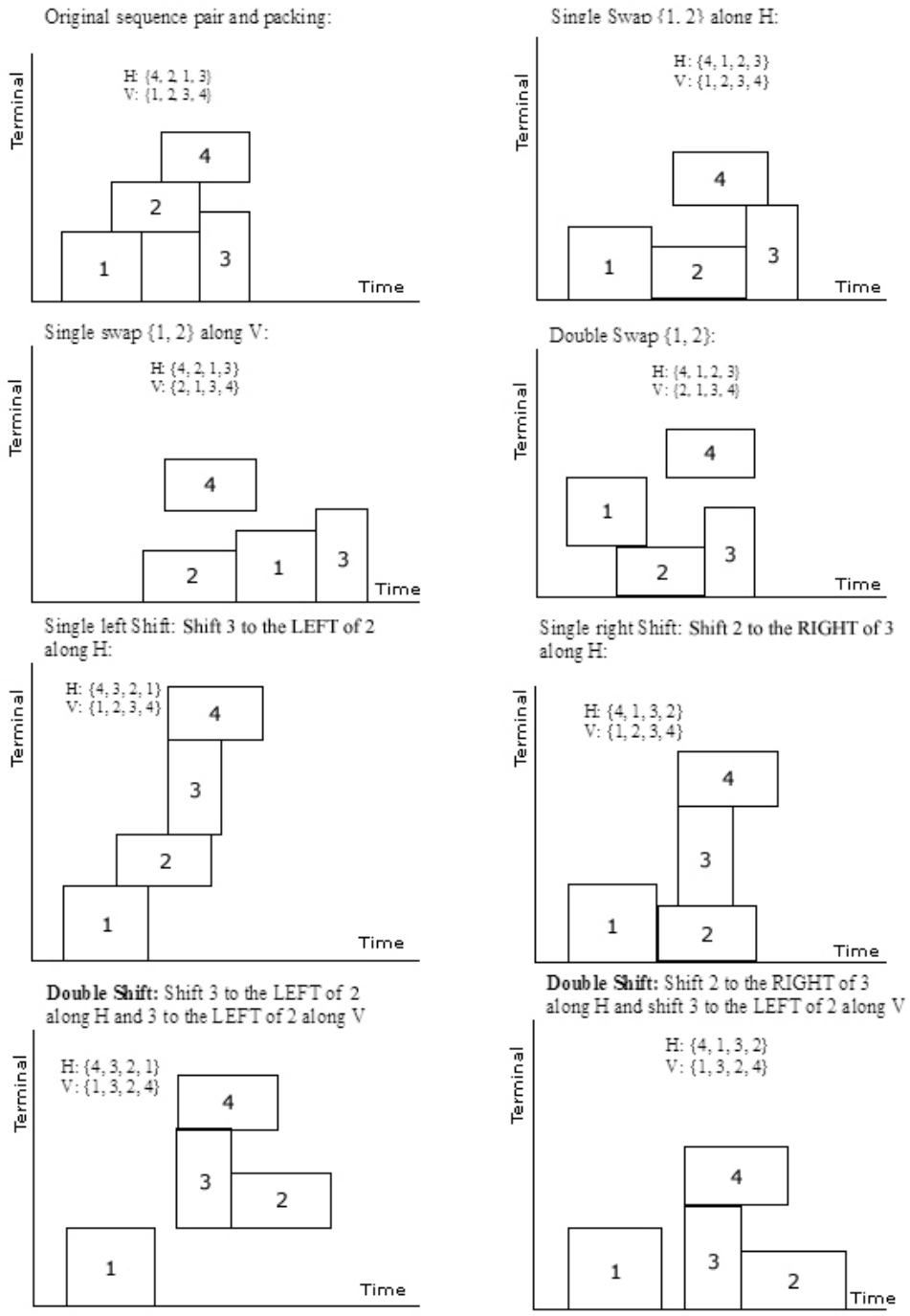

Figure 6: Examples of swap and shift neighborhoods

(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 7 shows the packing and the corresponding sequence pair obtained from a simple greedy neighbor.

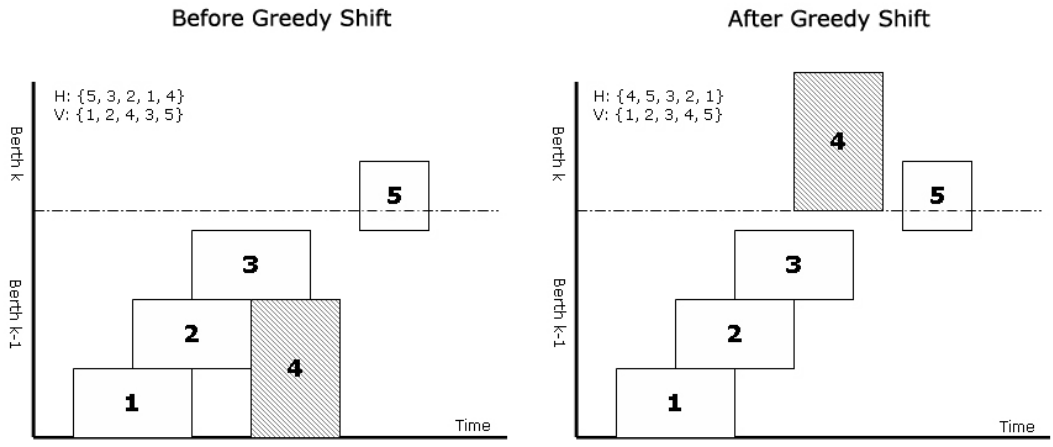

Figure 7: An example for greedy neighbor

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) $x_i = \sum_{l=1}^{k-1} b_l$ 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 moored above berth edge k.

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. Searching the greedy neighborhood is computationally more expensive than simple sequence pair manipulation and hence, for the experiments, the operators (a)-(d) are employed successively while, the operator (e) is used whenever the other operators get stuck in a local optima.

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 Rolling-Horizon Berth Allocation Planning

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 moored in the terminal? Note that the space allocated to these vessels cannot be used to moor other vessels.

We model these constraints as "forbidden zones" along the left edge of the time space 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 8 shows a typical scenario.

Along space, if a vessel does not have binding arcs in the space graph G_S, 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

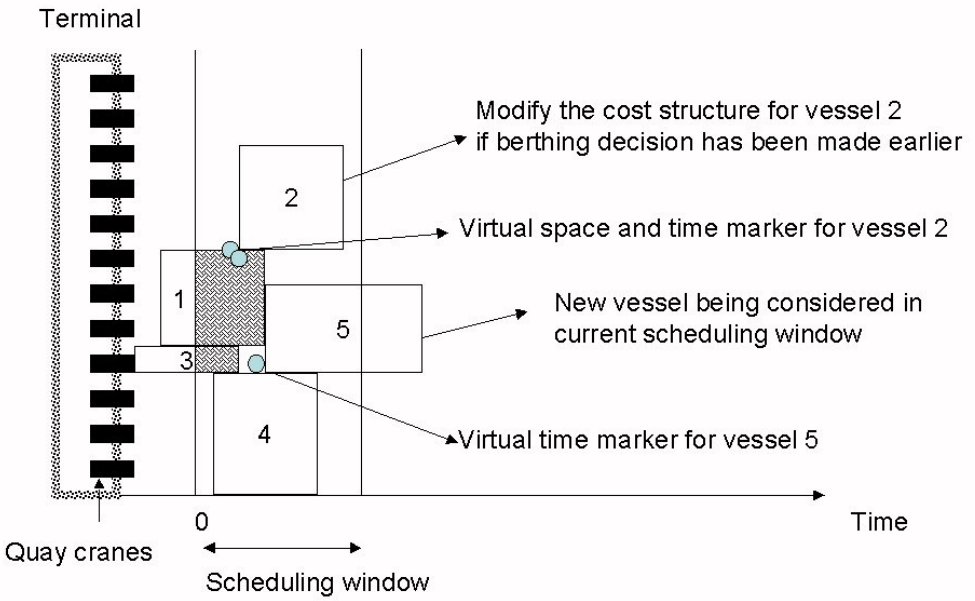

Figure 8: Handling pre-placed vessels

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 *steadiness* (i.e., minimize frequent revisions) in the berth plan, 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 moored at the original berthing location and time.

4 Throughput Consideration

The previous sections outlined a method to allocate berthing space to vessels in a rolling horizon framework. However, it does not preclude the possibility that certain classes of vessels will be delayed for an excessive period, and is thus lost from the system. 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, to ensure that the throughput of the terminal will be maximized. While the concerns with throughput can also be addressed using a more complicated cost function $f_j(t)$ to penalize excessive delays, finding the appropriate choice of the penalty function proves to be tricky in practice, in view of the fact that there are different priority classes of vessels waiting to be berthed at the terminal. The penalty cost for inducing additional delay should depend also on the priority classes of the vessels competing for space at the same time. Our approach here is neater and builds on the insights of the structure of provably throughput optimal policies in sotchastic processing network.

To study the issue of throughput in the berth allocation planning problem, 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[2]. All the vessels of the same class require the same number of sections. The classes of vessels are indexed by $i = 1, 2, \ldots, I$. We denote $Z_i(t)$ 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. $Z(t) = (Z_i(t), i = 1, \ldots, I)$ is called the buffer level at time t. We assume the vessels in the same class are moored in a FCFS fashion. Therefore the berth planning problem is to decide how to assign the sections to classes. We call each possible assignment of the sections to classes as an *allocation* denoted by a. We denote the number of class i vessels moored by allocation a as $n_i(a)$.

4.1 Discrete Maximum Pressure Policies

In this section, we propose a family of policies called *discrete maximum pressure* policies for berth allocation given the processing times are bounded by some constant U. Define the pressure of a berth allocation network with buffer level z under allocation a to be

$$p(a, z) = \sum_i n_i(a)\mu_i z_i,$$

where $\mu_i = 1/m_i$ and m_i is the average length of port stay for class i vessels.

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

[2]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.

Discrete Maximum Pressure Policy for Berth Allocation

- The time horizon is divided into cycles of length L.
- At the beginning of each cycle, an allocation with *maximum pressure* (i.e., $\text{argmax}_a\, p(a, Z(t))$) is selected, and each section is assigned to the vessel class given by the allocation until

  (i) there is no vessel of the corresponding class 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 vessel 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.

Note that if a vessel arrives near the start of any planning cycle, it may not be served even if there is space available in the terminal. This happens when its port stay time is long and hence it will not depart within the current planning cycle. In this case, instead of berthing the vessel, the space is reserved till the next planning cycle for redistribution of load, to accommodate the allocation with the maximum pressure. In a way, the discrete maximum pressure policy is similar to the intuitive approach of clearing the buffers whenever there are many vessels waiting in the system. It uses the notion of $p(a, z)$ to operationalize this concept.

Discrete maximum pressure policies are a variant of the maximum pressure policies (cf. Dai and Lin (2004)). The maximum pressure policies were proven to be throughput optimal for a class of stochastic processing networks. That is, the load can be handled by maximum pressure policies if it can be handled by any other scheduling policy. One can model the berth allocation problem as a stochastic processing network and prove similarly that, by appropriately choosing the length of the planning cycle, the discrete maximum pressure policies can handle the load that can be handled by any other scheduling policy. We omit the detailed proofs in this paper due to space limitation. The argument can be adapted from the proof of throughput optimality of maximum pressure policies given in Dai and Lin (2004). In the following, we point out the difference between the discrete maximum pressure policies and the maximum pressure policies.

- For maximum pressure policies, the allocation is updated every time when there is an arrival or a service completion, while for discrete maximum pressure policies, the allocation is updated only when all vessels are served from some class or the planning cycle is over.
- Maximum pressure policies are in general a preemption allowed policies, but service is not to be interrupted for discrete maximum pressure policies.

4.2 A Numerical Example

To illustrate how the discrete maximum pressure policy works, consider the application of this policy to the problem described in Section 1. We assume $U = 16$, and set $L = 160$. Figure 9 compares the buffer build up in the system when the discrete maximum pressure policy is employed against a naive static berth planning system in a rolling horizon framework. Firstly, it is observed that under the discrete maximum pressure policy, the system is stable with the dynamics repeating every 1920 hours as opposed to the static berth allocation system where the buffer builds up indefinitely. This shows the importance of the dynamic policy in a congested scenario. Secondly, at the beginning of every cycle (i.e. with length L), the system needs to be cleared. This purposely induces delays in the vessels and also increases the size of the static berth planning problem to be solved. The spike in the number of vessels waiting to be serviced at the beginning of each cycle as shown in Figure 9 and the increase in delay experienced by the class 1 vessels (Two vessels are delayed by 5 hours and 1 hour respectively every 160 hours, up from 0 hours when employing the static allocation alone) support this observation.

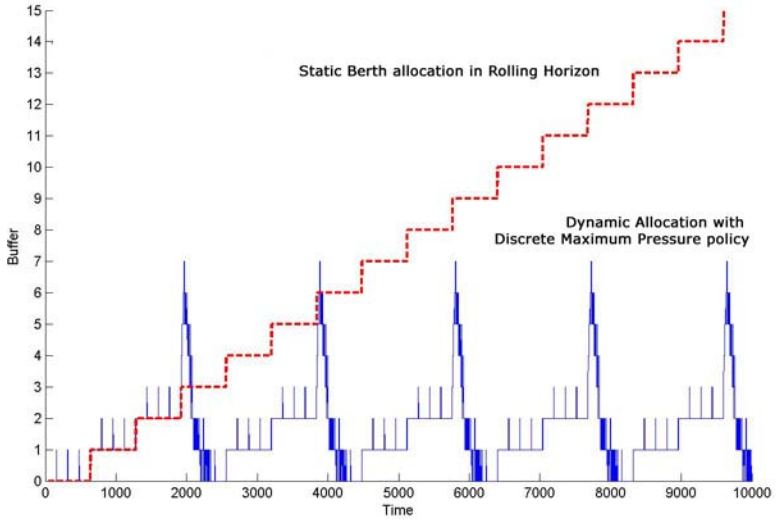

Figure 9: In a heavy load setting, the discrete maximum pressure policy ensures that the system remains stable. However, redistributing the load at the beginning of every cycle increases the number of vessels waiting to be serviced at that time.

In a moderate traffic scenario with a large number of classes of vessels, the buffer build up at the beginning of each cycle may potentially be extremely large, making the static berth planning problem computationally expensive. Further, under moderate load, we would hope that the issues of infinite buffer build up to be eliminated by employing a good static berth planning system and looking further ahead of the scheduling window. Hence, it would be advisable to select the load at a terminal such that no vessel will be substantially delayed. It would be

recommended to employ the discrete maximum pressure policy only in situations where heavy load scenarios cannot be avoided.

5 Computational and Simulation Results

5.1 Computational Results: Static Berth Allocation Planning Model

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.

Let $\mathcal{C}_i$ be the set of allowed positions that vessel i can be moored in the schedule. The set $\mathcal{C}_i$ takes care of all the constraints in berthing positions (eg., no mooring across wharfs) and berthing times (eg., not earlier than arrival) of vessel i. We can immediately construct a lower bound to the problem by solving the following:

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

The variables $\pi(x,t)$ are the dual prices associated with the non-overlapping constraints (for the position (x,t) in space). With fixed $\pi(x,t)$, the inner optimization problem is trivial. We use a version of subgradient algorithm, known as the volume algorithm in the literature (cf. Barahona and Anbil (2000)), to update the dual prices and 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 also discretized to moderate sizes. Note that the discretized version of the problem will still provide us with a lower bound for the static berth planning problem.

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 RU is a measure of traffic load at the terminal and the max RU 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 moored 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.

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 is still 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. However, we believe that the wide gap between SA and LB is primarily due to the weak lower bound. 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). Note that in the light load scenario, we know the optimal solution (0 vessels delayed, 0 vessels placed in inferior locations).

# 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

5.2 Simulation Results: Rolling Horizon Berth Allocation Planning

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 10). 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.

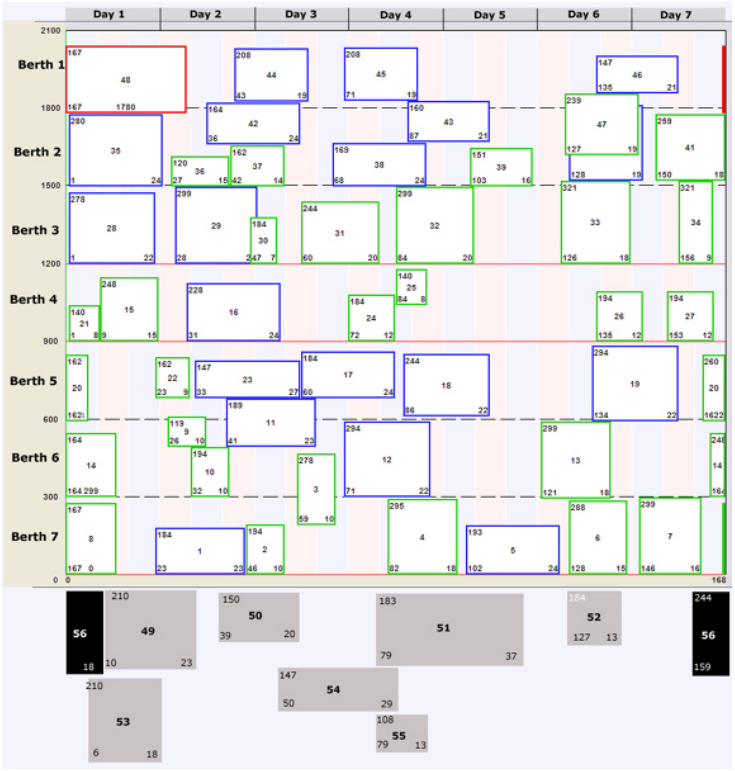

Figure 10: Template for the simulation

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.

- 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 IBTR as the mean arrival time.
- The vessels are randomly partitioned into seven priority classes, and the preferred berth location is generated according to the berthing location within the template in Figure 10.
- Typically, to ensure *steadiness*, 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. Steadiness is measured based on changes in the plan and the execution within the next 2 shifts (16 hrs) from the current time. Due to uncertainty in arrival time, the measure is based only on changes in

vessel location. In the simulation, to ensure 100% steadiness, we will also impose upper and lower bound constraints on the berthing location of the vessels.

The data template as shown in Figure 10 is clearly not heavily loaded since there is enough time in-between vessel arrivals to prevent delay propagation. Hence, in the computational experiments A - F, we avoid the discrete maximum pressure policy and the associated re-balancing exercise and analyze the dynamic performance of the proposed berth planning system. In Experiment G, we present a congested scenario and show the impact of the discrete maximum pressure policy on the throughput of the system.

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 IBTR and the UBTR 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 re-planning 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 10, 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.

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

96% out of 672 vessels served in the 14 week simulation environment received BOA treatment,

and close to 85% were moored 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.

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

Interestingly, the results show the advantage 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.

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 5, 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. 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

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. 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 10, including the vessels with a cycle time of 10 days. Table 7 shows the results obtained with the additional classes of vessels.

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

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.

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

Experiment G: Scheduling under congestion. In this experiment, we study the impact of the scheduling policy on the throughput of the system under a congested scenario. Congestion in a port is typically managed so that no class of vessels suffer throughput loss. However, when certain berths or wharfs are under maintenance or upgrade, the usable length of the terminal gets restricted which results in increased congestion when the arrival pattern remains unaltered.

We use this scenario to create a realistic data set by modifying the terminal length for the arrival pattern in Figure 10. To measure throughput loss, we restrict the maximum buffer for each vessel class to 4 and the vessels arriving after the buffer is full are categorized as *lost sales*[3]

Simulations were performed by varying terminal length settings (from the existing $2100m$ long terminal to $1600m$ - $1700m$) and we compare the lost sales, throughput and delays when scheduling with the naive rolling horizon setup as opposed to employing the discrete maximum pressure policy. The simulation was performed for 3300 hours generating 828 vessel arrivals and the load was redistributed using the discrete maximum pressure policy every 400 hours.

The progression of lost sales with time is shown for one of the simulations (with terminal length of 1600m) in Figure 11 and similar progression was observed for the remaining simulations. On an average over 3 simulations, employing the discrete maximum pressure policy reduced the lost sales by 10 vessels, a near 30% improvement over a naive rolling horizon framework.

The average throughput in the system increased from 95% to 97% while using the discrete maximum pressure policy as shown in Figure 12. However, the redistribution procedure results in increasing the average delay experienced by the vessels from 49.3 hours to 86.3 hours. Note

[3]Computing the actual throughput in a congested scenario proves to be tricky, as the BAP algorithm becomes very slow when there is huge number of vessels that need to be scheduled at each planning epoch. The lost sales provide a surrogate measurement for throughput, and prevent an excessive number of vessels building up at each planning epoch.

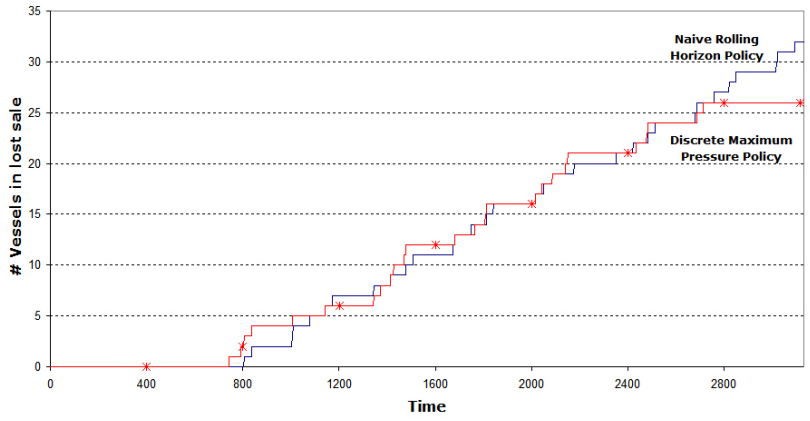

Figure 11: Plot comparing the lost sales while employing the naive rolling horizon policy against a discrete maximum pressure policy.

that * in the Figure 12 indicates that a particular class of vessel was never served in one of the simulations employing the naive rolling horizon framework, and the average delay excludes this class.

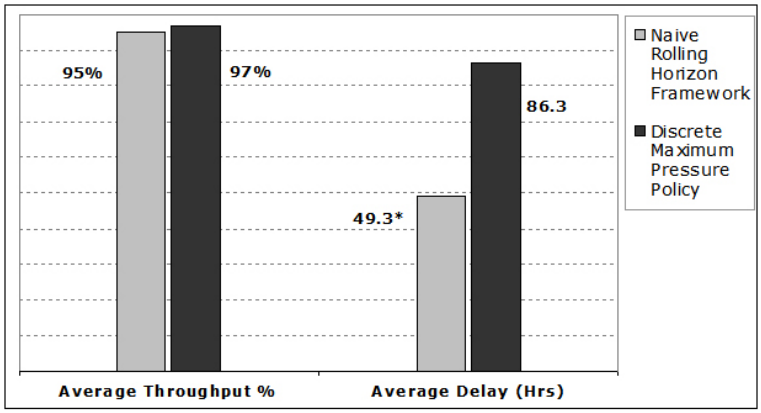

Figure 12: A comparison of average delay and throughput of the berth planning system

Although the redistribution procedure results in increased average delay, the improvement in throughput when employing the discrete maximum pressure policy is evident from the simulation results.

6 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 can be used effectively for the berth allocation planning problem, with the introduction of virtual wharf marks. Extensive simulation results based on a realistic 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 loading profiles at the terminal. Simulation results show that this strategy helps in the heavy traffic scenario to minimize throughput loss due to excessive waiting for certain classes of vessels.

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:
The authors would like to thank 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 (2004). Maximum Pressure Policies in Stochastic Processing Networks. To appear in *Operations Research*.

[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 , Raymond K. Cheung (2004). "The berth allocation problem: models and solution methods," *OR Spectrum* **26**, 75-92.

[13] 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.

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

[15] 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.

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

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

[18] K. H. Kim, and K. C. Moon (2003). "Berth scheduling by simulated annealing," *Transportation Research* Part B, 37, 541560.

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

[20] C. L. Li, X. Cai, and C. Y. Lee (1998). "Scheduling with multiple-job-on-one-processor pattern," *IIE Transactions*. Vol. 30, 433–446.

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

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

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

[24] 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.

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

[26] 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.

[27] 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.

[28] Y. M. Park, K. H. Kim (2003). "A scheduling method for Berth and Quay cranes," *OR Spectrum*, 25, 1-23.

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

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

[31] 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.

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

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

[34] 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.

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

[36] Chia Jim Tong, Hoong Chuin Lau, Andrew Lim (1999). "Ant Colony Optimization for the Ship Berthing Problem," *Asian Comp. Sci. Conf. (ASIAN)*, 359–370.

Appendix I

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 mooring 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. ■