SIMULATION ANALYSIS OF ALGORITHMS FOR CONTAINER STORAGE AND YARD CRANE SCHEDULING AT A CONTAINER TERMINAL

Matthew E. H. Petering*,** and Katta G. Murty*

**Department of Industrial and Operations Engineering, University of Michigan—Ann Arbor*
1205 Beal Avenue, Ann Arbor, Michigan 48109 U.S.A.
(mpeterin, murty}@umich.edu

***The Logistics Institute—Asia Pacific, National University of Singapore*
Block AS6, Level 5, 11 Law Link, Singapore 119260
tlimehp@nus.edu.sg

ABSTRACT

The average quay crane (QC) rate at most marine container ports is currently hovering around 25 moves per hour. In many cases, under current operating conditions, this rate can be improved only marginally by increasing the number of yard trucks (YTs) and yard cranes (YCs) per QC. However, a QC is technically capable of making 40 moves per hour when YTs are always on hand at the quay to deliver/receive the appropriate containers to/from the QC. How, then, can ports ever achieve QC rates approaching 40 moves per hour? Using a home-made simulation model of a container terminal, we show that the answer lies in the container storage and YC scheduling algorithms used within the container yard. For two different container terminals, we propose and evaluate various combinations of container storage and YC scheduling algorithms, and we identify which combination is the best. Our study ignores YTs and focuses on the question of whether it is possible to schedule YCs so they meet the deadlines set by QCs working at 40 moves per hour. Results show the speed, efficiency, and effectiveness of the proposed algorithms.

Key Words: *Maritime Container Terminal, Cargo Handling, Yard Crane Scheduling.*

1. INTRODUCTION AND LITERATURE REVIEW

The recent surge in international trade of consumer goods has placed the maritime container shipping industry at the center of the global economy. Today almost all overseas shipping of furniture, toys, footwear, clothing, auto parts, bananas, computers, and electronics components is done via standardized 20', 40', or 45' long steel containers aboard deep-sea container vessels. As of October 2005, the world cellular fleet consisted of 3488 vessels solely devoted to transporting containerized cargo. The total capacity of these vessels was some 7.77 million twenty-foot containers (Containerisation International, Nov 2005). Altogether, there are roughly 15 million containers currently circulating among thousands of factories, warehouses, distribution centers, retail stores, seaports, highways, railways, and transit depots around the world. With today's just-in-time global supply chain, improving the efficiency of container shipping processes is more important than ever.

This paper focuses on operational problems inside maritime container transhipment terminals. *Container terminals* are the places in seaports where container vessels are loaded and unloaded and containers are temporarily stored before changing to their next mode of transport—rail, road, or sea. Excellent surveys of recent research on container terminal operations have been done by Meersmans and Dekker (2001), Steenken *et al* (2004), and Vis and de Koster (2003). Good overviews of container terminal operations and equipment are

given in Kozan (2000) and Murty *et al* (2005). The storage area inside a container terminal, called the *stackyard*, typically has space for thousands of stacks of containers, with up to 6 containers per stack. The stackyard is divided into rectangular regions, called *blocks*, where a grid has been painted on the pavement indicating the positions where containers should be placed. Traffic lanes for trucks occupy the spaces between blocks. A typical block is six stacks (6*8.5 feet) wide and thirty stacks (30*20 feet) long. Blocks are divided along their length into 20' sections called *slots*. Containers are stored side-by-side and are stacked on top of one another in *rows* in rectilinear fashion in each slot (Figure 1). In many terminals, *yard zones* are formed by grouping adjacent linear blocks together. In the left portion of Figure 2, Blocks 1 and 2 make up one zone, Blocks 3 and 4 constitute another zone, and so on. Yard cranes move easily within yard zones but take a long time to move between different zones.

To manage cargo movement inside a stackyard, companies must have protocols for (1) determining storage locations for incoming cargo and (2) scheduling yard cranes to store and retrieve containers. In this paper, we develop several on-line algorithms for handling these two tasks and we evaluate their performance using a home-made computer simulation model that tracks the movement of every container that passes through a container terminal during a several-week period. Two container terminals of varying shape, storage capacity, and expected cargo throughput are considered. For each terminal, we identify the most suitable algorithms for container storage and yard crane scheduling among several alternatives. We focus on land scarce, sea-to-sea transhipment container terminals that deploy rubber tired gantry cranes (RTGCs), also known as transtainers (TTs). Such a terminal is depicted in Figure 1. In such terminals, the majority of containers are *export containers*, i.e. containers whose next mode of transport is deep-sea vessel. Some of our results are applicable to other kinds of terminals as well. The proposed algorithms are speedy routines that are tested on live data generated by the simulation model. We now survey the literature on container storage, YC scheduling, and simulation studies of container terminals.

Figure 1. A land-scarce container terminal: stackyard, TTs (foreground), QCs (upper right).

1.1. Container storage

Among hundreds of articles on container terminal operations, relatively few deal with container storage decisions. Of particular interest to us here is the storage of export containers. Extensive general discussions of export container storage can be found in Chen (1999) and Taleb-Ibrahimi *et al* (1993). The latter article develops analytical models to compare the performance of static space allocation (i.e. sort and store) strategies versus dynamic space allocation (i.e. re-marshalling) strategies. The former strategies do not allow a container's position to vary in the yard during its stay at the terminal while the latter strategies allow repositioning of containers. Kim and Bae (1998) propose a methodology for re-marshalling export containers. Bruzzone and Signorile (1998) imbed within a simulation model two genetic algorithms which provide solutions to the container storage problem.

These genetic algorithms schedule cluster creation and optimally locate clusters of containers in the export stackyard. They minimize the sum of the distances traveled by all containers over the planning horizon, but do not consider yard crane interference. Murty *et al* (2005) and Zhang *et al* (2003) consider import and export containers and assign storage spaces to arriving containers so the fill ratios of all blocks remain nearly equal. Consecutive trucks in the arriving stream are dispersed all around the stackyard to minimize traffic congestion. They allow import and export containers to be mixed at the block level. Kim *et al* (2000) propose a method for determining the storage location of an arriving export container given its weight. Kim and Park (2003) determine the storage locations of export containers over a certain horizon by categorizing containers according to the vessels they will be loaded onto, but do not consider yard crane interference. Preston and Kozan (2001) show how genetic algorithms can be used to determine storage locations for containers given fixed container handling schedules. Ambrosino and Sciomachen (2003) show how different yard management policies affect the length of time required to load a single containership.

In this study, we focus on static space allocation strategies for export container storage that are designed to minimize the occurrence of *yard crane overloading* during actual yard operations. Yard crane overloading occurs when the number of container moves set to occur in a small region of the yard over a certain time horizon exceeds a YC's actual capacity to make these moves. It is the most important consideration in yard crane operations, but is not addressed by the container storage policies in the above studies. In Section 2, we discuss yard crane overloading in detail and introduce container storage strategies designed to prevent it.

1.2. Yard crane scheduling

Many papers consider the scheduling of a single RTGC where containers are grouped and the crane must retrieve containers from specified groups according to a fixed sequence, without due dates or release times, while minimizing travel distance. Since containers belonging to a specific group may be stored in multiple locations, both the YC route and number of containers picked up at each slot are decisions to be made (Kim and Kim (1999), Kim and Kim (2003), Kim and Kim (1997), and Narasimhan and Palekar (2002)). Ng and Mak (2005) focus on a single YC scheduling problem where every job has a fixed location and ready time and the goal is to minimize job waiting time.

For the scheduling of multiple yard cranes, Cheung *et al* (2002), Linn *et al* (2003), and Zhang *et al* (2002) develop methods for allocating yard cranes among yard blocks in an entire terminal and for scheduling cross-gantry moves, but do not consider individual container moves or regular gantry moves. Kim and Bae (1998) construct detailed RTGC schedules for the re-marshalling operation within a single yard block. Kim *et al* (2004) construct detailed RTGC schedules for small scenarios involving the loading of a single vessel in isolation. Kim and Park (2004) use a greedy randomized adaptive search procedure to construct detailed schedules for QCs to process a vessel considering QC interference constraints. The basic ideas of their study may prove useful to future research on YC scheduling. Ng (2005) is the only paper we could find which proposes a method for constructing detailed schedules for all RTGCs at a container terminal on a continual basis. However, this study does not enforce the constraint that yard cranes must be separated by a minimum distance.

In this paper, we present a general method for constructing detailed schedules for all yard cranes at a container terminal on a continual basis, and we also consider crane separation constraints. In addition, our method is embedded in a simulation program to demonstrate that it works on a real-time data feed. In Section 3, we describe this method in detail.

1.3. Simulation studies of container terminals

Fifteen studies on the simulation of container terminal operations were found in the literature. Their emphasis ranges from strategic to operational aspects of container terminal management. The simulation program developed in the present study is designed to facilitate the investigation of container storage and yard crane scheduling operations. The components of a container terminal directly related to these two tasks—QCs, YCs, containers, and the stackyard—are modeled in relative detail while other components—most notably vessels—are modelled in less detail. Some components (e.g. YTs) are not modeled at all. To our knowledge, this is the first seaport simulation program that allows for a combined, detailed study of container storage and YC scheduling algorithms. Details are given in Section 4.

2. CONTAINER STORAGE

Throughout this paper, we assume that container stacks are homogenous. In other words, the container terminal adopts a storage policy whereby all containers in the same stack belong to the same *container group*. Containers in the same group are indistinguishable for vessel loading purposes because they share the same destination vessel and length, and probably the same destination port and weight class too. The advantage of a homogenous stacking policy is that containers never need to be shifted from one stack to another to dig out cargo at the bottom of a stack. We also assume that the terminal operates according to a weekly schedule. In other words, the terminal sees one vessel from a particular liner service each week at a regular time. The vessels that are seen each week follow the same routes as their respective counterparts from the previous week. Finally, we assume all containers are 20' long.

The container storage policies we consider in this paper are designed to minimize yard crane overloading. *Yard crane overloading*, sometimes referred to as *yard crane interference*, is a major concern at terminals where RTGCs are deployed. In such terminals, two yard cranes cannot operate simultaneously unless they are a minimum distance (e.g. 8 slots = 160 feet) apart. Thus, a maximum of one RTGC can be working at a time in any given 8-slot portion of the container yard. If the number of container moves scheduled in this area during a certain time interval exceeds the number of moves a yard crane can make, a yard crane is overloaded. Put another way, if YC A is working at slot #8 in a block, there can be no other activities in slots #1 - #15. A typical block has 6 rows with containers stacked 6 high. Thus there are up to 36 containers per slot and up to 36*15 = 540 storage locations that cannot be accessed while YC A is working. Very large terminals have as many as 2000 slots in the stackyard. While YC A is working, 15 out of 2000 slots are not accessible. In other words, almost 1% of the cargo in the yard is inaccessible. This constraint poses major problems for the flow of operations inside land-scarce container terminals. If containers cannot be stored or retrieved in the yard at the correct locations or times, the entire chain of events involved in transhipping a container from one vessel to another (i.e. from vessel to QC to YT to YC, then from YC to YT to QC to vessel) will be disrupted and vessels will be delayed and/or cargo will not be loaded onto vessels before they depart.

One way to avoid yard crane overloading is to choose good storage locations for containers immediately upon their arrival at the terminal. Containers that are expected to be loaded at the same time onto (one or more) vessels should be stored in locations that are at least 8 slots apart. Unfortunately, since containers can arrive at a terminal up to a week in advance of their departure, it is impossible to precisely know a container's loading time when the container enters the terminal. All that is known is the container's destination vessel and this vessel's expected arrival and departure times, which are fairly stable quantities. Within the vessel's stay, it is difficult to establish a smaller time window when the container will be loaded because the work lists for that vessel's QCs are not yet determined. Making a rough sketch of

the QC work lists before they are actually created may help to shorten the time interval, but this process is risky and can only go so far. Indeed, even for vessels belonging to liner services that call at the terminal at regular times each week, the QC job lists can show great variance from week to week. For example, in week 1, the QCs may start working at the front of the vessel and work their way backwards, whereas in week 2, they may start at the back. In week 3 the processing may consist of dozens of small patches of unloading followed by loading, but in week 4 all unloading might be completed before any loading begins. Overall, the only reliable departure information we have for a container when it arrives at a terminal is its destination vessel and this vessel's expected arrival and departure times.

Many container storage policies can be developed for minimizing yard crane overloading when the time windows for container departure are large. Among these, we choose a template-based policy in which the containers destined for a given liner service may only be stored in certain portions of the yard. We construct a *yard template* that governs all container storage activities for an extended period and remains fixed as long as the set of liner services visiting the terminal each week and their expected arrival and departure times remain unchanged. Tables 1-3 and Mathematical Program (I) show how input data is used to construct a yard template. The first step is to look at the total stackyard capacity, the stackyard shape (e.g. the block sizes), and the expected throughput for each vessel and then devise a *yard partition*. As its name suggests, a yard partition is an exhaustive partitioning of the stackyard into disjoint *regions*. Each region consists of one or more contiguous stacks and is devoted to the storage of containers that will be loaded onto a very small subset of vessels (preferably just a single vessel). The maximum size of a region is an entire block. Once the yard partition is determined and the parameters for Mathematical Program (I) are known (Table 2), Mathematical Program (I) is solved. The resulting solution gives a yard template which minimizes the long run estimated amount of yard crane overloading experienced in the terminal. Figure 2 shows two possible yard templates based on different yard partitions for a 10-block container terminal that sees 5 equally-sized vessels each week.

In Mathematical Program (I), constraint (1) ensures that each yard region is utilized. Constraint (2) ensures that enough yard space is reserved for storing the containers for each liner service, *assuming that each region stores containers that will be loaded onto a single liner service*. If it is clear at the outset that some regions will store containers belonging to more than one vessel, this constraint needs to be modified. Constraint (3) relates the two kinds of decision variables to each other.

The objective is to minimize a combination of three things: (a) the likelihood that two or more containers will be simultaneously retrieved from a small portion of the yard (where only one yard crane may work at a time); (b) the likelihood that two or more containers will be

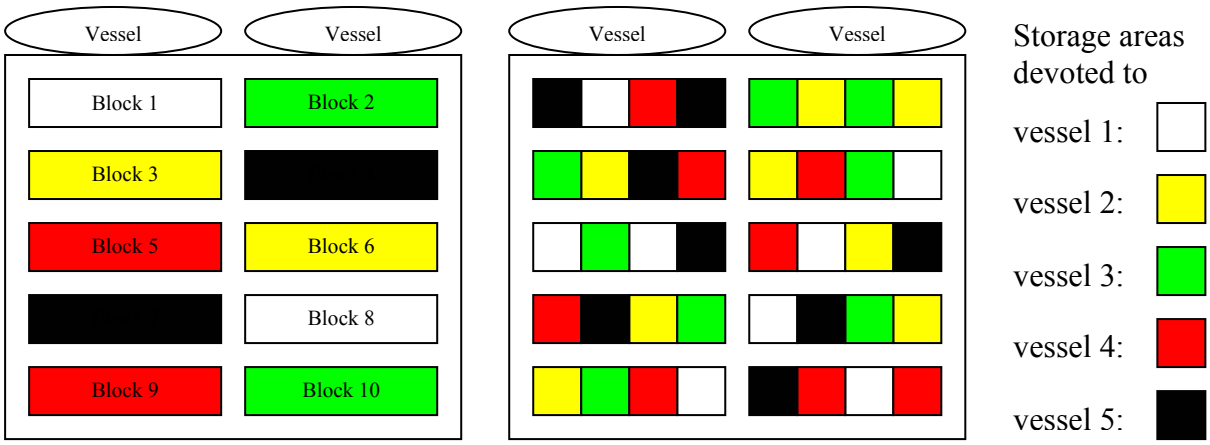

Figure 2. Two possible yard templates for a 10-block terminal with 5 vessel calls per week.

simultaneously retrieved from the same yard zone; and (c) the expected distance that containers travel over the course of a week from their stack locations to their destination vessels. The goal of sub-objective (a) is to prevent yard-crane overloading. The goal of (b) is to spread workload among different yard zones so that yard cranes do not have to make lengthy, inter-zone cross-gantry moves to help absorb the workload in other zones. Overall, we believe that sub-objective (a) is very important and therefore suggest that W_1 and W_2 take small values. In particular, because the number of terms in sub-objective (b) is much higher than (a), we suggest $W_1 = 0.00001$. Note that the first two sub-objectives are quadratic as we

Table 1. Indices for Mathematical Program (I)

r	yard region	(r = 1 to R)
s,t	stack number	(s,t = 1 to S)
v,w	vessel (i.e. liner service)	(v,w = 1 to V)

Table 2. Parameters for Mathematical Program (I)

R	Number of regions into which the yard is divided as specified by the yard partition (integer, > 0).
S	Number of stacks in the yard (integer, > 0).
M_v	Minimum number of stacks needed for storing the containers destined for vessel v (integer, > 0).
S_r	Number of stacks in yard region r as specified by the yard partition (integer, > 0).
$I_{r,s}$	= 1 if yard region r contains stack s (binary).
V	Number of weekly liner services visiting the terminal (integer, > 0).
$T_{v,w}$	Arbitrary measure of the "expected amount of time overlap" for vessels v and w at the terminal each week. If vessels v and w are expected to be at the terminal during overlapping time intervals, we suggest that $T_{v,w}$ be equal to the duration of the overlap in minutes. For the particular case $v = w$, we suggest this amount be multiplied by 10. If vessels v and w are expected to be at the terminal during non-overlapping time intervals, we suggest that $T_{v,w}$ be equal to the negative of the duration (in minutes) of the smallest interval that separates the expected departure of one vessel from the expected arrival of the other vessel, divided by 100 (real).
Sep	Minimum difference in slot numbers allowed for two cranes at the same time (e.g. 8, integer, > 0).
$C_{s,t}$	= 1 if stacks s and t are less than Sep slots apart and $s \neq t$ (binary).
$Z_{s,t}$	= 1 if stacks s and t are in the same yard zone and $s \neq t$ (binary).
$D_{s,v}$	Distance from stack s to the expected berthing position of vessel v (real, > 0).
W_1, W_2	Weighting factors used to assign relative importance to portions of the objective function (real).

Table 3. Decision variables for Mathematical Program (I)

$X_{r,v}$	= 1 if region r stores containers destined for vessel v (binary).
$Y_{s,v}$	= 1 if stack s stores containers destined for vessel v (binary).

Mathematical Progam (I):

$$\text{minimize} \sum_{s=1}^{S}\left(\left(\sum_{t:C_{s,t}=1}\sum_{v=1}^{V}\sum_{w=1}^{V}T_{v,w}Y_{s,v}Y_{t,w}\right)+W_1\left(\sum_{t:Z_{s,t}=1}\sum_{v=1}^{V}\sum_{w=1}^{V}T_{v,w}Y_{s,v}Y_{t,w}\right)+W_2\left(\sum_{v=1}^{V}D_{s,v}Y_{s,v}\right)\right)$$

$$\text{subject to} \quad \sum_{v=1}^{V}X_{r,v} \geq 1 \quad \forall r \tag{1}$$

$$\sum_{r=1}^{R}S_r X_{r,v} \geq M_v \quad \forall v \tag{2}$$

$$S_r X_{r,v} = \sum_{s:I_{r,s}=1}Y_{s,v} \quad \forall r \, \forall v \tag{3}$$

are minimizing interaction cost (Carlson and Nemhauser (1966)).

In the first two sub-objectives, we confine our attention to yard congestion caused by vessel loading operations. An online algorithm is needed for managing congestion caused by unloading operations. Such an online algorithm would be activated every time a container is about to be discharged from a vessel and could possibly be (i) interweaved with the template policy considered here; (ii) a stand-alone algorithm that still considers unloading (immediate) as well as loading (projected future) congestion; or (iii) a stand-alone algorithm that only considers unloading (immediate) congestion. All three options have promise. In this study, we only pursue option (i). Future studies will consider options (ii) and (iii). The online portion of our container storage policy is explained in Section 4.

Solving Mathematical Program (I) directly is a hopeless task, so we resort to using a simulated annealing neighbourhood search heuristic. The procedure starts with an initial feasible solution which can be generated easily. During each iteration, a neighboring solution is generated and the principles of simulated annealing are used to determine whether or not the neighbor replaces the initial solution. In the experiments described in Section 4, we perform 10,000 iterations per simulation run. To save CPU time, W_1 and W_2 are both 0.

3. YARD CRANE SCHEDULING

Among many different ways to model the multiple yard crane scheduling problem, we take the following approach. We assume the QC work lists are autonomous sequences that are constructed to maximize vessel processing efficiency assuming YTs are always available at the quay (i.e. apron) with the appropriate containers or empty trailers as desired by the QCs. Our YC scheduling algorithm is designed to react to last-minute changes in the QC work lists, but does not play any role in deciding what these changes should be.

The work list for each QC is an ordered list of container moves expected to take place in the ensuing two-hour period plus an unordered list of moves for the two-hour period after that. Every move on the list has either one or zero yard locations assigned to it. For unloading (loading) moves, this location is the stack in the yard where the container is to be stored (retrieved). For unloading moves, it is usually a stack in the yard that is partially filled by containers belonging to the same group as the unloaded container. An unloading move that does not have an associated storage location has either (a) no stacks in the yard storing containers from its group or (b) one or more stacks in the yard storing containers from its group that are all full. For loading moves that do not have associated storage locations, the exact stacks that will provide the containers have yet to be determined. In either case, some kind of online algorithm is needed to determine the storage (retrieval) locations of these containers as the time for unloading (loading) approaches.

For the purposes of yard crane scheduling, we only pay attention to those moves in the QC work lists that are ordered (i.e. are scheduled to take place within the next two hours) and have associated yard locations. Since they are ordered, tentative QC job start times can be assigned to these moves. The list of all such moves for a given QC is a QC *fixed work list*. Portions of two QC fixed work lists are shown in Table 4, columns 1-6 (L = load,U = unload).

The QC fixed work lists can be translated into an overall yard work list (Table 4, columns 5-8) using readily available data on average container handling time for QCs and YCs and average travel time for YTs between pairs of locations in the terminal. Table 4 is constructed assuming the QC and YC handling times are 1.5 and 3 minutes respectively; the YT travel times between QC 3 and blocks (5,8) are (4,3) minutes respectively; and the YT travel times between QC 9 and blocks (6,13,2,1) are (2,8,3,5) minutes respectively. The YT and YC times are equal to the average times observed during actual operations plus some buffer time. The QC handling time assumes the QC is performing 40 moves per hour. Note that we assume YTs are always on hand to receive containers from YCs and QCs. The yard work list is the

Table 4. Information from two QC fixed work lists is translated into a yard work list.

crane	move #	QC start time	QC finish time	type	yard location	YC start time	YC finish time
QC 3	23	06:24:17	06:25:47	L	bl 5, sl 7, rw 3	06:17:17	06:20:17
	24	06:25:47	06:27:17	L	bl 5, sl 7, rw 3	06:18:47	06:21:47
	25	06:27:17	06:28:47	L	bl 8, sl 22, rw 1	06:21:17	06:24:17
	26	06:28:47	06:30:17	L	bl 8, sl 22, rw 1	06:22:47	06:25:47
	27	06:30:17	06:31:47	L	bl 8, sl 22, rw 1	06:24:17	06:27:17
QC 9	81	06:26:00	06:27:30	U	bl 6, sl 40, rw 2	06:29:30	06:32:30
	82	06:27:30	06:29:00	U	bl 13, sl 9, rw 2	06:37:00	06:40:00
	83	06:29:00	06:30:30	U	bl 2, sl 14, rw 6	06:33:30	06:36:30
	84	06:30:30	06:32:00	U	bl 1, sl 32, rw 4	06:37:00	06:40:00

basis for our yard crane scheduling algorithms.

The scheduling of YCs according to the yard work list is now discussed. We first split the overall yard work list into several smaller lists, one for each yard zone. Our YC scheduling algorithms only consider YCs within a single yard zone. The study of inter-zone cross-gantry moves is a topic soon to be considered but for now is out of the scope of this paper.

3.1. Initial approach

A realistic model of the YC scheduling problem needs to consider YC gantry activities between container handling jobs. Since YC gantry speed is typically 1 slot every 4 seconds, it might make sense to discretize the time axis into 4-second intervals. Every activity is then scheduled to take place during an integral number of consecutive time intervals. Container handling moves are assumed to require exactly 45 time intervals for completion (45*4 sec = 180 sec = 3 minutes). Gantry moves require an integral number of intervals depending on the origin and destination slots. The YC start times in Table 4, column 7 are then rounded to the nearest 4 seconds; times for yard storage (retrieval) moves are rounded up (down). The YC finish times are also adjusted. Time intervals are used instead of hours, minutes, and seconds as the unit of temporal measure. Each move in the yard work list has a *target time interval* for *starting* the move in the yard. For retrieval moves, this is the latest time interval when the move can be started and still meet the deadline set by the QCs. For storage moves, this is the earliest time interval following the release of the job in the yard.

We could then construct an integer programming model for YC scheduling whose decision variables are given in Table 5. Such a program fairly accurately represents crane gantry time. However, even for a relatively small scenario of a 60-slot block with 2 YCs and 25 container moves to be made in a one-hour look-ahead window, the number of binary variables exceeds 150,000. Integer programming is therefore useless. Heuristic methods may prove useful, especially when the number of YCs is 1 or 2. But such methods appear insufficient for handling YC interference constraints when there are 3 or more cranes. In the next section, we present a novel method for yard crane scheduling which overcomes the difficulty presented by YC interference. The method does not explicitly consider cranes at the outset, but nonetheless generates realistic schedules for any number of cranes C in a very speedy manner.

Table 5. Decision variables for YC scheduling considering detailed gantry times

$X_{m,t}$	= 1 if move m is **started** during time interval t, (binary).
$Y_{m,t}$	= 1 if move m is being made during time interval t, (binary).
$Z_{c,s,t}$	= 1 if yard crane c occupies slot s during time interval t, (binary).

3.2. A simpler model

Our novel method for yard crane scheduling is a two-step procedure. Ironically, in Step 1, we do not explicitly model crane movement. Instead, we focus on the storage area proper and we fix the times and locations where container moves are made. In step 2, we use dynamic

programming to assign cranes to jobs.

We first choose a *reference time* and we discretize the time axis into 4-minute time intervals based on the reference time. Each container move shall take place during exactly one interval and shall consist of 1 minute of yard crane gantry time followed by 3 minutes of container handling. For example, in Table 4, if the reference time is 06:15:00, the YC start times in column 7 are adjusted to be (06:15:00, 06:15:00, 06:19:00, 06:19:00, 06:23:00, 06:31:00, 06:39:00, 06:35:00, 06:39:00) respectively. The YC finish times are adjusted to be 4 minutes later. Time intervals are used instead of hours, minutes, and seconds as the unit of temporal measure, and each move is assigned a *target time interval*. Each YC makes at most one container move per interval. During a given interval, both the gantrying and handling operations of different cranes start and end simultaneously; the cranes themselves are not modeled. Since an RTGC container transfer takes an average of 2.2 minutes, the 3 minutes allotted per container move is usually adequate. However, since RTGCs only gantry about 16

Table 6. Indices for Mathematical Program (II)

m,n	container move	(m,n = 1 to M)
t	time interval	(t = 1 to T)

Table 7. Parameters for Mathematical Program (II)

C	Number of yard cranes working in the zone (integer, > 0).
M	Number of individual container moves to be scheduled (integer, > 0).
T	Large number representing the number of (4-minute) time intervals on the horizon. This should be large enough to allow for all M moves to be scheduled (integer, > 0).
R	Set of yard retrieval moves.
S	Set of yard storage moves (the total number of items in R and S is M).
$Trget_m$	Target time interval for move m in the yard. For retrieval moves, this is the latest time interval that meets the deadline set by the quay cranes. For storage moves, this is the earliest time interval following the release of the job in the yard (integer, > 0, ≤ T).
$Slot_m$	Slot number where move m takes place. We assume that if $m < n$, then $Slot_m \leq Slot_n$ (integer, > 0).
Sep	Minimum difference in slot numbers allowed for two cranes at the same time (e.g. 8, integer, > 0).

Table 8. Decision variables for Mathematical Program (II)

$X_{m,t}$	= 1 if move m is scheduled to take place during time interval t (binary).

Mathematical Progam (II):

$$\text{minimize} \quad \sum_{m\in R}(Trget_m - (\sum_{t=1}^{T} t * X_{m,t})) + \sum_{m\in S}((\sum_{t=1}^{T} t * X_{m,t}) - Trget_m)$$

$$\text{subject to} \quad \sum_{t=1}^{T} X_{m,t} = 1 \quad \forall m \tag{4}$$

$$\sum_{t=1}^{T} t * X_{m,t} \leq Trget_m \quad \forall m \in R \quad , \quad \sum_{t=1}^{T} t * X_{m,t} \geq Trget_m \quad \forall m \in S \tag{5,6}$$

$$\sum_{m=1}^{M} X_{m,t} \leq C \quad \forall t \tag{7}$$

$$Slot_n - Slot_m \geq Sep * (X_{m,t} + X_{n,t} - 1) \quad \forall (m,n) : m < n \quad \forall t \tag{8}$$

slots per minute, the 1 minute allotted for gantrying between container handling operations may not be adequate. We discuss this shortly. We now have a much simpler model of YC scheduling, shown in Tables 6-8 and Mathematical Program (II).

The objective is to minimize the total amount of retrieval earliness plus storage tardiness for the upcoming jobs in the yard. The unit of measurement is container-time-intervals. This objective function is a substitute measure for the *actual* amount of earliness-tardiness, which is measured in container-minutes. The actual amount of earliness-tardiness equals 4 times the objective value plus a quantity that depends on the reference time used to discretize the time axis. Note that we do not allow retrieval lateness, which is somewhat unrealistic. Thus the YCs are being scheduled so as to strictly adhere to the fixed QC schedules.

Constraint (4) states that each move takes place during exactly 1 time interval (i.e. each move consists of a 1-minute gantry followed by 3 minutes of container handling). Constraints (5) and (6) ensure that retrieval (storage) moves are made no later (earlier) than their target time intervals. Constraint (7) ensures that no more than *C* moves take place in any time interval. Constraint (8) ensures that cranes remain at least *Sep* slots apart during all time intervals. For a 60-slot block with 25 container moves to be made over a one-hour horizon, the number of binary variables is 375, regardless of the number of YCs.

In a real setting, the above problem must be solved often, perhaps every 3-4 minutes for a given yard zone, in rolling horizon fashion. Solutions must be obtained quickly in order to be integrated with the terminal information system in real time. In particular, we must guarantee that a solution is obtained within 1 second on one hundred percent of occasions. Integer programming routines appear unsuitable for this task given their unreliable processing times.

3.3. Heuristic solution

We solve the problem using the following heuristic. For illustration purposes, assume we are scheduling 3 YCs for handling 22 jobs in a 12-slot, 4-row block where *Sep* = 2. Figure 3 shows this yard block and the moves to be scheduled. Each job is indicated by a letter (R = retrieval, S = storage) followed by the ideal start time (i.e. Table 4, column 7). We first identify the latest "R" move on the horizon. Observing that it is the "R,37" move in row 3 and slot 12, we choose 37 as the reference time. The target start times of all moves are then modified so they equal 37 + 4k for some integer k. Figure 4 displays the result. We then employ the following routine to fix the start times so they are feasible with respect to the number of YCs (7) and the minimum separation between YCs (8):

A. If there are no 'R' jobs at all, or if the start times of all 'R' jobs have been fixed, go to (E).
B. Select the latest 'R' job on the horizon that has not been fixed. If two or more such jobs tie, select the one with the latest original start time.
C. If the job selected in (B) can be fixed without violating constraints (7) and (8) considering the jobs that are already fixed, fix the job and go to (A).
D. If fixing the job selected in (B) would violate either constraint (7) or (8) considering the jobs that are already fixed, move its start time back (i.e. earlier) 4 minutes. Go to (A).
E. If there are no 'S' jobs at all, or if the start times of all 'S' jobs have been fixed, *stop the procedure*. Otherwise go to (F).
F. Select the earliest 'S' job on the horizon that has not been fixed. If two or more such jobs tie, select the one with the earliest original start time.
G. If the job selected in (F) can be fixed without violating constraints (7) and (8) considering the jobs that are already fixed, fix the job and go to (E).
H. If fixing the job selected in (F) would violate either constraint (7) or (8) considering the jobs that are already fixed, move its start time ahead (i.e. forward) 4 minutes. Go to (E).

Figure 5 shows the result after the heuristic has finished fixing the job start times. Overall, 29 iterations of the algorithm are performed, 19 for modifying/fixing the start times of the 'R' jobs and 10 for modifying/fixing the start times of the 'S' jobs.

R,8	S,14				R,23	S,0		S,4		R,8	R,32
	S,25	R,5			S,11	R,36			R,29	R,23	
		S,26				R,33	R,29				R,37
S,3		R,0				R,10	R,9				R,19

Figure 3. Initial setup for YC scheduling heuristic.

R,5	S,17				R,21	S,1		S,5		R,5	R,29
	S,25	R,5			S,13	R,33			R,29	R,21	
		S,29				R,33	R,29				R,37
S,5		R,-3				R,9	R,9				R,17

Figure 4. The job start times are adjusted according to the reference time.

R,5	S,17				R,21	S,1		S,9		R,1	R,29
	S,25	R,5			S,13	R,33			R,29	R,21	
		S,33				R,29	R,25				R,37
S,9		R,-3				R,9	R,5				R,17

Figure 5. The final job start times do not violate constraints (7) and (8).

3.4. Dynamic programming

After scheduling the times when all moves are performed, we use dynamic programming (DP) to assign yard cranes to container moves. We first convert Figure 5 into an activity matrix, where each row indicates a time interval, each column indicates a slot number, and an 'X' in row *r* column *c* indicates that an activity takes place in slot *c* during time interval *r* (Figure 6, left). Each time interval in the activity matrix is a stage in the DP model. The states in each stage of the model are the possible assignments of cranes to jobs during that stage. For *c* cranes and *j* jobs, there are (*c* choose *j*) states. In this example, there are 1 or 3 states in each stage. The transition cost from one state to another is the minimum total gantry distance travelled by all yard cranes during the 1-minute interval between handling operations. The right portion of Figure 6 shows the optimal YC assignment for this problem. Here, the 'X's are replaced by numbers indicating which crane handles each job. The costs of the transitions between stages (1 and 2), (2 and 3), and (3 and 4) are 0, 7, and 5 respectively. Figure 7 shows two YC schedules, generated using different time references, for a realistic scenario involving a 60-slot block, 2 YCs, 57 jobs, and a minimum YC separation of 8 slots. Each job is represented as a rectangle whose height is 3 minutes and width is 8 slots.

interval		slot											
start	end	1	2	3	4	5	6	7	8	9	10	11	12
-3	1			X									
1	5							X				X	
5	9	X		X					X				
9	13	X						X		X			
13	17						X						
17	21		X										X
21	25						X					X	
25	29		X						X				
29	33							X			X		X
33	37			X				X					
37	41												X

slot											
1	2	3	4	5	6	7	8	9	10	11	12
		1									
						2				3	
1		2					3				
1						2		3			
					2						
	1										3
					2					3	
	1						2				
						1			2		3
		1				2					
											3

Figure 6. The activity matrix for Figure 5, and the optimal YC assignment found using DP.

In Section 3.2. it was mentioned that, since RTGCs only gantry about 16 slots per minute, the 1 minute allotted for gantrying between container handling operations may not be adequate. This is a drawback to the scheduling paradigm introduced here. However, let us observe that the DP procedure adopted in Stage 2 is designed to minimize YC gantrying. Furthermore, since the majority of handling and gantry moves require substantially less than 3 and 1 minutes respectively, it is very likely that YCs will have more than 1 minute to make a gantry move during actual operations. For example, consider the job that the arrow points to in the left-hand schedule in Figure 7. Even if YC 4 begins this job late (due to the preceding 32-slot, (i.e. 2-minute) gantry move), it can still make up for lost time during the next 5 moves. It is unlikely that YC 4's lateness during the next 5 moves will interfere with YC 3. But in a more congested scenario with 3 YCs instead of 2, YC 4's lateness might indeed

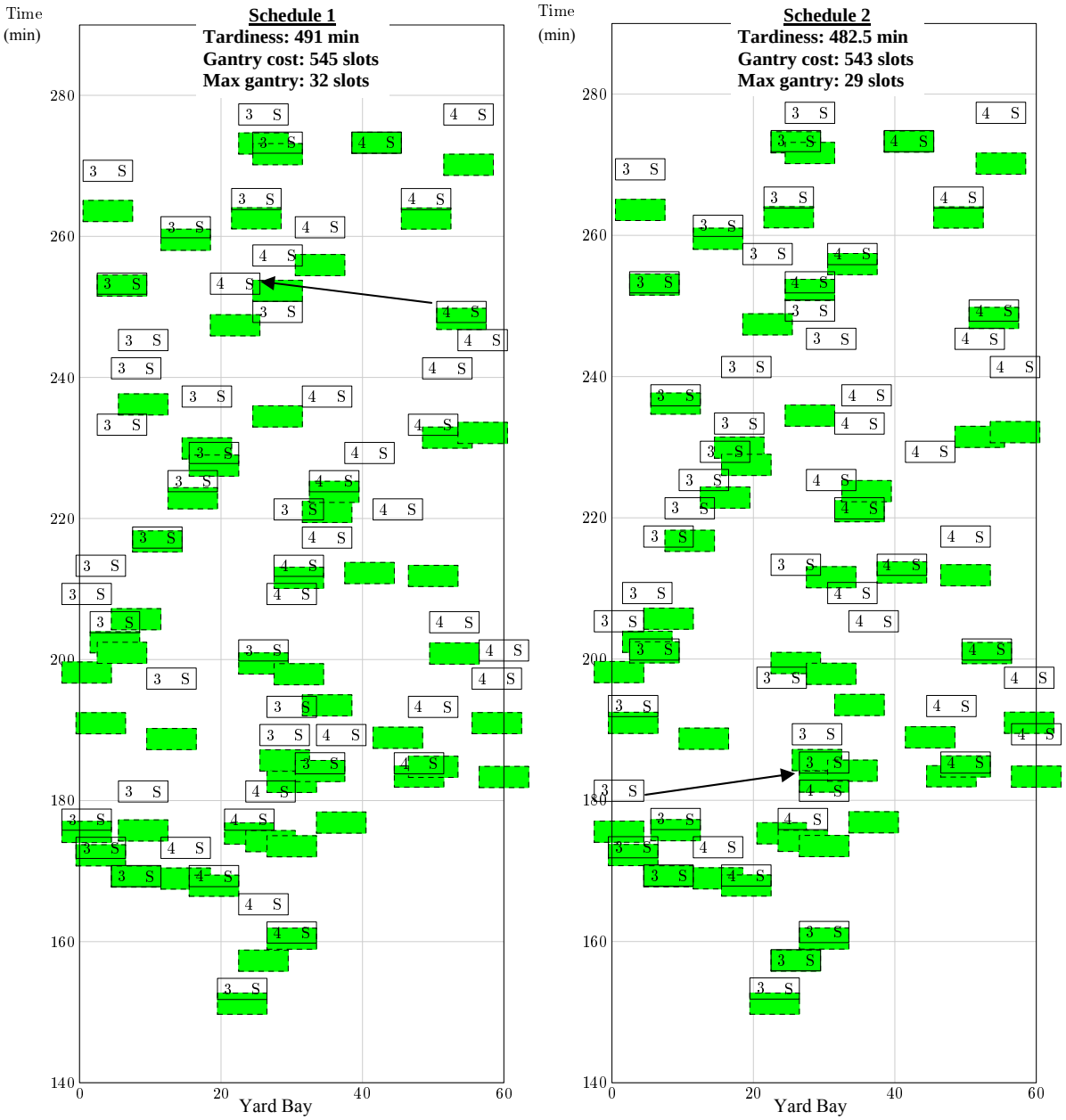

Figure 7. Two possible schedules (light rectangles) for yard cranes '3' and '4,' handling 57 expected storage jobs (dark rectangles) during a two-hour period. Arrows indicate the longest gantry move in each schedule. Both schedules are "full" between times 167 and 255.

affect YC 3. This would be a problem. However, if there were 3 YCs instead of 2, the chances that a 32-slot gantry move would be scheduled in a 60-slot block would be very low.

The schedules in Figure 7 contain discrete time/space holes that can be used as input data for online algorithms that decide which stack among many should service a last-minute QC or external truck job. For example, if suddenly a container needs to be stored at time 253, we see that YC 3 is available and could store the container anywhere among slots 1-21 during the 4-minute time interval from time 255-259 (Figure 7, left). Overall, the above YC scheduling algorithm works particularly well on congested scenarios. Thus it works best in situations where the need for decision support is greatest. Future studies will consider how use the YC schedules to guide on-line container storage algorithms such as those mentioned in Section 2.

4. SIMULATION EXPERIMENTS

Our simulation model is a computer program written in the C++ language. It tracks the movement of every container that passes through a terminal during a several-week period and has been designed to evaluate container storage and YC scheduling algorithms. It uses less than 2 hours of CPU time on a PC to simulate two weeks of activity at a mega-terminal with 7 million QC lifts annually. Within the two-week simulation period, roughly 180 vessels are processed; 270,000 individual QC lifts are made; the storage locations, and times of yard storage and retrieval, for about 140,000 containers are tracked; and the schedules for up to 120 yard cranes are constructed from scratch about 7000 times. Thus, YC scheduling takes on average less than 1 second per instance. New schedules are constructed for all YCs in rolling horizon fashion every 3 minutes of simulation time. The planning horizon is one hour. We assume all containers are 20 feet long. Containers in the QC lists always have an assigned yard location unless they are the first container from their respective group to be unloaded in a particular week. Such first arrivals are called *trailblazing containers*. As the unloading of such a container approaches, an online procedure is activated which chooses a stack for the container among those stacks devoted to that container's destination vessel. In this paper, the algorithm simply selects a random stack among all possible candidates. More sophisticated procedures will be studied in the future.

Parameters that can be adjusted include the number of berths; number of scheduled vessel calls per week; number of QCs processing each vessel; number of QC lifts for each vessel; number of yard blocks and zones; number of rows and slots per block; number of YCs operating in each zone; duration of stay for each vessel; time required per handling move for QCs and YCs; YC gantry time allotted between handling moves; and the yard partition.

In the experiments, we consider two container terminals: a 2-berth terminal (Terminal 1) and a 9-berth mega-terminal (Terminal 2). The specifications of these terminals are given in Table 9. Both terminals are strictly sea-to-sea transhipment terminals. For the small and large terminals, we test 15 and 8 different yard partitions respectively. In each partition, all regions have the same size as shown in Table 10. In partitions 13-15 for Terminal 1 and 7-8 for Terminal 2, regions are smaller than 1 slot; for all other partitions, the regions consist of sets of contiguous slots. The number of vessels per region is always 1. For each combination of terminal and yard partition, we test thirteen different scenarios. In each scenario, a unique combination of values is assigned to five different parameters: the number of YCs per zone; minimum separation between YCs (in slots); YC handling time per job; YC gantry time allotted between handling jobs; and whether clumping is used in the YC scheduling algorithm. Clumping saves CPU time by scheduling all jobs in the same stack consecutively, but not necessarily on the same YC. Scenario 1 is considered standard operating practice; other scenarios are designed for comparison with Scenario 1. The scenarios, and the results of the simulation experiments, are shown in Tables 11 and 12. For each scenario and yard partition, we show the average amount of earliness-tardiness (in container-minutes) observed

per YC schedule. This value indicates how difficult it is for the YCs to process the workload generated by the QCs, which are working at 40 moves per hour. A high (low) value means that the YCs have substantial (little) difficulty keeping up with the QCs. YC schedules are

Table 9. Container terminals used for simulation experiments

	Terminal 1	Terminal 2
berths	2	9
vessel calls per week	10	90
expected QC lifts per week	35000	135000
yard blocks, zones	10, 5	45, 15
slots per block	60	40
rows per slot	6	8

Table 10. Yard partitions considered for Terminals 1 and 2

yard partition	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
stacks/region for Terminal 1	360	180	120	90	72	60	36	30	24	18	12	6	3	2	1
stacks/region for Terminal 2	160	80	40	32	16	8	4	2	-	-	-	-	-	-	-

Table 11. Average total job earliness + tardiness in YC schedules for Terminal 1

Scenario		1	2	3	4	5	6	7	8	9	10	11	12	13
cranes/zone		4	3	5	6	7	8	4	4	4	4	4	8	8
separation		8	8	8	8	8	8	4	12	8	8	8	2	2
handl time		3	3	3	3	3	3	3	3	3	3	3	1.5	1.5
gantry time		1	1	1	1	1	1	1	1	0	2	1	0	0
clumping?		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N
	1	7414	10314	6056	6015	5801	5813	6996	8764	4695	11032	7166	277	267
	2	5926	7048	5501	5678	5550	5470	4244	8098	3993	8180	6022	271	256
	3	5678	6606	5330	5353	5237	5234	4275	7444	3499	7943	5428	273	264
	4	5327	5991	4926	4989	5026	4825	3541	6998	3214	7449	5155	274	266
	5	4879	5541	4659	4679	4667	4711	3552	6179	3099	7039	4819	267	261
	6	4723	5788	4528	4492	4471	4503	3178	5202	2864	6577	4673	268	262
yard	7	3234	3954	3094	3056	3045	3097	2799	3518	1968	4787	3239	268	262
partition	8	2992	3784	2768	2691	2726	2718	2789	3041	1738	4313	2933	265	259
	9	2653	3449	2395	2393	2329	2386	2524	2782	1528	4035	2593	264	254
	10	2340	3234	2140	2114	2034	2057	2426	2522	1350	3877	2286	253	251
	11	2210	3284	1850	1973	1824	1741	2102	2378	1296	3234	2053	242	237
	12	1918	3444	1789	1629	1650	1612	1850	2237	1113	2989	1777	217	215
	13	2208	3014	1827	1732	1626	1551	1829	2380	1166	3392	1933	197	206
	14	2064	3100	1744	1716	1626	1624	1899	2374	1188	3177	2056	199	201
	15	2016	3030	1762	1631	1732	1731	1834	2281	1158	2973	2036	202	199

Table 12. Average total job earliness + tardiness in YC schedules for Terminal 2

Scenario		1	2	3	4	5	6	7	8	9	10	11	12	13
cranes/zone		4	3	5	6	7	8	4	4	4	4	4	8	8
separation		8	8	8	8	8	8	4	12	8	8	8	2	2
handl time		3	3	3	3	3	3	3	3	3	3	3	1.5	1.5
gantry time		1	1	1	1	1	1	1	1	0	2	1	0	0
clumping?		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N
	1	43426	46719	43091	42844	42663	42968	25640	59853	29655	57939	41899	1444	1343
	2	37881	39344	37303	37292	37111	37272	22294	44336	25197	50664	36423	1393	1331
	3	21250	22745	20950	20930	20913	20918	18582	21806	13397	29367	20598	1341	1276
yard	4	17349	19176	16956	16868	16942	16912	16890	17816	10583	24211	16554	1303	1237
partition	5	10692	12999	10062	9945	10120	9983	10503	11495	6326	15469	9972	1136	1084
	6	8450	11434	7714	7534	7372	7679	7826	9296	5121	13859	8015	861	836
	7	8198	11921	6939	6822	6856	6739	7390	8817	4675	12452	7392	747	732
	8	7628	10971	6727	6479	6305	6398	7317	8564	4455	11868	7340	700	694

generated on a terminal-wide basis. Each entry in Table 11 (12) is obtained by averaging results for all YC schedules generated during 3 (2) two-week simulation runs. In each run, a unique yard template is constructed before containers begin to flow. The entries in Table 12 are larger than in Table 11 because more jobs are considered per YC schedule.

Results for the two terminals follow the same general trend. Overall, for both terminals, the higher-numbered yard partitions vastly outperform the lower-numbered ones. Indeed, the best partition for each scenario divides the stackyard into regions that are at most the size of a single slot. This is true despite the fact that near-optimal yard templates were constructed for *every* partition using simulated annealing. Note that the higher-numbered partitions when YC gantry time is 2 minutes (Scenario 10) still outperform the lower-numbered partitions when gantry time is only 1 minute (Scenario 1). These results show that it is best to spread out cargo in the yard. They also indicate that pure online container storage algorithms have excellent promise. These online algorithms do not use yard templates and do not store containers in batches. They decide where to store a container based upon a scan of the real-time status of every stack in the yard just prior to unloading from a vessel.

Results for Scenarios 2-6 show that increasing the number of YCs per zone beyond 5 yields very little benefit. Scenarios 7 and 8 show how crane separation constraints hamper YC operations. In Scenario 7, cranes are allowed to come within 4 slots of each other, while in Scenario 8, they must remain at least 12 slots apart. Results show that crane separation requirements definitely affect the operational efficiency of YCs, but perhaps less than was expected. Scenarios 9 and 10 show that YC gantry speed (and by extension YC handling speed) plays a large role in determining YC operational efficiency. Scenario shows 11 shows that clumping has a slight negative effect on the quality of YC schedules. Scenarios 12 and 13 are the terminal manager's dream; YCs are ubiquitous and each crane works as fast as a QC. These scenarios are used as a control to make sure that earliness-tardiness is being measured properly. The majority of earliness-tardiness in these scenarios comes from the initial adjustment of YC start times based on the reference time. Overall, the best yard partitions for Terminals 1 and 2 are partitions 12 and 8 respectively. For reasons of computational efficiency, the YC scheduling algorithm that uses clumping is preferred.

5. CONCLUSION AND FUTURE RESEARCH

In this paper, we developed on-line algorithms for making container storage decisions and constructing yard crane schedules inside a maritime container transhipment terminal. We evaluated their performance using a home-made computer simulation model that tracks activity at a container terminal during a several-week period. Although our results do not conclusively show how ports can achieve QC rates approaching 40 moves per hour, they do indicate that QC rates can be increased by spreading out cargo in the container yard. So far, we have only investigated template-based container storage policies and two different YC scheduling algorithms. Future studies will expand the number of algorithms considered and will compute the long-run, average QC rates that these algorithms are capable of sustaining.

ACKNOWLEDGEMENTS

The authors wish to thank Ku Liang Ping of PSA Corporation for several enlightening discussions on container terminal operations and Robert de Souza at the Logistics Institute—Asia Pacific for providing the facilities used to conduct this research.

REFERENCES

Ambrosino D and Sciomachen A (2003), Impact of yard organisation on the master bay planning problem, *Maritime Economics and Logistics* **5**, 285-300.

Bruzzone A and Signorile R (1998), Simulation and genetic algorithms for ship planning and shipyard layout, *Simulation* **71**, 74-83.

Carlson R C and Nemhauser G L (1966), Scheduling to minimize interaction cost, *Operations Research* **14**, 52-58.

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

Cheung R K, Li C, and Lin W (2002), Interblock crane deployment in container terminals, *Transportation Science* **36**, 79-93.

Kim K H and Bae J W (1998), Re-marshaling export containers in port container terminals, *Computers and Industrial Engineering* **35**, 655-658.

Kim K H, Kang J S, and Ryu K R (2004), A beam search algorithm for the load sequencing of outbound containers in port container terminals, *OR Spectrum* **26**, 93-116.

Kim K H and Kim K Y (1999), An optimal routing algorithm for a transfer crane in port container terminals, *Transportation Science* **33**, 17–33.

Kim K H and Park K T (2003), A note on a dynamic space-allocation method for outbound containers, *European Journal of Operational Research* **148**, 92–101.

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

Kim K H, Park Y M, and Ryu K R (2000), Deriving decision rules to locate export containers in container yards, *European Journal of Operational Research* **124**, 89–101.

Kim K Y and Kim K H (2003), Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals, *Naval Research Logistics* **50**, 498-514.

Kim K Y and Kim K H (1997), A routing algorithm for a single transfer crane to load export containers onto a containership, *Computers and Industrial Engineering* **33**, 673-676.

Kozan E (2000), Optimising container transfers at multimodal terminals, *Mathematical and Computer Modelling* **31**, 235-243.

Linn R, Liu J,Wan Y, Zhang C, and Murty K G (2003), Rubber tired gantry crane deployment for container yard operation, *Computers and Industrial Engineering* **45**, 429-442.

Meersmans P J M and Dekker R (2001), Operations research supports container handling, Econometric Institute Report EI 2001-22.

Murty K G, Liu J, Wan Y, and Linn R J (2005), A decision support system for operations in a container terminal, *Decision Support Systems* **39**, 309-332.

Narasimhan A and Palekar U S (2002), Analysis and algorithms for the transtainer routing problem in container port operations, *Transportation Science* **36**, 63-78.

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

Ng W C and Mak K L (2005), Yard crane scheduling in port container terminals, *Applied Mathematical Modelling* **29**, 263-276.

Preston P and Kozan E (2001), An approach to determine storage locations of containers at seaport terminals, *Computers & Operations Research* **28**, 983-995.

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

Taleb-Ibrahimi M, Castilho B D, and Daganzo C F (1993), Storage space vs handling work in container terminals, *Transportation Research B* **27**, 13-32.

Vis I F A and de Koster R (2003), Transshipment of containers at a container terminal: an overview, *European Journal of Operational Research* **147**, 1-16.

Zhang C, Liu J, Wan Y, Murty K G, and Linn R J (2003), Storage space allocation in container terminals, *Transportation Research B* **37**, 883-903.

Zhang C, Wan Y, Liu J, and Linn R J (2002), Dynamic crane deployment in container storage yards, *Transportation Research B* **36**, 537-555.