OR Spectrum (2009) 31:801–835
DOI 10.1007/s00291-008-0142-7

REGULAR ARTICLE

Development and simulation analysis of real-time yard crane control systems for seaport container transshipment terminals

Matthew E. H. Petering · Yong Wu · Wenkai Li · Mark Goh · Robert de Souza

Published online: 30 July 2008

Abstract As more and more container terminals open up all over the world, terminal operators are discovering that they must increase quay crane work rates to remain competitive. In this paper, we develop a real-time yard crane control system and show that a terminal's long-run average quay crane rate depends on the portion of this system that dispatches yard cranes in the storage area in real time. Several real-time yard crane dispatching systems are evaluated by a fully-integrated, discrete event simulation model of a pure transshipment terminal that is designed to reproduce the multi-objective, stochastic, real-time environment at an RTGC-based, multiple-berth facility. Results indicate that yard cranes should prioritize the retrieval of containers from the stacks, rather than the storage of containers into stacks. Also, the yard crane dispatching system should not only consider the trucks already waiting for service in the yard, but also the trucks that are heading towards the yard. The experiments

M. E. H. Petering (✉)
Department of Industrial and Manufacturing Engineering, University of Wisconsin—Milwaukee, P. O. Box 784, Milwaukee, WI 53201, USA
e-mail: mattpete@uwm.edu

Y. Wu · W. Li · M. Goh · R. de Souza
The Logistics Institute—Asia Pacific, National University of Singapore, Block E3A Level 3, 7 Engineering Drive 1, Singapore 117574, Singapore
e-mail: tliwy@nus.edu.sg

W. Li
e-mail: tlilw@nus.edu.sg

R. de Souza
e-mail: tlirbrtm@nus.eds.sg

M. Goh
NUS Business School, National University of Singapore, 1Business Link, Singapore 117592, Singapore
e-mail: bizgohkh@nus.edu.sg; tligohkh@nus.edu.sg

provide the first direct connection in the literature between real-time yard crane control systems and long-run performance at a seaport container terminal. We also make a qualitative comparison between rule-based and look-ahead yard crane dispatching schemes, and discuss deadlocking issues in detail.

Keywords Seaport container terminal · Yard crane dispatching · Real-time control · Gross crane rate · Simulation

1 Introduction

The recent increase in international trade of finished consumer goods has placed the maritime container shipping industry at the center of our global economy. Today, almost all overseas shipping of furniture, toys, footwear, clothing, auto parts, electronics components, and computers is done via standardized 20′, 40′, and 45′ long steel containers aboard deep-sea container vessels. In addition, the amount of fruit, vegetables, fish, meat, and general foodstuffs shipped in refrigerated containers is increasing.

As of March 2008, the world cellular fleet consisted of 4,393 vessels solely devoted to transporting containerized cargo. The total capacity of these vessels was some 11.1 million twenty foot containers (*Containerisation International*, May 2008). Four years earlier, in February 2004, the figure was only 6.54 million 20 foot containers (*Containerisation International*, March 2004). This represents a growth rate exceeding *1% per month*. Such a rapid expansion of the container sector, combined with a heightened concern over customer service and security, has made container shipping a major focus of operational research in the past decade. Indeed, 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 control problems at seaport container terminals. Container terminals are places in seaports where container vessels are loaded and unloaded, and where containerized cargo is temporarily stored while awaiting a future journey. A brief summary of container terminal equipment and operations now follows.

Containers (boxes) come in three standard lengths—20′, 40′, and 45′—and are 8′ wide and 8.5′ or 9.5′ high. When a containership arrives at the port, the terminal provides a berth where it docks. Then the QCs (quay cranes, shore cranes) begin unloading and loading it, with each QC handling cargo in a different section along the length of the vessel known as a hatch. Large vessels have up to two dozen hatches, so typically three to four QCs work on a vessel at a time.

Cargo may pass through a container terminal in three ways: it may be imported, exported, or transshipped. Figure 1 shows these processes and the equipment involved. Import containers arrive by vessel and leave the terminal via truck or train. These containers are first unloaded by a QC which puts them onto a YT (yard truck, internal tractor, prime mover, hustler, UTR) waiting under it on the ground. The YT then drives the container to a storage yard (container yard, yard) where a YC (yard crane, rubber tired gantry crane, RTGC, transtainer, TT) picks it off the YT's trailer and places it in a stack in the yard. At some later time, a YC retrieves the container from the yard and places it onto an XT (external truck) or train, which takes the container through the

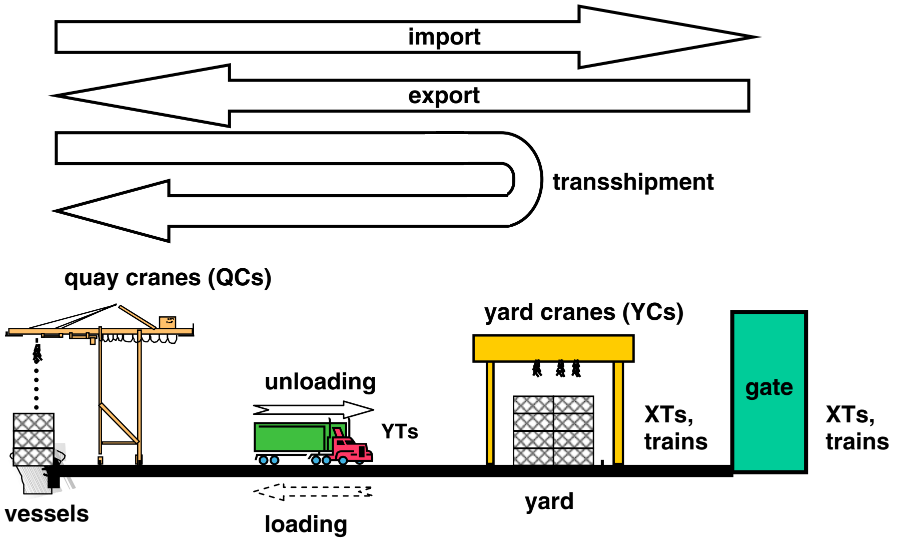

Fig. 1 Cargo may pass through a container terminal in three different ways

gate to its mainland destination. In this paper, we assume that all YTs are manually driven. In addition, we assume an RTGC-based yard operation.

Export containers are packed off-site and delivered by XTs or trains to the terminal. At the terminal these containers are removed from XTs or trains, and put into temporary storage in the yard by the YCs. When the vessel into which they are to be loaded docks at the terminal, these containers are retrieved from storage by YCs and carried by YTs to the berth; these YTs park under the appropriate QC which lifts the containers and loads them into the vessel.

Containers that are transshipped both arrive and depart by vessel. Upon arrival, each such container is unloaded by a QC, transported via YT from the quay to the yard, and then placed in a stack in the yard by a YC. When the vessel into which it will be loaded arrives at the terminal, the container is then retrieved by a YC, transported via YT from the yard to the quay, and then loaded by a QC into the vessel.

Figure 2 shows the general layout of a container terminal from a bird's eye view. Vessels, QCs, YCs, and YTs are labeled for easy identification. The yard is divided into rectangular regions called blocks. The width of a block is typically divided into seven rows—six for stacks of containers and the seventh for trucks that interact with the YCs. Traffic lanes for trucks occupy the spaces between blocks. Blocks are divided along their length into $20'$ sections called slots. A typical block is about forty slots long. Two adjacent slots in the same block comprise a $40'$ yard bay. The region occupied by a stack of $20'$ containers is called a groundslot ($20'$ stack) and that occupied by a stack of $40'$ containers as a $40'$ stack. $40'$ stacks occupy two adjacent groundslots in the same row. In each stack, containers are stored one on top of the other 3 to 6 tiers high depending on the height of the YC serving the block.

Yard cranes transfer containers between trucks (YTs and XTs) and the stacks in the yard. They straddle the entire width of the block beneath them and move along the

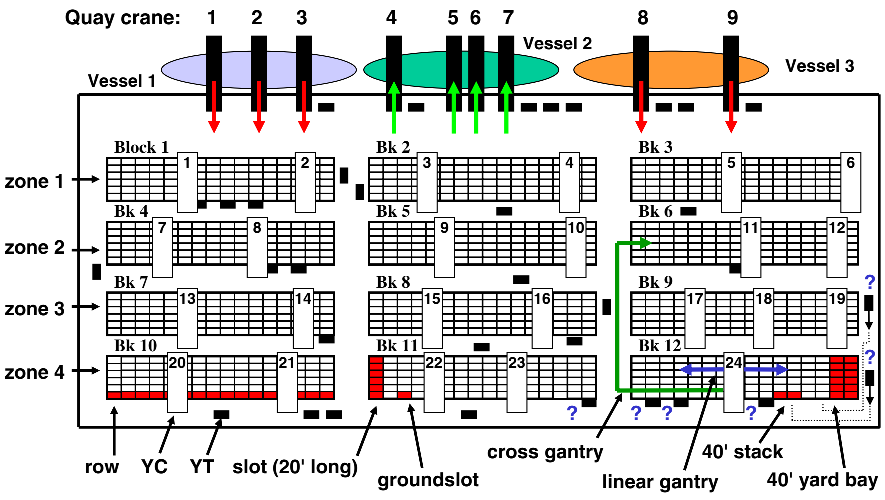

Fig. 2 Bird's eye view of a container terminal

length of the block. A zone is a sequence of blocks that together form a single lane for YC movement. In Fig. 2, blocks 1–3 are in zone 1; blocks 4–6 are in zone 2; and so on. YCs move easily within a block and from block to block within a zone; such movement is called linear gantrying. But to move between zones, YCs must spend at least 15 min executing a maneuver called a cross gantry. A cross-gantrying YC blocks the road for a long duration, thus disrupting traffic. YTs carry one 20′, one 40′, one 45′, or two 20′ containers at a time. Empty YTs travel faster than laden YTs.

Quay cranes load and unload containers according predetermined job sequences that have limited flexibility for changing the order in which lifts are performed. A QC's top speed is typically 40 lifts (moves between vessel and shore) per hour if it does not have to wait for YTs under it to take away the import containers it is unloading, or to bring export containers for it to load while it is loading. However, QCs at most terminals average only 25 or so lifts/h. Each lift typically involves one 40′ container, one 20′ container, or two 20′ containers. The average number of lifts achieved at a terminal per QC working hour is known as the GCR (gross crane rate, QC rate). GCR is perhaps the most important performance measure of a terminal. In this paper, we use GCR as the measure of performance to maximize over the long run. To attain a high GCR, the flow of containers between the shore and the yard has to proceed smoothly like clockwork, so that QCs do not incur idle time waiting for YTs.

Maximizing GCR is important for several reasons. For the terminal operator, a higher GCR results in greater business turnover using the same equipment and labor force. For the vessel operator (shipping line), a higher GCR leads to higher vessel utilization as vessels spend less time at port and more time at sea. With annual revenues for the largest container terminal operating companies at around USD $2 billion and revenues for the largest liner shipping companies as much as ten times this amount, the potential for financial gain by improving GCR is nontrivial. Maximizing GCR is also smart from an environmental perspective. Indeed, if existing terminals and vessels can operate at higher efficiencies, the need for building new terminals and vessels will be diminished.

This paper is organized as follows. Section 2 describes the problem in detail. Section 3 summarizes the relevant literature. Section 4 discusses the core YC control system developed in this research; rule-based versus look-ahead YC dispatching algorithms; and deadlocking issues. Section 5 describes the simulation model used in the experiments. In Sect. 6, we describe the experiments, present the results, and discuss their significance. We conclude in Sect. 7.

2 Problem description

2.1 Container yard operations

In this paper, we investigate how various real-time yard crane dispatching systems affect the long-run performance of a container terminal as measured in terms of GCR. To attain a high GCR, the activities in the storage yard should be properly coordinated so the YCs and YTs serve the QCs effectively. However, several aspects of real-time yard operations make this a challenging task. First of all, the maximum handling speed of a YC is roughly 25 lifts/h, much slower than a QC's 40 lifts/h. Secondly, YCs, unlike QCs, must multi-task. When two or more vessels are present, YCs often have to store import containers being unloaded from one vessel while also retrieving stored export containers that are loaded onto other vessels. For example, YC 24 in Fig. 2 must be ready to serve not only the YTs that are bringing containers from QCs 1–3 but also the YTs that are retrieving containers to be loaded by QCs 4–7 into Vessel 2. Thirdly, YCs move great distances while QCs are virtually immobile. Hence, at least 2–3 YCs are typically needed per QC to keep the QCs smoothly working at a high speed. Fourthly, most terminals never close; the workload must be processed 24 h per day, 365 days per year. Fifthly, there is an uneven distribution of workload over time. The varying workload is caused not only by scheduled vessel arrivals that are known well in advance, but also by unpredictable events such as late vessel arrivals due to weather conditions at sea or delays at a previous port. Sixthly, YC (YT) gantrying (traveling) speed varies with operator style, operator error, and real-time road traffic conditions inside the terminal.

Seventhly, YCs and QCs have highly variable container handling times. For example, according to a major terminal operator, the time taken by a YC to handle a single container is a triangularly distributed random variable with parameters (1.2, 2.0, 3.4) min. The high variance is due to (1) the trial-and-error method by which crane operators place containers precisely onto stacks in the yard, (2) the varying skill levels of YC operators, (3) the varying skill levels of the YT operators with whom YCs interact, (4) the clashing or conformity of the individual styles of the YC and YT operators involved in an individual transaction, (5) the different stack locations to/from which containers are transferred (some are close to the YT, others far), and (6) the possibility that a container might be hoisted higher than normal to avoid hitting other containers already stacked in the same slot. The large variation in YC handling time makes it difficult to coordinate the operations of multiple YCs working in close proximity.

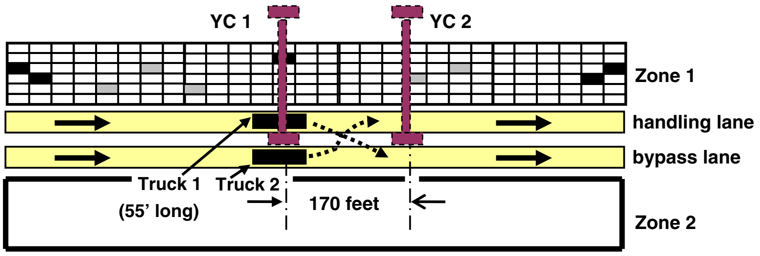

Fig. 3 Two YCs in the same block must be separated by at least 170 feet in order for two streams of (55-foot long) trucks to be independently served by the two YCs without delay

Finally, YCs that come too close to each other are subject to slowdowns. Figure 3 illustrates how such a slowdown can develop. In the figure, Trucks 1 and 2 are attempting to transfer containers to/from stacks in the yard that lie beneath YCs 1 and 2 respectively. The trucks, each 55 feet long, are assumed to be mobile and the YCs are assumed to be stationary. If YCs 1 and 2 are less than 170 feet apart, Truck 2 does not have enough space to pull into the handling lane and become parallel to the storage block before reaching YC 2. Even if Truck 2 somehow manages to reach YC 2, Truck 1 does not have enough space to pull out of the handling lane and into the bypass lane before hitting the backside of Truck 2. In other words, the trucks have difficulty (A) gaining access to the downstream YC and (B) departing from the upstream YC. The overall result is a YT traffic jam that cuts YC productivity in half. Thus, two YCs in the same block must be separated by $170'$ (8 slots when accounting for spaces between stacks) in order for two streams of trucks to be served independently by the YCs without delay.

2.2 Real-time yard crane control

Within the above environment, the YC control system is responsible for determining the sequence of linear gantry, cross gantry, container retrieval, container storage, and idling operations for all YCs at all times. The decisions made by this system are represented by the three arrows and six question marks near YC 24 in Fig. 2. The six question marks correspond to six different trucks—four are waiting for service near YC 24, and two are traveling towards yard locations near YC 24. If YC 24 becomes free, which of these six trucks should it serve next? Should the YC dispatching system consider the two trucks that are still traveling, or should these trucks be ignored? These questions are the concern of the YC control system.

Currently, most terminals around the world do not have an automated system for controlling or dispatching YCs. Instead, YC dispatching decisions are made on the fly by the operators of the YCs. This paper indicates that such an arbitrary control mechanism may be inferior to one that is more automated. Indeed, our experiments show GCR differences between various automated YC dispatching systems exceeding 10% in some cases. Thus, the tens of thousands of YC dispatching micro-decisions made per day at a large facility can have a major impact on the overall, long-run productivity of a container terminal, even though each individual decision has virtually no impact on productivity. In this paper, we develop and evaluate several YC control systems, each of which (1) tracks the real-time status of all YCs at all times, (2) controls the detailed movement and container handling tasks undertaken by each YC at all times, (3) guarantees that YCs remain separated by at least 170 feet at all times, and (4) guarantees that operational deadlocks are avoided on 100% of occasions. We now discuss the relevant literature.

3 Literature review

The literature relevant to this investigation includes all papers on container terminals that discuss YC operations, simulation modeling, or the literature itself. Excellent surveys of the container terminal literature have been done by Stahlbock and Voß (2008), Steenken et al. (2004), and Vis and de Koster (2003). An overview of container terminal operations is given by Günther and Kim (2006). Summaries of the various operational decisions made in container terminals are given in Murty et al. (2005a,b).

Forty-one container terminal simulation models were found in the literature. Some of the recent models include: Alessandri et al. (2007), Briskorn et al. (2006), Canonaco et al. (2008), Dekker et al. (2006), Froyland et al. (2008), Grunow et al. (2006), Kim et al. (2003), Kozan (1997), Legato and Mazza (2001), Linn et al. (2003), Liu et al. (2002, 2004), Ottjes et al. (2006), Parola and Sciomachen (2005), and Yang et al. (2004). The emphasis of these studies ranges from strategic to operational aspects of container terminal management. The articles by Liu et al. (2002, 2004) offer perhaps the most comprehensive models among those listed above. Their models are unique in that they (A) measure performance using a global indicator such as GCR; (B) track other performance measures such as yard utilization, truck productivity, and YC productivity; and (C) show how different values of an input parameter (e.g., the type of handling equipment used, vehicle fleet size, terminal layout) affect overall performance as measured by GCR or average vessel turnaround time. One limitation common to their models, and most other simulation models in the literature, is that they only consider a single vessel in isolation. Thus, all yard equipment is devoted to serving a single vessel at a given time. In addition, the duration simulated is only 1 day (i.e., the time taken for processing a single vessel). The simulation model presented in this paper, on the other hand, considers the detailed operations at a multiple-berth facility over an extended time period that is user-defined (e.g., 3 weeks, 6 months).

Regarding YC control, many papers develop integer programming models for the off-line scheduling of a single YC. Kim et al. (2003) compare real-time YC dispatching policies similar to those considered in this paper for a small scenario involving a single YC operating in three adjacent 25-slot blocks. Papers considering the control of multiple yard cranes typically fall into two categories. Some articles develop methods for allocating YCs among yard blocks and for scheduling inter-block YC movements. However, those articles do not consider individual container handling tasks and other details relevant to real-time YC control. On the other hand, many other papers present detailed integer programming models for scheduling multiple YCs or for the integrated scheduling of YCs together with YTs and/or QCs. Overall, none of the above studies considers the real-time control of multiple YCs. In addition, no article requires YCs to be separated by a minimum distance so they can be productive (Fig. 3).

Very few articles consider the real-time control of multiple YCs. Petering et al. (2006) discuss the real-time control of multiple YCs but do not present experimental results. Petering and Murty (2008) use the same simulation model presented here to show how real-time, inter-block YC deployment systems affect GCR at a multiple-berth container terminal. However, they do not compare alternative systems for the detailed assignment of container handling tasks to YCs in real time. Thus, a comprehensive analysis of how real-time YC dispatching should be done at an operational level is still lacking. The current research helps to fill the above gap by considering the real-time dispatching of YCs at a pure transshipment terminal where RTGCs are deployed. In particular, we use the model described in Petering (2007), Petering and Murty (2008), and Petering and Murty (2007) to directly compare several real-time YC dispatching systems in terms of the GCR that they can sustain over the long run at a multiple-berth container terminal. In doing so, we expand upon the dialogue that was begun in Petering et al. (2006).

Overall, a review of the literature has yielded many outstanding articles but no models that test how *real-time* decision making systems affect the overall performance (i.e., GCR) of a *multiple-berth* container terminal over *an extended period*. In Sects. 5 and 6, we present a simulation model capable of running such tests and show how it is used to obtain new numerical results comparing real-time YC dispatching systems. We now discuss the YC control system.

4 YC control system; rule-based versus look-ahead YC dispatching; deadlocking

4.1 The core YC control system

Our efforts to create a real-time YC control system resulted in the development of a core system that (1) tracks the real-time status of all YCs at all times, (2) determines the exact routing of every YC at all times, (3) guarantees that YCs remain separated by at least $170'$ at all times, and (4) guarantees that operational deadlocks are avoided on 100% of occasions. This system is designed for use at a *pure vessel-to-vessel transshipment terminal*. Various YC dispatching algorithms can be plugged into the core system to be tested and compared.

The core YC control system allows each YC to be in one of seven states at a given time. Figure 4 shows the seven possible states for YCs and how the status of a YC may change as operations unfold. YCs may pass through idling, waiting, or linear gantrying states instantaneously if conditions are right. Note that YCs are assumed to be 100% reliable.

The YC dispatching system controls the detailed movement of each YC by generating instructions for YCs whenever they become free, i.e., whenever they enter state 1. The first part of a YC's instruction is always a linear gantry move to a specific slot. Once the YC finishes linear gantrying and arrives at the destination slot, the second part of the instruction tells the YC to either (1) make a cross-gantry move, (2) handle a

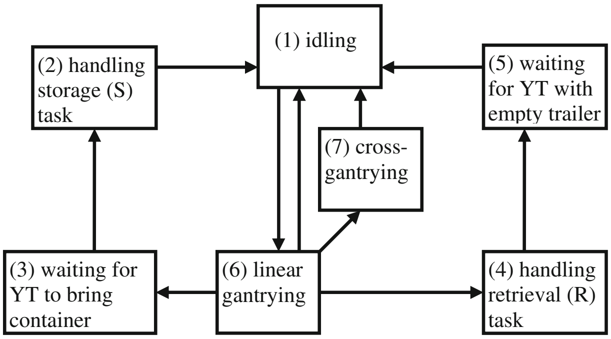
(1) idling
(2) handling storage (S) task
(3) waiting for YT to bring container
(4) handling retrieval (R) task
(5) waiting for YT with empty trailer
(6) linear gantrying
(7) cross-gantrying

Fig. 4 The core YC control system allows a YC to be in one of seven states at a given time

container at that slot, or (3) become idle. For case (1), the aforementioned destination slot must be either the first or last slot the YC's block. For case (2), the instructions also specify the type of job to be performed: 'S' (take container from a YT and store in a stack) or 'R' (retrieve from a stack and give to a YT). If no such YT is present and the job type is 'S', the YC waits at the slot until a YT with instructions containing the appropriate job type arrives at that slot. Once the YT arrives, the YC begins the storage operation. On the other hand, if no such YT is present and the job type is 'R', the YC still goes ahead with the retrieval. After the YC has finished the retrieval and is still grasping the container in its spreader, it may or may not have to wait for a suitable YT to arrive with the appropriate status. Once the YT arrives, the YC immediately transfers the container to the YT and then receives new instructions. If two or more YTs happen to be present at the specified slot, the YC serves the YT that arrived first. YCs handle only one container at a time. Once given, YC instructions are not changed. Examples of YC instructions are "slot 41, cross gantry to slot 419′ (case 1), "slot 489, R′ (case 2), and "slot 196, idle" (case 3). The states that a YC passes through for the 3 cases are shown in Fig. 4.

Within the simulation model, the "YC cross-gantry dispatching algorithm," "YC job dispatching algorithm," and "YC gantry-only dispatching algorithm" are responsible for generating the instructions corresponding to cases 1, 2, and 3 (see Table 1). In this research, we evaluate various alternatives for the YC job dispatching algorithm. These variants are plugged into the core YC control system to be tested and compared.

In the following paragraphs, we discuss both *rule-based* and *look-ahead* versions of the YC job dispatching algorithm. Rule-based algorithms select the YC's next job according to a simple, myopic rule. Look-ahead algorithms, in contrast, select the YC's next job by first solving an integer programming problem that considers how one or more YCs can "best" complete a set of jobs over a planning horizon.

4.2 Rule-based YC dispatching

The rule-based YC dispatching algorithms work as follows. Upon activation, these algorithms first identify the *feasible region* for the YC's next job. The feasible region is a contiguous area that extends left and right from the YC's current slot to include

Table 1 Discrete events in the simulation model

Primary events	Secondary events
Vessel arrives	Vessel starts berthing
Vessel finishes berthing	Vessel starts un-berthing
Vessel finishes un-berthing	QC job sequences generated
All QC job sequences scanned	QC job sequences erased
QC finishes handling	QC starts handling
YC cross-gantry dispatching algorithm called	General storage and retrieval algorithm called
YC finishes cross gantry	Real-time container storage algorithm called
YC finishes linear gantry	YC cross gantry scoring re-computed
YC finishes handling	YC gantry-only dispatching algorithm called
Workcenter YT assignment algorithm called	YC job dispatching algorithm called
YT finishes journey	YC starts cross gantry
Check terminal status consistency	YC starts linear gantry
Start data collection	YC starts handling
	Individual YT dispatching algorithm called
	YT starts journey

all slots in the same zone that are at least eight slots (i.e., 170 feet) away from the YC's neighboring cranes. The YC can be dispatched to any slot in the feasible region without violating the minimum separation requirement (Fig. 3). A list of all container moves (jobs) assigned to YTs that are already waiting in the YC's feasible region or are traveling towards the YC's feasible region is then constructed, and the algorithm selects one job in the list as the next job handled by the YC.

Selecting the next job handled by a YC presents several competing trade-offs. On one hand, proximity might be emphasized over urgency, so that nearby YTs/containers have priority over YTs/containers that are farther away from the YC. Such logic minimizes nonproductive YC and YT time in the short run. On the other hand, urgency might be emphasized over proximity, so that containers with earlier expected QC handling times have priority over those with later times. Such logic could potentially reduce QC idle time at the cost of less productive YCs and YTs in the short run. Finally, a compromise between proximity and urgency could be made, allowing both objectives to be pursued to a lesser degree. These considerations were incorporated into the various algorithms investigated in the experiments.

Table 2 shows the twelve rule-based YC dispatching algorithms considered in the experiments. In all 12 algorithms, the jobs in the YC's current slot have priority over all other jobs. If there are no jobs to be handled in the YC's current slot, the next job handled by the YC is selected as follows. In algorithm 1, the YC serves the first YT that arrived (or is expected to arrive) in the feasible region. In algorithm 2, the YC serves the earliest arriving YT first, but YTs that are retrieving containers from the yard (that are assigned 'R' jobs) are given priority over the YTs that are bringing containers to be stored ('S') in the yard. Algorithm 3 is the same as algorithm 2 except the 'R' and 'S' priorities are switched. In algorithm 4, the YC serves the nearest YT first. Algorithm

Table 2 Rule-based YC dispatching algorithms considered in the experiments

Algorithm	Description
1	Earliest YT to arrive
2	Earliest YT with ‘R’ job; otherwise earliest YT with ‘S’ job
3	Earliest YT with ‘S’ job; otherwise earliest YT with ‘R’ job
4	Nearest YT
5	Nearest YT with ‘R’ job; otherwise nearest YT with ‘S’ job
6	Nearest YT with ‘S’ job; otherwise nearest YT with ‘R’ job
7	Most urgent job
8	Most urgent ‘R’ job; otherwise most urgent ‘S’ job
9	Most urgent ‘S’ job; otherwise most urgent ‘R’ job
10	Most urgent ‘R’ job; otherwise nearest ‘S’ job
11	Opposite of 10 (farthest ‘S’ job; otherwise least urgent ‘R’ job)
12	Nearest YT to left (right) if currently moving left (right); otherwise change dir.

5 (6) is the same as algorithm 4 except that YTs with ‘R’ (‘S’) jobs are served first. In algorithm 7, the YC is instructed to handle the “most urgent” job (i.e., the job with the earliest expected QC handling start time). Algorithm 8 (9) is the same as algorithm 7 except that YTs with ‘R’ (‘S’) jobs are served first. In algorithm 10, the YC serves the most urgent ‘R’ job first. If there are no such jobs, the YC handles the nearest ‘S’ job. Algorithm 11 is the exact opposite of algorithm 10; the YC serves the farthest ‘S’ job first, and if there are no such jobs, the YC handles the least urgent ‘R’ job. Algorithm 10 turns out to be one of the best algorithms, so algorithm 11 represents a kind of “worst case scenario” where YCs are dispatched by an idiot. In algorithm 12, the YC serves the nearest YT to its left (right) if it is currently moving left (right). The YC changes direction only when there are no more jobs to be handled in the forward direction. The performance of these twelve algorithms is compared in Sect. 6.

4.3 Look-ahead YC dispatching and YC scheduling

Look-ahead YC dispatching algorithms typically use more information to select a YC’s next job than rule-based algorithms. Such additional information might include the set of upcoming, non-immediate jobs near a YC and the status of other YCs in the same block or zone. In this research, we developed a look-ahead YC dispatching model based upon the idea of *YC scheduling*. YC scheduling is a way of assigning a sequence of tasks to a YC, the first of which is the next task undertaken by the YC. The goal is to devise a plan for how a set of YCs in the same zone can work together to handle a set of jobs over a given planning horizon. Thus, two or more YCs are typically considered when a YC schedule is constructed.

To illustrate YC scheduling principles, consider YCs 11 and 12 in Fig. 2 and assume these YCs are confined to block 6. The upcoming workload in block 6 is obtained by translating the QC job sequences into a *yard work list* (YWL). The YWL is a list of the individual container handling tasks (container moves) expected to occur in the yard within the next 2 h, along with their yard stack locations and expected handling times. In translating the QC job sequences into a YWL, we make use of readily available data on average container handling time for QCs and YCs and average YT travel time between pairs of locations in the terminal, and we also assume that the YTs are always on hand to receive containers from YCs and QCs. The YWL is then used to construct a *job diagram* that visually depicts the upcoming workload in block 6.

Two examples of job diagrams for block 6 are given in Figs. 5 and 6 (left). In these diagrams, each container move is represented by a rectangle that is located in space and time according to its slot location, expected YC handling start time, and expected YC handling finish time as listed in the YWL. The horizontal axis corresponds to the slot where the move takes place; the vertical axis corresponds to the time when the move is expected to occur. The center of each rectangle is located at the coordinates (slot, (expected YC start time + expected YC finish time)/2). Each rectangle is 3 min high—a conservative estimate of the time required to handle a container—and eight slots wide, which is equal to the required separation distance between YCs. The planning horizon for these job diagrams is roughly 2 h. Note that the workload in both diagrams is not well distributed; such is typically the case at a real facility. Also, there are several clusters of overlapping rectangles in both diagrams, indicating that it is not feasible for these container moves to be made at the times shown.

YC scheduling is a method for sorting out the confusion in a job diagram. In a YC schedule, the start/finish time of each container move is modified, and each container move is assigned to a YC, so that all moves can be "feasibly" completed by the YCs on hand with minimum earliness + tardiness. From a YC's point of view, the YC schedule specifies the sequence of activities the YC undertakes over the planning horizon—which containers it handles, and when. Examples of YC schedules are given beside the job diagrams in Figs. 5 and 6 (right). In these YC schedules, each container move is again represented by a rectangle. In Fig. 5, the 'R' and 'S' labels stand for retrieval and storage moves respectively, and the shading of the rectangles indicates which YC performs the container move. In Fig. 6, the numbers inside the clear rectangles indicate which YC handles which container. The job diagrams have been reproduced in the background behind the YC schedules to allow comparison of the input and output times for each container move. In Fig. 5, for example, the storage move at slot 2, which has start time 48 in the job diagram, is scheduled to be handled by YC 11 at time 66 (see annotation in the center of Fig. 5). Thus, the YT that brings this container to the yard will wait at slot 2 for about 18 min before being served by YC 11. Also, the long diagonal arrow in Fig. 6 shows a retrieval job in slot 50 with start time 55 in the job diagram and start time 21 in the YC schedule. Finally, the jagged arrow in Fig. 6 shows a storage job whose output handling time is roughly 2 h after its input handling time.

Job diagrams and YC schedules are used for real-time YC dispatching as follows. Assume that YC 11 has just become free at time 20 and needs to be given new instructions. The "YC job dispatching algorithm" is then called (Table 1) and the aforementioned YC scheduling procedure is initiated. The output from the procedure is a YC schedule for block 6 (Figs. 5, 6, right), and YC 11 is immediately dispatched to handle the next container listed in its job sequence. The entire process is very

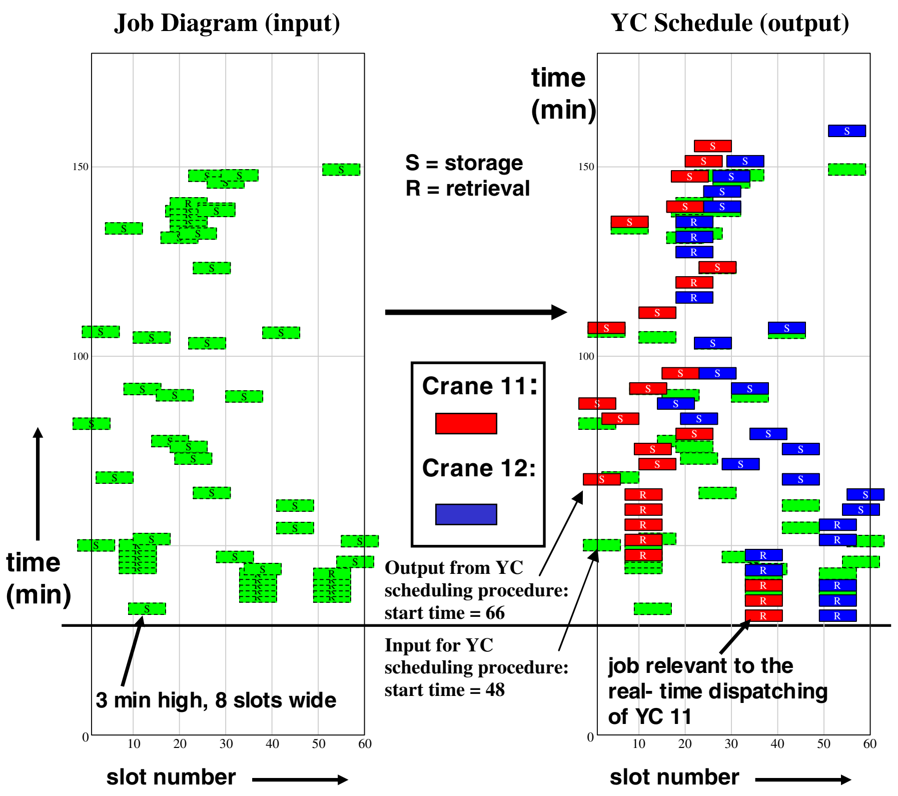

Fig. 5 Job diagram (*left*) showing the expected times and locations of 53 upcoming container moves (*dashed rectangles*) in a yard block over a 2-h time horizon, and YC schedule (*right*) showing how two YCs, '11' and '12', are scheduled to handle this workload (*solid rectangles*). Note that the *solid rectangles* are non-overlapping

complicated, but in the end, the only information relevant to the real-time dispatching of YC 11 is the first job assigned to YC 11 in the YC schedule.

4.4 Integer programming models for YC scheduling

Several integer programming (IP) models for scheduling YCs in a particular zone are described in Petering (2007). The two models that appear most promising in the context of real-time control are discussed in Petering and Murty (2006) and Petering et al. (2006). Preliminary experiments indicate that, depending on the IP model and job diagram, ILOG CPLEX takes from 1 to 4,000 s to find an optimal solution to problems whose size is comparable to those depicted in Figs. 5 and 6. This lengthy runtime has prompted the authors to search for heuristic algorithms for generating near-optimal YC schedules more quickly (Li et al. 2008). However, as we discuss below, the prospects for meaningfully embedding such heuristics within a real-time, look-ahead YC dispatching system remain doubtful.

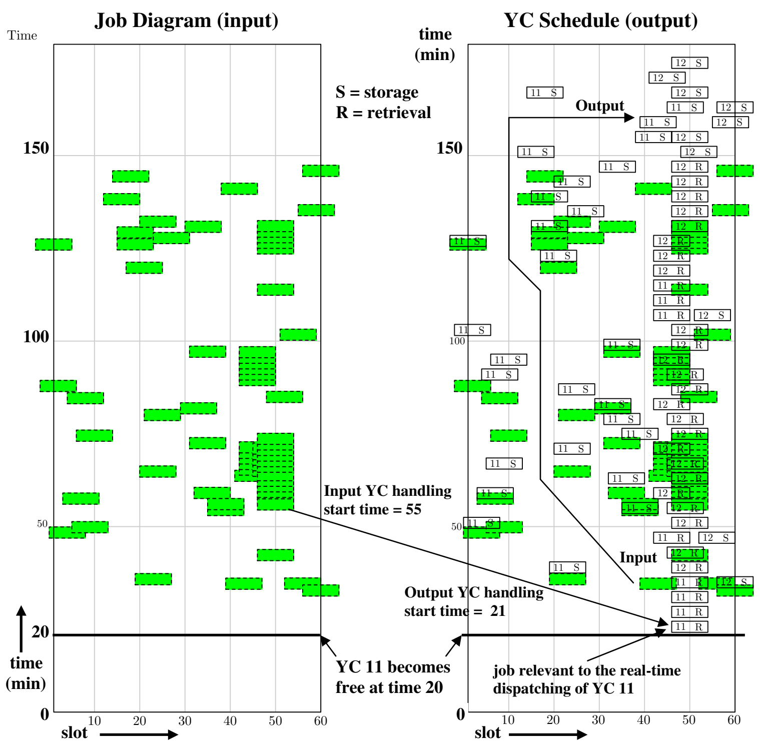

Fig. 6 A YC scheduling problem (*left*) whose optimal solution (*right*) requires the immediate retrieval at time 21 of a container (indicated by the *long diagonal arrow*) whose original, input yard handling time was time 55

4.5 Look-ahead YC dispatching and deadlocking

Deadlocking casts a long shadow over the development of real-time, automated control systems for RTGC-based container terminals because these facilities have no inter-equipment buffers and the operations exhibit a high degree of stochasticity. Our efforts to develop real-time YC dispatching systems were highly influenced by deadlocking considerations. In this section, we show that look-ahead YC dispatching systems that use YC schedules with long planning horizons are susceptible to deadlocking in a real-time setting. In order to avoid deadlocks, the planning horizon must be shortened. However, the shortening of the planning horizon significantly reduces the number of container moves considered in the job diagram, resulting in greatly simplified YC scheduling problems whose solutions are trivial.

Let us recall the real-time conditions that give rise to the job diagram depicted in Fig. 6 (left). This diagram is reproduced in Fig. 7 (left). This diagram is generated when YC 11 in block 6 in Fig. 2 finishes a task at time 20 and requests a new set of instructions. The job diagram presents an interesting and challenging YC scheduling problem consisting of 69 container moves expected to occur over a 2 h planning horizon. In theory, any of these 69 moves could be the first one assigned to YC 11 by

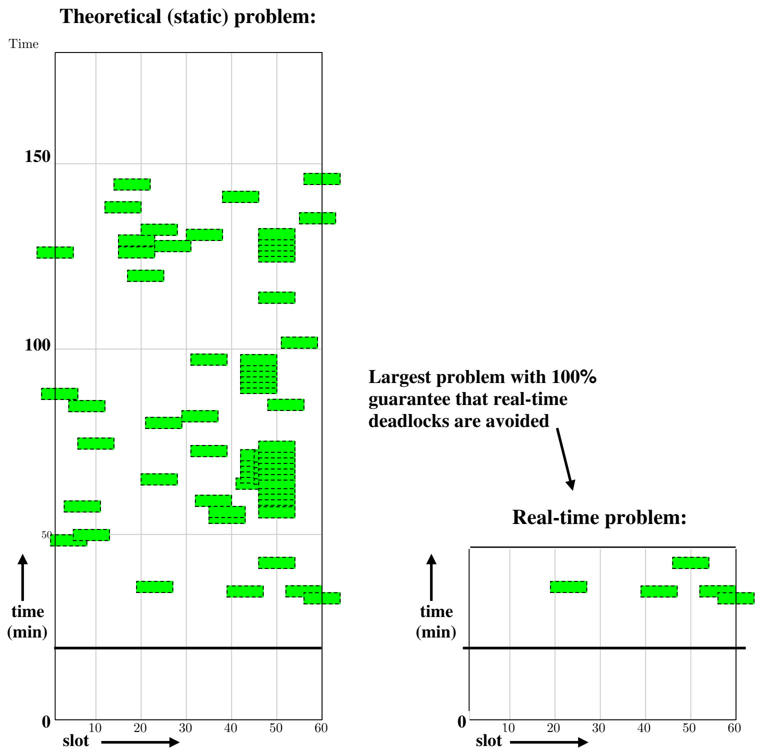

Fig. 7 In a theoretical YC scheduling problem (*left*), several dozen container moves are considered and an interesting integer programming problem is solved. However, this problem must be greatly simplified in a real-time setting to avoid operational deadlocks (*right*)

the YC scheduling procedure. Thus, it is possible for a container move whose handling starts after time 50 in the job diagram to be scheduled first for YC 11. In fact, looking at the long diagonal arrow in Fig. 6, we see this is indeed the case; the first job in YC 11's schedule is the retrieval move in slot 50 that has start time 55 in the job diagram. In a real-time setting, YC 11 is therefore instructed to (1) gantry to slot 50, (2) retrieve a container, and then (3) pass the container to a YT.

However, it is highly unlikely that any YT will arrive at slot 50 in the near future. In fact, at time 20, it is very likely that no YT has received instructions to travel to slot 50. This is because the YTs must first handle the jobs that are more urgent from the perspective of the QCs; otherwise the quay operations will be deadlocked. For example, at time 20 there might be 5 YTs assigned to the 5 storage jobs at the bottom of the job diagram (Fig. 7, right) but no YTs assigned to the retrieval job in slot 50. Moreover, these 5 YTs might already be waiting at (or at least be heading towards) block 6. However, YC 11 has not been instructed to serve any of these YTs. This situation is dangerous and can potentially lead to a deadlock. A deadlock looms because there is a mismatch in YC and YT job assignments and YC instructions cannot be changed once they are given (Sect. 4.1). This "job assignment deadlock" threatens to bring all operational progress in the terminal to a halt. Indeed, several of our preliminary experiments resulted in such deadlocks.

The above scenario teaches us the following lesson. To avoid deadlocks on 100% of occasions, in YC scheduling problems we should only consider jobs that have already been assigned to YTs that are either (1) already waiting in the yard or (2) traveling towards the yard. In other words, we should only consider the container moves for which a corresponding YT is guaranteed to show up in the yard. However, the number of container moves satisfying this criterion in real-time is typically very small (Fig. 7, right). Indeed, the ratio of YTs to YCs at a container terminal rarely exceeds 3. Therefore, at any given moment, at most 3 container moves can be satisfying the criterion per YC deployed. Furthermore, in a typical real-time situation, only half of the YTs are waiting at or heading towards the yard; the other half are waiting at or heading towards the quay. Thus, in a real-time setting, the job diagram for an average YC scheduling problem has no more than 1.5 container moves per YC. In other words, to avoid deadlocks in a real-time setting, YC scheduling problems must be made so trivial that they resemble rule-based dispatching problems! Our experimentation, described in Sect. 6, is therefore limited to the rule-based algorithms introduced in Sect. 4.2.

5 Simulation model of a seaport container transshipment terminal

5.1 Introduction

We now present a discrete event simulation model of operations inside a seaport container transshipment terminal. The model has been developed based on the authors' combined experience at several container terminals and on extended discussions with managers and staff members at these terminals. The model is noteworthy in that (A) it allows direct comparisons to be made between alternate terminal designs and real-time yard control systems at a multiple-berth facility; (B) it measures performance according to the industry standard metric, GCR; (C) it considers all major entity types—vessels, QCs, YCs, and YTs; (D) all cranes and trucks have stochastic handling and traveling times; (E) delays are propagated realistically from one part of the system to another; and (F) the arrival, stay, and departure of every individual container is explicitly modeled. The model simulates the activities associated with individual containers, vessels, QCs, YCs, YTs, and groundslots in the storage yard over an arbitrarily long, user-defined time period (e.g., 3 weeks, 6 months) to a level of detail indicated by the list of events in Table 1.

Before continuing, we first discuss our motivation for using a simulation methodology in this research. The methodologies of operations research fall into three main categories: deterministic optimization, stochastic modeling, and simulation. Deterministic optimization is the dominant methodology in the majority of articles on container shipping. Yet, as mentioned above, container terminal operations exhibit a high degree of stochasticity. Plans and schedules for detailed yard operations that are based on integer programming (IP) models become out-of-date almost immediately after their construction. Reconstructing such schedules in real time using the latest information is often not practical because runtimes for IP heuristics are usually too long to be viable in a real-time environment. Even if an IP heuristic has a fast runtime, it is often susceptible to deadlocking. The only way to demonstrate an IP heuristic's viability in a real-time environment is to embed it within a stochastic or simulation model. Therefore, the methodology which provides the ultimate justification and validation of a model for real-time, micro-decision making should be stochastic in nature.

Stochastic modeling, in the absence of simulation, has been extensively used to obtain insights on systems where the state space is relatively small, transitions between states exhibit a regular structure, and/or transition times between states have special properties (e.g., they are exponentially distributed). Container terminals, however, are massive facilities with hundreds of trucks and cranes that can each be in a dozen or more states, and thousands of yard locations that can each be in hundreds of different states. In addition, the transition times between states at a container terminal do not exhibit "nice" properties. Overall, simulation is the only methodology that captures the stochasticity of real-time container yard operations while handling a massive state space. This is the motivation behind our use of simulation in this research.

5.2 Events

Table 1 shows the events that occur within the simulation model. Overall, there are two kinds of events: primary and secondary. Only primary events may appear in the future event list (FEL). A primary event generally (1) causes an immediate change in the state of the container terminal, (2) triggers other events, some primary and some secondary, that occur together with the primary event, and/or (3) causes other primary events to be placed at some future time in the FEL. The event scheduling/time advance algorithm ensures that all events take place in proper chronological order. Note that several events listed in Table 1 are calls to algorithms that make container storage, YC deployment, YC dispatching, and YT dispatching decisions in real time.

5.3 Main features and limitations of simulation model

A complete description of the simulation model is provided in Petering (2007). A less detailed description is available in Petering and Murty (2008). The model has several important features. First of all, it has built-in systems for selecting container storage locations in real time; for deploying YCs among yard zones and initiating YC cross-gantry moves in real time; and for dispatching YTs in real time. These systems are discussed in Petering and Murty (2007), Petering and Murty (2008), and Petering (2007), respectively.

The model accommodates two container sizes ($20'$ and $40'$). YTs are allowed to haul two $20'$ containers at the same time. In other words, YTs have dual-load capability. Although the total number of YTs is fixed throughout each simulation run, the number of active YTs is a fixed multiple of the number of busy berths at all times. Inactive YTs are only reactivated when a new vessel starts to dock at the terminal. The expected YT travel time between two locations in the terminal is calculated based on (A) the length of the shortest route along terminal roadways that connects the origin and destination and adheres to traffic rules (e.g., one-way restrictions along the narrow horizontal avenues between storage blocks), (B) the number of turns involved in the route selected in part A, and (C) whether the YT is empty or laden. The actual YT travel time between these locations is computed by adding a noise factor to the expected travel time.

QCs handle one container at a time. Each QC handles containers according to a sequence with limited flexibility for changing the order in which lifts are performed. This means that during vessel loading, YTs arriving beneath a QC may not be served until the YTs assigned to most of the QC's earlier jobs have already passed the appropriate containers to the QC. Thus, a poorly sequenced arrival of YTs beneath a QC that is loading a vessel can significantly harm GCR as measured by the simulation model.

As mentioned earlier, the YC control system automatically prevents YC-YC interference. In particular, YCs are not allowed to be closer than 8 slots = 170 feet apart at any time (Fig. 3). This YC-YC interference prevention system ensures that performance, as measured by the simulation model, deteriorates as additional YCs are added to a terminal where YCs are already plentiful. The model also assumes that there are bypass lanes, such as that depicted in Fig. 3, for trucks along the horizontal roadways between the storage blocks. Bypass lanes allow YTs to reach more than one YC in a block simultaneously without creating a traffic jam.

The traveling and handling times for all machines (QCs, YTs, YCs) follow probability distributions that are user-defined. The distributions used in the experiments are given in Sect. 6. The transfer of individual containers between cranes and trucks is explicitly modeled. During vessel loading, for example, the event "QC starts handling" (Table 1) is only triggered if (1) the QC has just finished loading a container and the next container is on a YT that is waiting beneath the QC or (2) the YT bringing the next container in the sequence has just arrived beneath the QC. In other words, the event "QC starts handling" can only be triggered by the events "QC finishes handling" or "YT finishes journey," subject to many restrictions.

The model also tracks the groundslots where individual containers are stored when they are sitting in the storage yard. Containers are assigned storage locations immediately after being placed onto YTs during unloading. Tracking individual groundslots and storage locations is important for several reasons. First of all, these locations affect the journey times of YTs that are transporting cargo between the yard and quay. They also affect the times taken by YCs to gantry between jobs. Most importantly, they affect the performance of YCs that work in close proximity through the 170-foot (8-slot) separation requirement (Fig. 3). Each container that is stored in the yard is retrieved from the same location before being loaded onto a vessel. Vessels may not leave the terminal unless all of their containers have been unloaded and loaded.

The model's main limitations are as follows. Firstly, only $20'$ and $40'$ standard dry containers are considered; 45-foot long, refrigerated, dangerous goods (DG), and other kinds of containers that make up about 15% of overall cargo volume are ignored. Secondly, YT congestion on the roads and at handling points is not explicitly modeled. This means that, barring a probabilistic miracle, adding more YTs to a scenario *never* erodes the observed performance of the system. To the authors' knowledge, this shortcoming is found within all other simulation models in the literature.

The most important limitation is that the model considers a pure transshipment terminal. That is, all containers enter and leave the terminal by vessel. In 2006, the world's busiest container port, Singapore, was primarily a transshipment port, with roughly 80% of cargo being transferred from one vessel to another. In addition, at least two other ports in the top 10—Kaohsiung and Dubai—had a significant proportion ($>40\%$) of transshipment cargo.

The pure transshipment assumption leads to several other assumptions. Firstly, there is no gate or rail yard, and there are no XTs or trains. Thus, every container sitting in the yard is destined to be loaded onto a vessel. In other words, the entire storage yard is an export yard. In practice, containers in an export yard are typically stored according to a few important categories such as length, height, weight class, loading vessel, and destination port. Containers having identical status in all five categories are substitutable for vessel loading purposes. Such containers are considered to be in the same *group* and are usually stored according to a *homogenous stacking policy*. This policy requires the contents in each stack in the yard to be homogenous; in other words, all containers in a stack must belong to the same group. Such a policy improves YC performance because it eliminates the need for YCs to shift containers between stacks in order to dig out containers at lower levels. Homogenous stacking policies are commonly adopted in export yards at terminals throughout the world, and we enforce such a policy in the simulation model. Thus, we also assume that YCs do not shift containers directly between stacks.

Although it considers a pure transshipment facility, the model captures many important features of non-transshipment (import/export) terminals. For example, the vessel-to-vessel transshipment operations considered by the model are very similar to the export operations at non-transshipment terminals. This is because transshipment and export cargo both depart by vessel. Thus, the strategies for storing and loading such cargo are basically the same.

5.4 Vessel schedules and initialization

The terminal operates according to a regular weekly schedule. That is, the terminal sees one vessel from a particular liner service at more-or-less the same time each week. The vessels visiting the terminal each week are assumed to be following the same routes as their respective counterparts from the previous week. Most major container terminals operate in this manner.

The terminal's weekly vessel schedule is generated at the beginning of each simulation run. This schedule consists of a home berth assignment and scheduled weekly arrival time for each liner service. Home berths are assigned so the liner services are evenly distributed among the berths. After assigning home berths, the model randomly generates each liner service's scheduled weekly arrival time, under the constraint that the vessels visiting each berth should be present during non-overlapping time intervals.

The actual arrival time of a vessel during the course of the simulation may deviate from the scheduled arrival time. If a vessel's home berth is available when it arrives, the vessel immediately begins to dock at its home berth. Otherwise, if another berth is available, the vessel proceeds to the berth that is closest to its home berth. If no berth is available, the vessel drops anchor in the harbor and joins the end of the queue of vessels that are waiting for a berth.

The container yard is empty at the beginning of each simulation run. During week 1, vessels bring in containers and the majority of container traffic flows from the quay to the yard. By the end of week 1, the yard becomes "filled up" and equilibrium between the quay-to-yard and yard-to-quay flows is established; during weeks 2 and later, these two flows balance out. Data collection starts at the beginning of week 2.

6 Experiments, results, and discussion

6.1 General setup for experiments

Sections 4 and 5 highlight the need for a dynamic and integrated simulation model of a container terminal to demonstrate the viability of, and evaluate the performance of, a proposed algorithm/method/model in a real-time environment. This research has produced such a simulation model. The simulation model was written and compiled using the Professional Edition of Microsoft Visual C++ 6.0. Experiments were run in the Windows XP environment using a 2.0 GHz Pentium 4 desktop computer with 512 MB of RAM.

In all simulation runs, data collection started at the beginning of week 2. The simulation terminated after 3 weeks worth of vessels were fully processed at the terminal. This instant may fall within week 4 or later if significant backlogging has occurred due to a lack of equipment or an inferior YC dispatching system.

The experiments considered two different terminals—a small terminal and a large terminal—and two different yard fleet size scenarios for each terminal—"less equipment" and "more equipment." Thus, a total of four different container terminal scenarios were considered. Table 3 gives the major specifications of these scenarios. In each scenario, all vessels are the same size and have equal expected cargo throughput each week, but the throughput associated with a particular vessel in a given week is a stochastic quantity. The physical layout of the small terminal is depicted in Fig. 8. This terminal has 5 yard zones, 4 blocks per zone, 42 slots per block, and 6 rows per block. The physical layout of the large terminal is depicted in Fig. 9. This terminal has 10 yard zones, 9 blocks per zone, 42 slots per block, and 6 rows per block. As Figs. 8 and 9 indicate, YTs may travel in both directions on all vertical lanes and on the two horizontal lanes at the top and bottom of the terminal, but may only travel in one direction along the horizontal lanes in the middle of the terminal.

In all experiments, GCR is measured as follows:

$$\text{GCR} = (\text{total number of QC lifts})/(\text{total number of QC hours beside a busy berth}).$$

A busy berth is a berth with a vessel alongside such that at least one QC is transferring containers between the vessel and the shore. The denominator is directly proportional to the average vessel turnaround time, so GCR is inversely proportional to the average vessel turnaround time.

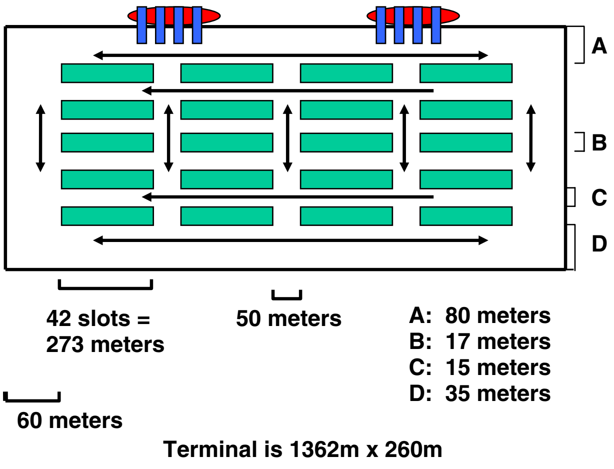

Fig. 8 Layout of the small container terminal considered in the experiments

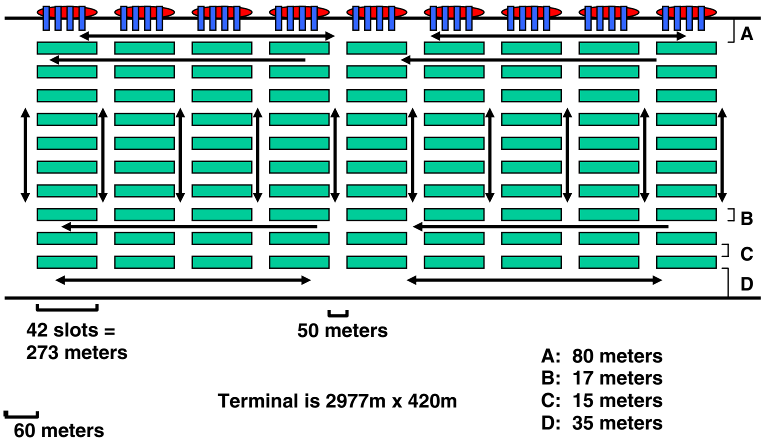

Fig. 9 Layout of the large container terminal considered in the experiments

6.2 Vessel, QC, YT, YC, and container storage settings

The values of the input parameters (except for the real-time algorithm settings) were chosen based upon information received from a major container terminal operator. In all experiments, vessels arrive anywhere from 2 h prior to 12 h after their scheduled arrival times. The time taken by a QC to handle a single container is a triangularly distributed random variable with parameters (1.0, 1.5, 2.0) min. This distribution has a mean of 1.5 min, so the long-run technical work rate of each QC, if it is never starved of YTs, is 40 lifts/h. This means that the maximum possible GCR in any experiment is 40 lifts/h.

Table 3 Fully-dynamic container terminal scenarios considered in the experiments

	Small terminal less equipment	Small terminal more equipment	Large terminal less equipment	Large terminal more equipment
Vessel calls per week	10	10	90	90
Expected QC lifts per vessel	3,600	3,600	1,920	1,920
Expected QC lifts per week	36,000	36,000	172,800	172,800
Berths	2	2	9	9
QCs	8	8	36	36
Groundslots in yard	5,040	5,040	22,680	22,680
Yard zones	5	5	10	10
Max. cont. stacking height in Yard	6	6	5	5
Ratio of 20′ to 40′ conts.	2:1	2:1	3:2	3:2
YCs	20	25	90	110
YTs	40	72	180	324
Active YTs per busy berth	20	36	20	36

In all experiments, YTs travel 40 km/h on average when empty and 25 km/h on average when carrying one or more containers. They spend an average of 10 s making a turn. Preliminary experiments, many of which are described in Petering (2007), allowed us to identify one YT dispatching system, out of several alternatives, for use in this paper. (A comparison of these YT dispatching systems is the subject of a future study.) This YT dispatching system performed well in a variety of situations. It allows YTs to carry two 20-foot containers simultaneously only if they have the same pick-up location (QC or stack in the yard). Whenever a YT becomes free, the system identifies the "most starved QC" and assigns to the YT the earliest container in that QC's job sequence that has not already been assigned to another YT. In other words, the YT serves the "most idle QC" first. This idea is essentially the same as the "inventory-based" AGV dispatching approach that was found to be effective in Briskorn et al. (2006). Two containers are assigned to the YT if the two earliest containers for the QC are both 20′ long and have the same pick-up location. As mentioned earlier, the number of active YTs is a fixed multiple of the number of busy berths at all times (see Table 3). Inactive YTs can only be reactivated when a new vessel starts to dock at the terminal.

In all experiments, YCs gantry at an average rate of one slot every four seconds. The time taken by a YC to handle a single container is a triangularly distributed random variable with parameters (1.2, 2.0, 3.4) min. YCs are not allowed to make inter-zone cross-gantry moves like that indicated by the arrow to the left of YC 24 in Fig. 2. They are allowed to move between blocks in the same zone. However, such intra-zone, inter-block movement is restricted to the extent possible in accordance with the findings in Petering and Murty (2008). In that study, it is shown that a "restrictive" YC deployment system, which sets a minimum and maximum number of YCs that must be present in each block at all times, outperforms a "free" system that does not include such restrictions.

Containers are assigned storage locations in real time immediately after being placed onto YTs during unloading. The experiments described in Petering and Murty (2007) allowed us to identify one container storage location assignment system, out of several alternatives, for use in the experiments in this paper. This system performed well in a variety of situations. Overall, the system tries to disperse both the containers unloaded by the same vessel and the containers loaded onto the same vessel throughout the storage yard to make sure that several YCs are supporting each working QC during vessel unloading and loading. This helps to minimize the QC idle time during both these operations. The dispersion of containers also helps to equalize the YCs' workloads at all times.

The container storage location assignment system works as follows. Immediately after a QC finishes placing a container onto a YT, the system scans the entire storage yard to see if there is a stack in the yard with available capacity that already stores containers belonging to the same group to which the container belongs. If one or more such stacks are found, the system assigns the container to one of these stacks. Otherwise, the system assigns the container to an empty stack in the yard that is in the block with the lowest combined (1) number of YTs heading to or already waiting at the block and (2) number of container retrievals that are expected to coincide with (clash with) the retrieval of the current container if the container were placed in the block. The overall decision is made by weighting these two measures. Put another way, the container storage system tries to strike a balance between (1) minimizing the delay/congestion associated with storing the container in the yard and (2) minimizing the delay/congestion associated with retrieving the container from the yard when the time comes to retrieve it.

6.3 YT availability criteria

Besides investigating the performance of different rule-based YC dispatching algorithms, the experiments also evaluate various YT availability criteria within the YC dispatching system. Overall, four different YT availability criteria are investigated. These criteria correspond to the four possible combinations for (1) whether or not YTs with storage jobs that are traveling towards the yard (i.e., TS-YTs, traveling-storage-YTs) are "available" to be served by the YCs and (2) whether or not YTs with retrieval jobs that are traveling towards the yard (i.e., TR-YTs, traveling-retrieval-YTs) are "available" to be served by the YCs. In all cases, YTs that have already arrived in the yard are always eligible to be served by YCs.

The YT availability criteria are relevant for terminals that have installed sensors at the blocks. These sensors detect when YTs heading towards the yard have arrived at their destination blocks. Such information could be used by the YC dispatching algorithm to narrow down the set of YTs that are eligible to be immediately served by a YC, which may lead to a higher GCR. For example, it may be more efficient for YCs to only serve YTs that have already arrived at their yard destinations and to ignore YTs that are still traveling towards the yard. The default scenario has no sensors at the blocks; all YTs that are either traveling towards the yard or are already waiting in the yard are eligible to be served by a YC.

6.4 Results and discussion

Our experiments considered all possible combinations of the twelve rule-based YC dispatching algorithms (Table 2), four YT availability criteria (Sect. 6.3), and four container terminal scenarios (Table 3). Thus, $12 \times 4 \times 4 = 192$ different setups were investigated.

The overall results are displayed in Table 4. Each number in the table is the average GCR observed in six probabilistically identical simulation runs, where the data collection period in each run is at least 2 weeks. Thus, the table includes results from 1,152 individual experiments. More than 90,000 and 440,000 QC lifts occur in each experiment for the small and large terminal scenarios respectively, so Table 4 is based upon more than $576 * 90,000 + 576 * 440,000 = 305$ million QC lifts worth of simulated activity. No deadlocks or errors occurred during any of the experiments. The CPU runtime for every individual experiment at the small (large) terminal was less than 2 (22) min. The four highest GCR values for each container terminal scenario are highlighted in bold.

The final row of Table 4 shows the average GCR for the different YT availability criteria at each of the four container terminal scenarios. For each terminal scenario, the two highest GCR averages are highlighted in bold. These averages strongly indicate that the YTs with ‘R’ jobs that are traveling towards the yard (the TR-YTs) should be considered “available” by the YC dispatching system. Indeed, for each terminal scenario, the GCR observed for any instance when the TR-YTs are available is at least 1.1 lifts/h greater than the GCR for any instance when these YTs are not available. On the other hand, the results regarding the availability status of the TS-YTs are inconclusive. In other words, it does not seem to matter whether or not the YTs with ‘S’ jobs that are traveling towards the yard are considered “available” by the YC dispatching system. These results are intuitively plausible. Indeed, since the container transfer between YC and YT for ‘R’ jobs occurs at the end of the YC’s handling of the container (which lasts 2.2 min on average), it is crucial for the YCs to get an early start retrieving containers; otherwise the lower bound for the average waiting time in the yard for YTs with ‘R’ jobs is 2.2 min. On the other hand, the container transfer between YC and YT for ‘S’ jobs occurs at the beginning of the YC’s handling of the container, so the lower bound for the waiting time in the yard for YTs with ‘S’ jobs is still zero even if the YCs are not aware they are approaching.

The final column of Table 4 displays the average GCR for each rule-based YC dispatching algorithm. The average combines all four terminal scenarios and the instances in which the TR-YTs are available; the other half of the instances, in which the TR-YTs are not available, are removed from consideration due to their inferior GCR. The highest two such averages are highlighted in bold. Note in this final column that the algorithms that prioritize the retrieval of containers from the stacks are outperforming the algorithms that use the opposite priority or that do not differentiate between ‘R’ and ‘S’ jobs. In fact, this trend is visible without exception in each container terminal scenario for the experiments in which the TR-YTs are considered to be available. Indeed, in all four scenarios, algorithm 2 performs the best among algorithms 1, 2, and 3; algorithm 5 performs the best among algorithms 4, 5, and 6; and algorithm 8 performs the best among algorithms 7, 8, and 9.

Table 4 GCR results from the experiments (QC lifts/h)

	YC dispatching algorithm	TS-YTs avail?	TR-YTs avail?								Avg. for TR-YTs avail. = Yes
			Small term less equipment		Small term more equipment		Large term less equipment		Large term more equipment		
			Y	N	Y	N	Y	N	Y	N	
1	Earliest YT	Y	28.81	27.18	35.57	34.51	27.48	26.20	35.20	34.08	31.77
		N	28.89	27.41	35.53	34.67	27.50	26.18	35.16	33.99	
2	Earliest YT-R	Y	29.84	27.50	36.42	35.22	28.05	26.46	36.43	35.06	32.65
		N	29.85	27.57	36.13	35.12	28.00	26.39	36.44	34.93	
3	Earliest YT-S	Y	28.02	27.01	34.23	33.66	26.20	24.93	32.17	30.98	30.16
		N	28.06	26.82	34.28	33.36	26.14	25.14	32.16	30.96	
4	Nearest YT	Y	30.83	28.36	36.15	34.84	28.22	26.48	35.53	33.91	32.74
		N	31.01	28.22	36.39	34.68	28.19	26.47	35.56	33.91	
5	Nearest YT-R	Y	**31.11**	28.42	36.77	35.57	**28.70**	26.93	**36.87**	35.16	**33.38**
		N	**31.23**	28.28	36.80	35.23	**28.70**	26.95	**36.90**	35.15	
6	Nearest YT-S	Y	29.55	27.92	34.96	33.73	26.71	25.37	32.19	30.95	30.79
		N	29.45	27.32	34.77	33.80	26.48	25.40	32.18	30.95	

Table 4 continued

	YC dispatching algorithm	TS-YTs avail?	TR-YTs avail?								Avg. for TR-YTs avail. = Yes
			Small term less equipment		Small term more equipment		Large term less equipment		Large term more equipment		
			Y	N	Y	N	Y	N	Y	N	
7	Urgent	Y	30.45	27.92	36.68	35.32	28.04	26.17	36.05	34.63	32.79
		N	30.32	28.00	36.72	35.65	28.04	26.13	36.00	34.68	
8	Urgent-R	Y	30.44	28.01	**37.07**	35.84	28.15	26.20	36.69	35.13	33.08
		N	30.53	27.90	**36.93**	35.68	28.15	26.19	36.69	35.12	
9	Urgent-S	Y	28.44	26.42	35.41	34.01	25.53	24.25	32.43	31.02	30.43
		N	28.45	26.36	35.24	33.95	25.54	24.28	32.42	31.03	
10	Urgent-R /Near-S	Y	**31.07**	28.47	**37.27**	35.86	**28.25**	26.29	**36.75**	35.18	**33.33**
		N	**31.14**	28.34	**37.20**	35.88	**28.23**	26.32	**36.72**	35.17	
11	Opposite of 10	Y	27.09	25.25	33.19	32.03	25.45	24.34	31.03	30.36	29.19
		N	27.04	25.42	33.04	32.09	25.50	24.35	31.17	30.50	
12	Nearest YT on left/right	Y	30.60	27.77	36.46	34.50	28.05	26.19	35.38	33.37	32.57
		N	30.72	27.44	36.08	34.39	27.98	26.26	35.29	33.38	
Average		Y	**29.69**	27.52	**35.85**	34.59	**27.40**	25.82	**34.73**	33.32	
		N	**29.72**	27.42	**35.76**	34.54	**27.37**	25.84	**34.72**	33.31	

Bold values indicate the highest values in their respective groups

Overall, the best performing algorithms are 5 and 10; algorithm 5 is the best in three of the four terminal scenarios, and algorithm 10 is the best at the small terminal with more equipment. The GCR differences between algorithms 5 and 10 are statistically significant at the 40% confidence level for the small terminal with less equipment and at the 99% confidence level for the other three terminal scenarios. Both of these algorithms give priority to the retrieval of containers from the stacks. Algorithm 5 is probably the best overall, whereas algorithm 10 is the most consistent performer, being the only algorithm that is among the best two algorithms in all four terminal scenarios. Note that both algorithms 5 and 10 rely on priority rules that are so simple that they can easily be understood by YC operators. Thus, even at terminals where there is no computerized information system, managers can still boost GCR simply by telling their YC operators which priority rules are the best. On another note, it is not surprising to see that algorithm 11 performs the worst in every terminal scenario. In particular, this algorithm performs between 11 and 16% worse than the best algorithm in each scenario. Overall, these results show for the first time that tens of thousands of real-time YC dispatching decisions have a major impact on the overall, long-run productivity of a container terminal.

Tables 5, 6, 7 and 8 give a more detailed account of the experiments in which the TR-YTs are considered to be available. Each row in these tables corresponds to two GCR values in Table 4 and is obtained by averaging the results from the twelve experiments for which the TR-YTs are available. (Six of these experiments consider the TS-YTs to be available; six consider the TS-YTs as unavailable.) Twenty different performance measures are displayed in these tables. These include the GCR, berth-on-arrival rate, average YC productivity, average YT productivity, and several other performance indicators. A key to these measures is provided beneath Table 5. The two highest GCR values in each table are highlighted in bold.

In Tables 5, 6, 7 and 8, note that YT productivity is highly correlated with GCR; YC productivity, on the other hand is somewhat less correlated with GCR. Also, note that the numbers in the "QC waitL" column are much larger than those in the "QC waitU" column. This indicates that most QC delays occur during loading, not unloading. This is not surprising given the fact that YTs bringing containers to the quay during loading are typically not substitutable, whereas YTs bringing their empty trailers to the quay during unloading are substitutable. The above phenomena, which have been observed by container terminal personnel known to the authors, help to validate the simulation model. The fact that most QC delays occur during loading helps to explain the success of those algorithms—particularly algorithms 5 and 10—that prioritize the retrieval of containers from the yard. Such algorithms are effective because they prioritize the loading operations that give rise to the lion's share of QC delays.

The "Berth Occ" and "%area" columns reveal relatively busy conditions in all four container terminal scenarios. Indeed, the average berth occupancy ranges from 74 to 100% depending on the terminal scenario and the YC dispatching algorithm that is used. In addition, roughly 85% of the storage area in the yard is occupied at any given instant in each scenario. The "Gant" column shows that the amount of YC gantrying done when algorithms 4–6 are used is less than the amount of YC gantrying done when any of the algorithms 1–3 or 7–9 are used for all terminal scenarios. This is expected, because algorithms 4–6 by definition are sending the YCs to the nearest jobs. Also,

Table 5 Detailed results for the small terminal with less equipment

YC disp. algorithm	GCR	Berth occ	# Vess arrived	BOA rate	%vol	%area	QCLifts	QC waitU	QC waitL	YCLifts	YC prod	Gant	YTHauls	Dual load	YT prod	YT waitU	YT waitS	YT waitR	YT waitL	YT wait
1	28.85	0.93	21.5	0.36	45	87	77,707	0.011	0.209	77,720	10.45	0.124	77,715	23,833	5.59	0.158	0.131	0.237	0.164	0.690
2	29.85	0.90	21.3	0.45	45	87	76,053	0.009	0.184	76,051	10.40	0.123	76,049	23,278	5.77	0.147	0.164	0.205	0.140	0.656
3	28.04	0.95	22.0	0.32	45	87	77,550	0.011	0.230	77,590	10.37	0.124	77,576	23,778	5.44	0.174	0.099	0.259	0.177	0.709
4	30.92	0.88	21.3	0.58	44	86	76,341	0.005	0.160	76,356	10.50	0.109	76,349	23,415	5.97	0.167	0.118	0.216	0.129	0.630
5	**31.17**	0.87	21.3	0.66	44	86	75,855	0.008	0.152	75,858	10.51	0.108	75,856	23,232	6.01	0.142	0.144	0.210	0.129	0.625
6	29.50	0.92	21.5	0.49	44	86	76,610	0.009	0.193	76,605	10.45	0.110	76,605	23,491	5.70	0.162	0.094	0.253	0.159	0.668
7	30.38	0.89	21.6	0.53	44	87	75,192	0.012	0.181	75,178	10.47	0.123	75,182	23,052	5.86	0.143	0.173	0.208	0.123	0.646
8	30.49	0.89	21.5	0.57	44	86	76,555	0.013	0.177	76,558	10.44	0.121	76,552	23,467	5.88	0.137	0.175	0.206	0.123	0.642
9	28.44	0.94	21.8	0.36	46	88	76,609	0.005	0.244	76,630	10.39	0.124	76,625	23,448	5.51	0.196	0.083	0.270	0.150	0.699
10	**31.10**	0.88	21.1	0.58	44	87	76,097	0.005	0.169	76,106	10.54	0.112	76,102	23,347	6.00	0.155	0.156	0.202	0.117	0.630
11	27.07	0.97	22.4	0.22	44	86	77,128	0.040	0.196	77,132	10.16	0.130	77,126	23,646	5.25	0.133	0.148	0.244	0.201	0.726
12	30.66	0.89	21.3	0.53	44	86	75,759	0.006	0.168	75,747	10.50	0.110	75,750	23,209	5.92	0.165	0.117	0.224	0.132	0.638

GCR Gross crane rate (total QC lifts/total hours of QC time beside a busy berth), *Berth Occ* berth occupancy (average fraction of berths occupied at any instant), *# Vess Arrived* number of vessels that arrive during the data collection period (most, but not all, of these vessels also depart during the data collection period), *BOA rate* the berth-on-arrival rate, i.e., the fraction of vessels that are able to berth upon their arrival, *%vol* average percentage of storage volume occupied at any instant, *%area* average percentage of storage area occupied at any instant, *QCLifts* total number of QC lifts made, *QC waitU* fraction of time that an average QC spends waiting for YTs to appear during unloading (averaged over the entire simulation run), *QC waitL* fraction of time that an average QC spends waiting for YTs to appear during loading (averaged over the entire simulation run), *YCLifts* total number of YC lifts made, *YC Prod* average yard crane productivity (total YC lifts/total hours of YC time), *Gant* fraction of time that an average YC spends linear gantrying (averaged over the entire simulation run), *YTHauls* total number of containers placed on YT trailers, *Dual Load* total number of $20'$ containers placed on YT trailers that are already half full, *YT Prod* average yard truck productivity (total number of containers placed on YT trailers/total hours of active YT time), *YT waitU* fraction of time that an average YT spends waiting to receive unloaded containers from QCs (averaged over the entire simulation run), *YT waitS* fraction of time that an average YT spends waiting for YCs to store containers it has brought to the yard (averaged over the entire simulation run), *YT waitR* fraction of time that an average YT spends waiting for YCs to retrieve containers (averaged over the entire simulation run), *YT waitL* fraction of time that an average YT spends waiting to give containers to QCs during loading (averaged over the entire simulation run), *YT wait* = YT waitU + YT waitS + YT waitR + YT waitL
Bold values indicate the highest values in their respective groups

Table 6 Detailed results for the small terminal with more equipment

YC Disp. algorithm	GCR	Berth occ	# Vess arrived	BOA rate	%vol	%area	QCLifts	QC waitU	QC waitL	YCLifts	YC prod	Gant	YTHauls	Dual load	YT prod	YT waitU	YT waitS	YT waitR	YT waitL	YT wait
1	35.55	0.77	20.7	0.80	43	85	76,498	0.000	0.061	76,514	8.44	0.103	76,502	23,447	3.78	0.248	0.052	0.126	0.204	0.630
2	36.27	0.76	20.8	0.85	43	85	76,157	0.000	0.048	76,156	8.52	0.103	76,149	23,342	3.86	0.243	0.068	0.117	0.193	0.620
3	34.25	0.80	20.9	0.72	44	85	76,014	0.000	0.084	76,008	8.47	0.104	76,004	23,306	3.65	0.259	0.037	0.142	0.222	0.660
4	36.27	0.76	20.7	0.85	43	84	74,555	0.000	0.047	74,528	8.49	0.092	74,529	22,846	3.85	0.243	0.053	0.124	0.195	0.615
5	36.78	0.75	21.0	0.87	43	84	74,927	0.000	0.038	74,919	8.46	0.093	74,915	23,001	3.91	0.241	0.063	0.116	0.185	0.605
6	34.86	0.79	20.9	0.79	43	85	76,635	0.000	0.074	76,656	8.49	0.096	76,650	23,472	3.72	0.257	0.038	0.138	0.214	0.647
7	36.70	0.75	20.5	0.89	43	84	74,915	0.000	0.047	74,901	8.48	0.102	74,897	22,932	3.90	0.237	0.063	0.123	0.187	0.609
8	**37.00**	0.75	20.5	0.88	43	84	74,505	0.000	0.041	74,481	8.50	0.101	74,482	22,803	3.93	0.234	0.070	0.118	0.181	0.604
9	35.33	0.78	20.8	0.82	44	85	76,489	0.000	0.074	76,544	8.50	0.104	76,530	23,452	3.76	0.254	0.040	0.140	0.205	0.639
10	**37.23**	0.74	20.8	0.84	43	85	76,468	0.000	0.036	76,471	8.45	0.095	76,460	23,402	3.95	0.246	0.061	0.114	0.177	0.598
11	33.12	0.82	21.0	0.70	43	84	76,057	0.000	0.088	76,055	8.42	0.112	76,040	23,314	3.54	0.219	0.068	0.137	0.253	0.677
12	36.27	0.76	20.5	0.87	43	84	74,746	0.000	0.048	74,722	8.48	0.093	74,721	22,902	3.85	0.245	0.051	0.125	0.195	0.616

Bold values indicate the highest values in their respective groups

Table 7 Detailed results for the large terminal with less equipment

YC Disp. algorithm	GCR	Berth occ	# Vess arrived	BOA rate	%vol	%area	QCLifts	QC waitU	QC waitL	YCLifts	YC prod	Gant	YTHauls	Dual load	YT prod	YT waitU	YT waitS	YT waitR	YT waitL	YT wait
1	27.49	0.99	204.2	0.06	56	83	363,893	0.016	0.234	363,897	10.46	0.129	363,888	86,845	5.30	0.152	0.111	0.210	0.116	0.590
2	28.02	0.99	201.3	0.07	56	84	363,365	0.012	0.228	363,390	10.63	0.131	363,378	86,808	5.39	0.154	0.150	0.185	0.093	0.582
3	26.17	0.99	213.2	0.04	56	84	364,502	0.008	0.268	364,474	10.05	0.122	364,482	87,075	5.05	0.174	0.070	0.243	0.124	0.611
4	28.20	0.98	200.7	0.07	56	84	365,029	0.009	0.221	365,065	10.68	0.113	365,045	87,161	5.43	0.169	0.108	0.202	0.098	0.578
5	**28.70**	0.98	196.8	0.09	56	84	363,166	0.012	0.211	363,194	10.83	0.119	363,177	86,755	5.52	0.157	0.152	0.177	0.085	0.570
6	26.60	0.99	210.3	0.04	56	84	364,462	0.007	0.257	364,492	10.21	0.112	364,473	87,194	5.13	0.175	0.074	0.237	0.120	0.606
7	28.04	0.98	200.8	0.08	57	84	362,757	0.001	0.247	362,816	10.63	0.133	362,782	86,917	5.40	0.184	0.157	0.179	0.064	0.584
8	28.15	0.98	199.9	0.08	57	84	362,636	0.002	0.243	362,641	10.66	0.133	362,637	86,831	5.42	0.177	0.164	0.173	0.065	0.580
9	25.54	0.99	219.3	0.04	57	85	365,943	0.000	0.306	365,989	9.81	0.120	365,968	87,687	4.93	0.206	0.047	0.262	0.107	0.622
10	**28.24**	0.98	199.0	0.08	57	84	363,383	0.001	0.242	363,477	10.70	0.126	363,423	87,022	5.44	0.184	0.159	0.174	0.064	0.580
11	25.48	1.00	218.8	0.02	56	83	365,906	0.017	0.257	366,021	9.81	0.127	365,938	87,534	4.92	0.154	0.104	0.229	0.134	0.622
12	28.01	0.98	201.2	0.06	56	83	364,254	0.011	0.225	364,287	10.62	0.113	364,275	87,022	5.39	0.163	0.103	0.208	0.108	0.581

Bold values indicate the highest values in their respective groups

Table 8 Detailed results for the large terminal with more equipment

YC Disp. algorithm	GCR	Berth occ	# Vess arrived	BOA rate	%vol	%area	QCLifts	QC waitU	QC waitL	YCLifts	YC prod	Gant	YTHauls	Dual load	YT prod	YT waitU	YT waitS	YT waitR	YT waitL	YT wait
1	35.18	0.84	187.6	0.76	55	83	360,834	0.000	0.062	360,894	9.23	0.108	360,880	86,417	3.72	0.284	0.048	0.115	0.156	0.603
2	36.44	0.81	186.8	0.84	55	82	359,648	0.000	0.039	359,621	9.24	0.107	359,632	86,166	3.85	0.270	0.069	0.095	0.142	0.577
3	32.17	0.91	188.5	0.52	56	83	360,420	0.000	0.122	360,435	9.20	0.109	360,433	86,180	3.42	0.296	0.027	0.156	0.198	0.676
4	35.54	0.83	186.5	0.79	55	83	359,515	0.000	0.052	359,519	9.22	0.096	359,515	86,095	3.76	0.282	0.049	0.111	0.153	0.595
5	**36.89**	0.80	186.0	0.87	55	82	358,739	0.000	0.031	358,807	9.23	0.099	358,789	85,862	3.89	0.271	0.067	0.092	0.137	0.567
6	32.19	0.91	189.4	0.54	56	83	361,285	0.000	0.119	361,338	9.19	0.102	361,319	86,576	3.42	0.296	0.028	0.152	0.199	0.675
7	36.03	0.82	187.6	0.85	55	83	359,862	0.000	0.052	359,854	9.22	0.107	359,864	86,135	3.81	0.275	0.059	0.107	0.144	0.584
8	36.69	0.81	187.1	0.83	55	83	358,831	0.000	0.038	358,803	9.22	0.107	358,814	85,839	3.87	0.268	0.071	0.095	0.137	0.571
9	32.42	0.91	188.6	0.59	56	83	359,808	0.000	0.128	359,783	9.20	0.109	359,802	86,179	3.44	0.295	0.027	0.158	0.190	0.670
10	**36.73**	0.81	187.3	0.86	55	83	360,132	0.000	0.037	360,162	9.22	0.101	360,151	86,187	3.88	0.268	0.069	0.094	0.139	0.570
11	31.10	0.94	189.2	0.36	55	83	361,904	0.000	0.134	361,905	9.17	0.115	361,920	86,575	3.31	0.272	0.046	0.151	0.233	0.702
12	35.33	0.84	186.8	0.79	55	83	359,936	0.000	0.058	359,974	9.23	0.098	359,965	86,175	3.74	0.284	0.046	0.115	0.155	0.600

Bold values indicate the highest values in their respective groups

note that relatively little gantrying is done for algorithms 10 and 12, which also have a distance component to their logic.

The last five columns of Tables 5, 6, 7 and 8 offer a partial explanation for why certain YC dispatching algorithms are performing better than others. The first four such columns show the fraction of time that an average YT spends (1) waiting to receive unloaded containers from QCs, (2) waiting for YCs to store containers it has brought to the yard, (3) waiting for YCs to retrieve containers, and (4) waiting to give containers to QCs during loading, averaged over the entire simulation run. The sum of these four values is displayed in the final column. Several trends are visible in these columns. For example, as expected, "YT waitR" ("YT waitS") is smaller for algorithms 2, 5, and 8 (3, 6, and 9), which prioritize the retrieval (storage) of containers from (into) the stacks, than for the other algorithms in their respective peer groups (1–3, 4–6, 7–9). As mentioned earlier, the algorithms that prioritize the retrieval of containers (2, 5, and 8) outperform the other algorithms in their peer groups (1–3, 4–6, and 7–9). Note that, with few exceptions, algorithms 2, 5, and 8 are also yielding lower values than their peers in the "YT waitU" and "YT waitL" columns. This helps to explain their relative success.

Note that, in all terminal scenarios, YTs are spending roughly 60% of their time waiting. They spend the other 40% of the time traveling. Also, note in the more equipment scenarios that the sum of the "YT waitU" and "YT waitL" terms is greater than the sum of the "YT waitS" and "YT waitR" terms. This indicates that the YCs are turning around the YTs more quickly when there is more yard equipment, causing the YTs to wait more at the quay than in the yard. The opposite is true for the less equipment scenarios. Also, note that "YT waitR" is larger than "YT waitS" for all cases. The difference between the former and latter would be even greater if the results from the experiments in which the TR-YTs are unavailable were included. This indicates that yard operations, not the YT substitutability issues mentioned above, are primarily responsible for the extra delay associated with QC loading versus unloading. Indeed, the extra delay associated with yard retrievals translates almost directly into the extra delay experienced by QCs during vessel loading.

Most importantly, note in the final column of Tables 5, 6, 7 and 8 that GCR has a very strong negative correlation with the total amount of waiting done by an average YT. For example, the two lists formed by ordering the YC dispatching algorithms in Table 5 according to decreasing GCR and increasing total YT waiting time are identical. The respective lists for the other terminal scenarios also bear a striking resemblance. The strong negative correlation between GCR and total YT waiting time allow us to define the goals of the YC dispatching system in terms of YTs. In particular, the main goal of the YC dispatching system is to serve YTs in a manner which minimizes their waiting times both in the yard *and at the quay*.

Insights regarding the distribution of YT waiting time, however, seem less forthcoming. In particular, the exact distribution of YT waiting time among the four different categories appears to be very sensitive not only to the YC dispatching algorithm but also to the terminal scenario. Indeed, the relative magnitudes of these four waiting times would have been extremely hard to predict in the absence of a simulation model. Our attempts to derive an analytical expression that approximates these four waiting times have met with little success. Interested readers are invited to make their own speculations about the origins of these values and to formulate their own analytical expressions that relate the terminal specifications and YC dispatching algorithms to these waiting times. But they should be forewarned that the complex, integrated nature of container terminal operations may make such analytical expressions very hard to come by. Overall, it appears that more studies—based on simulation, stochastic, and/or deterministic modeling methodologies—are needed before we can formulate general analytical expressions that directly connect container terminal specifications and YC dispatching algorithms to the different kinds of waiting time experienced by YTs at a container terminal.

7 Conclusion

This paper has investigated how a container terminal's long-run average quay crane rate (GCR) depends on the system that is used for automatically dispatching yard cranes (YCs) to container handling tasks in the yard in real time. Toward this end, the authors constructed an integrated, discrete event simulation model of a pure transshipment terminal that is designed to reproduce the multi-objective, stochastic, real-time environment at a multiple-berth facility. The experiments considered 4 fully-dynamic container terminal scenarios and all combinations of 12 real-time YC dispatching algorithms and 4 yard truck (YT) availability criteria for each of these scenarios. Six probabilistically identical experiments were performed for each of the above setups, yielding 1152 experiments in all. The real-time YC dispatching systems are comprehensive systems that track the real-time status of all YCs at all times; determine the exact routing of every YC at all times; guarantee that YCs remain separated by a minimum distance at all times; and guarantee that real-time operational deadlocks are avoided on 100% of occasions.

Results are both quantitative and qualitative. Quantitatively, the experiments strongly indicate that YC dispatching systems which prioritize the retrieval of containers from the stacks are superior to systems that do not use such a priority. The former systems are effective because they serve the vessel loading operation that causes most of the quay crane delays. Moreover, the YC dispatching system should not only consider the trucks already waiting for service in the yard, but also the trucks that are heading towards the yard. On another front, the numerical results show a strong negative correlation between GCR and total YT waiting time. Thus, the main goal of a YC dispatching system should be to serve YTs in a manner which minimizes their waiting times both in the yard and at the quay. Finally, the simulation model's short runtime means that the proposed real-time YC dispatching systems can be integrated into the operating system at an actual facility. While the experiments are not exhaustive enough to cover all situations, they nevertheless do provide the first direct connection in the literature between real-time YC dispatching systems and long-run performance at a seaport container terminal.

Qualitatively, this research indicates that rule-based YC dispatching algorithms, which avoid deadlocks on 100% of occasions in a very transparent way, may be preferable to look-ahead YC dispatching systems. The research has also highlighted the need for integrated simulation models of container terminals that can test the viability of proposed algorithms and/or models within a real-time environment and evaluate performance in terms of a global indicator such as GCR. Such models are hard to come by in academia, probably because they (1) are hard to debug, (2) are highly susceptible to deadlocking, and (3) must be developed from scratch using a low-level language such as C++ so that the runtime is fast enough to permit adequate experimentation. However, since real-time capability is the essence of operational control, we believe that integrated simulation models are vital to container terminal operations research.

References

Alessandri A, Sacone S, Siri S (2007) Modelling and optimal receding-horizon control of maritime container terminals. J Math Model Algorithms 6:109–133

Briskorn D, Drexl A, Hartmann S (2006) Inventory-based dispatching of automated guided vehicles on container terminals. OR Spectrum 28:611–630

Canonaco P, Legato P, Mazza RM, Roberto M (2008) A queuing network model for the management of berth crane operations. Comp Oper Res. doi:10.1016/j.cor.2006.12.001

Dekker R, Voogd P, Asperen Ev (2006) Advanced methods for container stacking. OR Spectrum 28:563–586

Froyland G, Koch T, Megow N, Duane E, Wren H (2008) Optimizing the landside operation of a container terminal. OR Spectrum 30:53–75

Grunow M, Günther H-O, Lehmann M (2006) Strategies for dispatching AGVs at automated seaport container terminals. OR Spectrum 28:587–610

Günther H-O, Kim K-H (2006) Container terminals and terminal operations. OR Spectrum 28:437–445

Kim KH, Lee KM, Hwang H (2003) Sequencing delivery and receiving operations for yard cranes in port container terminals. Int J Prod Econ 84:283–292

Kozan E (1997) Comparison of analytical and simulation planning models of seaport container terminals. Transport Plan Technol 20:235–248

Legato P, Mazza RM (2001) Berth planning and resources optimisation at a container terminal via discrete event simulation. Euro J Oper Res 133:537–547

Li W, Wu Y, Petering MEH, Goh M, de Souza R (2008) Discrete time model and algorithms for container yard crane scheduling (submitted manuscript)

Linn R, Liu J-y, Wan Y-w, Zhang C, Murty KG (2003) Rubber tired gantry crane deployment for container yard operation. Comp Ind Eng 45:429–442

Liu CI, Jula H, Ioannou PA (2002) Design, simulation, and evaluation of automated container terminals. Intel Transport Syst IEEE Trans 3:12–26

Liu C-I, Jula H, Vukadinovic K, Ioannou P (2004) Automated guided vehicle system for two container yard layouts. Transport Res C Emerg Technol 12:349–368

Murty KG, Liu J, Wan Y-w, Linn R (2005a) A decision support system for operations in a container terminal. Decis Support Syst 39:309–332

Murty KG, Wan Y-W, Liu J, Tseng MM, Leung E, Lai K-K, Chiu HWC (2005b) Hongkong International Terminals gains elastic capacity using a data-intensive decision-support system. Interfaces 35:61–75

Ottjes JA, Veeke HPM, Duinkerken MB, Rijsenbrij JC, Lodewijks G (2006) Simulation of a multiterminal system for container handling. OR Spectrum 28:447–468

Parola F, Sciomachen A (2005) Intermodal container flows in a port system network: analysis of possible growths via simulation models. Int J Prod Econ 97:75–88

Petering MEH (2007) Design, analysis, and real-time control of seaport container transshipment terminals. PhD dissertation, University of Michigan, Ann Arbor, Michigan

Petering MEH, Murty KG (2008) Effect of block length and yard crane deployment systems on overall performance at a seaport container transshipment terminal. Comp Oper Res (in press). doi:10.1016/j.cor.2008.04.007

Petering MEH, Murty KG (2007) Real-time container storage location assignment at a seaport container transshipment terminal (submitted manuscript)

Petering MEH, Murty KG (2006) Simulation analysis of algorithms for container storage and yard crane scheduling at a container terminal. In: Proceedings of the second international intelligent logistics systems conference, Brisbane, Australia

Petering MEH, Wu Y, Li W, Goh M, Murty KG, de Souza R (2006) Simulation analysis of yard crane routing systems at a marine container transshipment terminal. In: Proceedings of the international congress on logistics and supply chain management systems, Kaohsiung, Taiwan

Stahlbock R, Voß S (2008) Operations research at container terminals: a literature update. OR Spectrum 30:1–52

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

Vis IFA, de Koster R (2003) Transshipment of containers at a container terminal: an overview. Eur J Oper Res 147:1–16

Yang C, Choi Y, Ha T (2004) Simulation-based performance evaluation of transport vehicles at automated container terminals. OR Spectrum 26:149–170