European Journal of Operational Research 198 (2009) 165–172

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing and Logistics

Discrete time model and algorithms for container yard crane scheduling

Wenkai Li*, Yong Wu, M.E.H. Petering[1], Mark Goh[2], Robert de Souza

The Logistics Institute, Asia Pacific, Block E3A, Level 3, 7 Engineering Drive 1, Singapore 117574, Singapore

ARTICLE INFO

Article history:
Received 4 February 2008
Accepted 28 August 2008
Available online 6 September 2008

Keywords:
Scheduling
Rolling-horizon algorithm
Container yard
MILP

ABSTRACT

Container terminal (CT) operations are often bottlenecked by slow YC (yard crane) movements. PM (prime mover) queues in front of the YCs are common. Hence, efficient YC scheduling to reduce the PM waiting time is critical in increasing a CT's throughput. We develop an efficient model for YC scheduling by taking into account realistic operational constraints such as inter-crane interference, fixed YC separation distances and simultaneous container storage/retrievals. Among them, only inter-crane interference has ever been considered in the literature. The model requires far fewer integer variables than the literature by using bi-index decision variables. We show how the model can be solved quickly using heuristics and rolling-horizon algorithm, yielding close to optimal solutions in seconds. The solution quality and solution time are both better than the literature even with additional constraints considered. The proposed formulations and algorithms can be extended to other problems with time windows and space constraints.

1. Introduction

Today, 90% of the world's trade is transported via containers (Henwood, 2006), mostly on containerships (Kim and Kim, 1999; Steenken et al., 2004), leading to greater volume flows at the ports. Coupled with increased demand for speed to market, the container terminal (CT) operator faces constant pressure to reduce turnaround time and increase flow efficiency. Hence, CT operators need better and efficient computational tools which can de-bottleneck a CT's container flow and increase container box throughput.

Vessels mooring at the berths wait for quay cranes (QCs) to upload/discharge their containers (see Fig. 1). The PMs shuttle between the QCs and YCs to move these boxes from the berth to the container yard (CY), and vice versa. Upon arrival at the blocks, the PMs queue in front of the YCs until they are served. As the typical YC to QC work rate is half (Ng and Mak, 2005), YC operations is a potential bottleneck. The efficiency of yard operations depends heavily on the YCs' operations (Zhang et al., 2002).

The CY, a container storage area inside the CT of a port, for planning purposes, is typically partitioned as contiguous rectangular blocks (see Fig. 2). A grid is painted on the pavement indicating the x–y locations of the boxes. Bi-directional traffic lanes for the PMs occupy the space between the blocks. The blocks are divided along their length into 20-foot sections called slots. Each slot has several rows. Containers are stored alongside in each row and are stacked on top of each other. A typical block is six rows (6×8.5 feet) deep and forty slots (40×20 feet) long. The YCs are used to transfer containers in and out of the slots, straddling above the containers in each block and moving parallel to the length of the block. For a transshipment CT, with 40–60 slots per block, up to 60 moves are expected to be handled at each block (Ng, 2005).

Generally, the terminal planners are informed of the vessel arrivals a few days prior. They then start to plan the storage/retrieval locations for the containers in the CY (Ng, 2005). Hence the slot to be picked is already pre-determined in most CTs. However, due to the arrival uncertainty of the vessels, the container loading/unloading sequence in the QC work list is planned only several hours before the actual vessel arrival. This work list is then translated into an YC work list, using historical average container handling times for the QCs and YCs, and travel times for the PMs, respectively. The terminal schedulers then dispatch the YCs manually, based on the actual container arrival pattern. Besides the import containers arriving from vessels, in a transshipment CT, there are export containers arriving from the landside of the terminal. Such export containers are stored at the stack yard temporarily and later moved to quayside by YCs and PMs (Froyland et al., 2008). This will also generate storage and retrieval moves for the YCs. Given this situation, it is optimal to generate an integrated YC work list which includes the work list translated from both the landside planning and quayside planning. However, in this paper, for ease of exposition, only the YC work list translated from the QC work list is considered. Coordinating the movement of containers on the quayside is already a difficult problem in itself.

* Corresponding author. Tel.: +65 65168987; fax: +65 67753391.
E-mail addresses: tlilw@nus.edu.sg, wenkaili2002@yahoo.com (W. Li).

[1] Also at Industrial & Manufacturing Engineering Department, University of Wisconsin–Milwaukee, USA.

[2] Also at NUS business school.

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.08.019

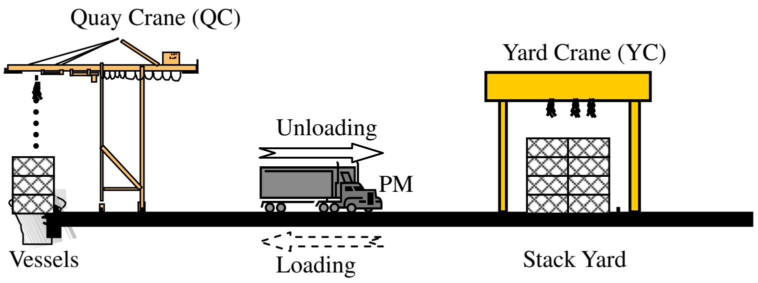

Fig. 1. Typical container terminal.

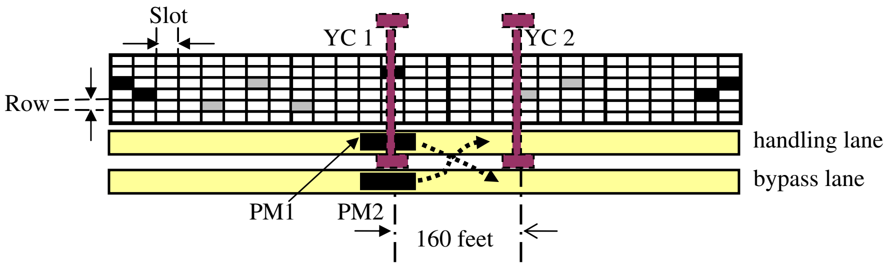

Fig. 2. 160 feet separation distance between working YCs for a 28-slot block.

The gross crane rate (GCR) measures the average rate at which the QCs transfer containers between vessel and shore and is the most significant performance measure of a CT operation. A high GCR, while desirable, is however constrained by the ability of the PMs to reach the QCs, which in turn is constrained by the YCs' work rate. In short, even though a QC is technically capable of making 40 moves (load/discharge containers) per hour, the average QC rate at most container ports is currently far less (more than 30%). An efficient YC schedule is therefore key.

Some unique operation constraints exist in YC scheduling. First, two YCs sharing the same bi-directional lane cannot cross gantry each other, i.e., an YC located at one position of a block cannot move to another position in the same block if another YC is in its path. Some studies (Ng, 2005; Lim et al., 2007; Froyland et al., 2008) have considered this constraint. Second, the literature does not consider a safety distance between YCs. Currently, the safety distance between two YCs operating in the same lane must be at least 160 feet apart (Fig. 2). This greatly limits its access to the containers in the intervening slots. For instance, if an YC is working at slot #8 in a block, then there can be no container handling in slots #1 to #15, effectively reducing the block workspace by 37.5% for a 40-slot block. This safety distance is intended for the PMs to park and move between the YCs, and to move from the handling lane to a bypass lane and back. If YCs are closer than this, the PM (PM1 in Fig. 2) leaving the upstream YC (YC1 in Fig. 2) will not be able to pull out of the handling lane and into the bypass lane through to the downstream YC (YC2 in Fig. 2). Third, in a transshipment CT, YCs often need to handle two kinds of moves, storage and retrieval, in a time window. Containers should either be stored or retrieved from CY by YCs from/to PMs (Fig. 1). Froyland et al. (2008) have considered storage and retrieval moves using heuristics to sequence container moves without setting a minimum YC separation distance. Other literature (Ng, 2005; Ng and Mak, 2005; Ng and Tsang, 2005) only consider storage moves.

2. Literature review

There are many papers on planning and scheduling problems in a container terminal. Vis and Koster (2003); Steenken et al. (2004) and Stahlbock and Voß, 2008 provide excellent surveys of CT operations and research. For instance, Li et al. (2007) have developed models for allocating empty containers among ports to better manage and control their containers. Similarly, Kim and Park (2004) and Bish (2003) studied the scheduling for QCs. Many papers have considered the scheduling of a single YC where containers are grouped and the YC must retrieve containers from specified groups according to a fixed sequence, without due dates or release times, while minimizing travel distance. Since containers belonging to a specific group may be stored in multiple locations, both the YC route and the number of containers picked at each slot are decision variables (Kim and Kim, 1999, 2003; Narasimhan and Palekar, 2002).

Scheduling algorithms for multiple YCs are rarely addressed in the literature (Steenken et al., 2004; Stahlbock and Voß, 2008). Cheung et al. (2002) and Zhang et al. (2002) have developed methods for allocating YCs among the blocks in an entire CT and for scheduling cross-gantry moves. However, they do not generate detailed work schedules for the YCs and individual containers. Chen et al. (2007) have developed an integrated scheduling model for container handling where a Tabu search algorithm was used to solved the model. Again, the detailed work schedules for YCs was not generated. So far, most of the algorithms applied on the YC scheduling problem (Ng, 2005; Ng and Mak, 2005; Ng and Tsang, 2005) only treat storage moves, probably because it is easier to develop algorithms for storage moves. Recently, Ng (2005) has proposed a heuristic for constructing detailed work schedules for multiple YCs in a block with inter-crane interference, albeit without retrieval moves and minimum crane-distance separation.

2.1. Some approaches to YC work schedules

Dispatching: The YC scheduling problem is NP-hard (Ng, 2005). Currently, dispatching is applied extensively in CTs using simple rules (e.g., SPT, EDD) for YC scheduling. However, such rules are myopic in nature (Hopp and Spearman, 2000).

Simulation: Another way to combat this NP-hard scheduling problem is through simulation, and generates schedules by applying different dispatching rules. However, finding an effective schedule is a time-consuming process and as dispatching rules are inherently myopic, simulation may not generate good schedules (Hopp and Spearman, 2000).

Heuristics-based algorithms: Some researchers (Ng, 2005) apply heuristics-based algorithms for YC scheduling. These algorithms can generate a schedule quickly. However, lacking an efficient way to search for a globally good schedule, the final schedules generated by such algorithms may be far from optimal.

Metaheuristics-based approaches: Another class of methods apply metaheuristics for YC scheduling. Recently, Ng and Mak (2005) proposed a branch-and-bound algorithm to solve the problem of storage moves, requiring about 444.7 seconds in average to solve YC scheduling problem with 25 jobs. Their algorithm cannot be applied on retrieval moves at the same time as the procedure fails to find the lower bound. Ng and Tsang (2005) later developed a genetic algorithm to solve a single YC scheduling problem. Again, only storage moves are considered.

MILP models for YC work schedules: Ng (2005) proposed an MILP model as the benchmark for his heuristics-based algorithm considering inter-crane interference. Ng found that CPLEX took about 10 minutes to find the optimal schedule for a problem with 2 YCs and 10 jobs in a 40-slot block. Due to the exponential increase in solution time, solving MILP-based models directly can be intractable for real-sized problems. All research thus far use MILP-based models as benchmarks only.

This paper develops an efficient model for YC work schedules by treating actual operation constraints such as inter-crane interference, fixed YC separation distances and simultaneous storage/retrieval jobs handling. By detailed formulating and exploiting the structure of the problem, the model size is dramatically decreased. An effective heuristic is integrated into the model to further reduce the model size. We then apply a rolling-horizon algorithm to sequence jobs based on job target times. Jobs are scheduled at each iteration until all jobs are fixed. This allows unscheduled new jobs to be inserted among the existing set of jobs. The rolling-horizon algorithm can find near-optimal solutions. Thus far, the solutions of many scenarios tested are optimal. Since the number of jobs involved at each iteration is set to a small value (7 jobs/iteration is used in this paper), the problem size of each iteration is small, reaching solution in less than one second usually. As the total problem size increases, the total solution time only increases linearly in n, i.e., $O\left(\frac{n}{\text{JFPI}} * e^{\text{JNPI}}\right)$, where n (>7) is the total number of jobs to be scheduled, JFPI is the number of jobs fixed at each iteration (2 is used in this paper) and *JNPI* is the job number per iteration (7 is used here). The exponential term, e^{JNPI}, is much smaller compared to e^n.

3. The model (DMIP1)

An MILP model is developed to handle the problem of scheduling single or multiple yard cranes in a yard block. Storage or retrieval moves arrive at the block with different target times. For retrieval jobs, the target time interval is the latest time interval during which a retrieval job can be handled and still meet the deadline set by the QCs. For storage jobs, the target time interval is the earliest time interval following the release of the job (i.e., the arrival of the corresponding PM) in the yard. In order to reduce the number of integer variables in our IP model, we discretize the time axis into 3.5-minute intervals. Because an YC takes about 2–4 minutes to make a container transfer, we assume a 3-minute handling time per container move. Each container is scheduled to take place in exactly one interval and shall consist of 0.5 minute of YC gantrying followed by 3 minutes of container handling. In the model, each YC handles at most one container per interval. In any given interval, all gantry moves start and end simultaneously and all container moves start and end simultaneously for all YCs.

3.1. Assumptions

The following assumptions are used in the model.

- Target times and locations of container moves are assumed known and fixed. The target times can be translated from the QC work list.
- The job handling time of an YC is usually 2–4 minutes (Ng and Mak, 2005). We assume that the job handling time of all YCs is 3 minutes (i.e., 20 moves/hour).
- 20–30 moves in a 2-hour time window are used in the scenarios tested in this paper. Each yard block contains 40–60 container slots.
- In a CT, YC–YC interference is usually more serious than PM–PM interference as YCs have more difficulty substituting for each other than PMs during actual operations (e.g. there are fewer YCs than PMs, YCs are much slower than PMs, and the minimum separation distance is larger for two YCs than for two PMs). Thus, more attention should be paid to scheduling YCs than to dispatching PMs. YCs also exhibit less variable travel and handling times than PMs, which means that they can follow a predetermined sequence of moves with a less variable result than PMs. Thus, generating YC schedules with the assumption that PMs are always available becomes very worthwhile.
- the minimum difference in slot numbers allowed for two YCs at the same time is assumed to be 8 slots (160 feet) in this paper ($SEP = 8$).

3.2. Objective function

The notations used for the indices, sets, parameters and variables in the mathematical formulation are defined in Appendix A. The objective is to minimize a linear combination of the retrieval earliness and storage and retrieval delays, namely

$$\text{Minimize} \quad TC = W_{\text{re}} \sum_{m \in R} RE_m + W_{\text{rd}} \sum_{m \in R} RL_m + W_{\text{sd}} \sum_{m \in S} STL_m. \qquad \text{(obj)}$$

As a retrieval delay by the YC directly leads to the QC schedules and berth operations being delayed, and the storage delay only affects the yard operations, the weight for the total retrieval delay, W_{rd}, is set larger. Hence, we set $W_{\text{re}} = W_{\text{sd}} = 1$ and $W_{\text{rd}} = 2$.

3.3. Constraints

Each move takes place during exactly 1 time interval

$$\sum_{t \in T} X_{mt} = 1, \quad \forall m \in M, \qquad (1)$$

where T is the set of time intervals scheduled. It is determined by dividing the upper bound of the makespan of the optimal schedule by the interval length. The upper bound of the makespan (assumed to be a known value in Ng (2005)) is decided by trial-and-error. In this paper, we set this upper bound to the maximum target time of all moves plus 50 minutes. That is, we reasonably assume that, in the optimal schedule, no move will be handled 50 minutes after the largest target time:

$$T = \{t : Ord_t * IntLen \leqslant MaxH + 50\},$$

where $\text{Max}H = \text{Max}\{m \in M{:}\text{Tgt}_m\}$, Ord_t is the relative position of time interval t in set T and IntLen is the length of time interval.

Each move should be assigned to one crane

$$\sum_{c \in C} W_{mc} = 1, \quad \forall m \in M, \; NC > 1. \qquad (2)$$

When the number of YCs is greater than one, constraint (2) ensures that each move is assigned to exactly one YC. If there is only one YC, the value of W_{mc} is fixed to 1

$$W_{m,c1} = 1, \quad \forall m \in M,\ NC = 1.$$

A storage move can take place no earlier than its target time interval

$$\sum_{t\in T} IntLen * (Ord_t - 1) * X_{mt} \geqslant Tgt_m, \quad \forall m \in S_m. \tag{3}$$

At time t, at most NC YCs can work simultaneously

$$\sum_{m} X_{mt} \leqslant NC, \quad \forall t \in T. \tag{4}$$

NC is the total number of YCs scheduled.

Computing the retrieval earliness, RE_m, of move m:

$$RE_m \geqslant Tgt_m - \sum_{t\in T} IntLen * (Ord_t - 1) * X_{mt}, \quad \forall m \in R_m. \tag{5}$$

Computing the retrieval lateness, RL_m, of move m:

$$RL_m \geqslant \sum_{t\in T} IntLen * (Ord_t - 1) * X_{mt} - Tgt_m, \quad \forall m \in R_m. \tag{6}$$

Computing the storage lateness, STL_m, of move m:

$$STL_m \geqslant \sum_{t\in T} IntLen * (Ord_t - 1) * X_{mt} - Tgt_m, \quad \forall m \in S_m. \tag{7}$$

Identifying the status of YC c at time t

$$YY_{mct} \geqslant X_{mt} + W_{mc} - 1, \quad \forall m \in M,\ \forall c \in C,\ NC > 1,\ \forall t \in T, \tag{8a}$$
$$YY_{mct} \leqslant W_{mc}, \quad \forall m \in M,\ \forall c \in C,\ NC > 1,\ \forall t \in T, \tag{8b}$$
$$YY_{mct} \leqslant X_{mt}, \quad \forall m \in M, \forall c \in C, NC > 1, \forall t \in T. \tag{8c}$$

Constraints (8a)–(8c) enforce that, if YC c is handling move m at time t (i.e., $X_{mt} = W_{mc} = 1$), YY_{mct} takes a value of 1. Note that constraints (8a)–(8c) ensure that the value of YY_{mct} is either 0 or 1. Thus, YY_{mct} can be defined as a continuous variable. Constraints (8a)–(8c) are only valid for multi-crane scenarios.

Moves handled by neighboring YCs

$$\sum\nolimits_{n\in PLJ_{mn}} YY_{n,c+1,t} \leqslant 1 - YY_{mct}, \quad \forall c \in C,\ \forall t \in T,\ \forall m \in M,$$
$$c < NC,\ NC > 1. \tag{9}$$

Constraint (9) is only valid for multi-crane scenarios. It states that, if job m is being handled by YC c at time interval t (i.e., $YY_{mct} = 1$), then YC $c + 1$ cannot handle moves which belong to PLJ_{mn}.

Without loss of generality, the slots of a block are ordered from left to right. We also assume that moves are ordered according to the slot number they are located in, i.e., if $m < n$, then $\text{slot}_m \leqslant \text{slot}_n$. With the above assumptions, we define set PLJ_{mn} as

$$PLJ_{mn} = \{m, n \in M : (\text{slot}_n - \text{slot}_m) < SEP \wedge m \neq n\}.$$

That is, those moves that are placed to the left of move m and those moves that are placed to the right of move m but their slot difference against move m is less than SEP belongs to PLJ_{mn} (Fig. 3). For example, in Fig. 3, YC 2 cannot handle moves that belong to PLJ_{mn} when YC 1 is handling move m. This is required by SEP apart of YCs and inter-crane interference.

At time t, an YC can handle at most one move

$$\sum_{m\in M} YY_{mct} \leqslant 1, \quad \forall c \in C,\ \forall t \in T,\ NC > 1. \tag{10}$$

Constraint (10) is only valid for multi-crane scenarios.

Consecutive moves handled by an YC

We do not allow any crane to gantry more than TVL (e.g. 8) slots between container handling operations. In other words, if YC c is handling move m at interval t (i.e., $YY_{mct} = 1$) and the slot position

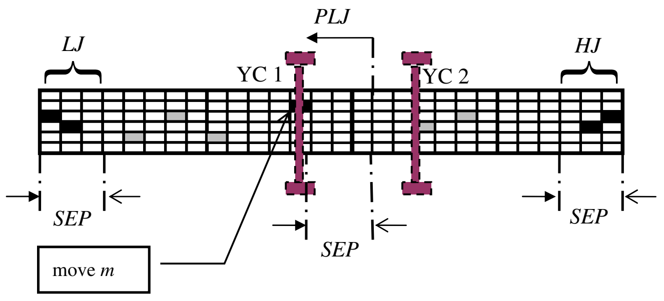

Fig. 3. Definitions of *PLJ*, *HJ* and *LJ*.

of move n is too far (farther than TVL), then YC c cannot handle move n at interval $t + 1$ (i.e., $YY_{nc,t+1} = 0$). This is enforced by constraint (11) as follows:

$$YY_{n,c,t+1} \leqslant 1 - YY_{mct}, \quad \forall c \in C,\ \forall t \in T,\ t < NT,$$
$$\forall m, n \in FJ_{mn},\ NC > 1. \tag{11}$$

NT is the total number of time intervals in set T. Constraint (11) is only valid for multi-crane scenarios. Set FJ_{mn} is defined as

$$FJ_{mn} = \{m, n \in M : |\text{slot}_n - \text{slot}_m| > TVL \wedge m \neq n\}.$$

That is, those moves that are TVL slots away from move m belong to FJ_{mn}.

If there is only one YC, the following constraint is used to replace (11):

$$X_{n,t+1} \leqslant 1 - X_{mt}, \quad \forall t < NT,\ \forall m, n \in FJ_{mn},\ NC = 1. \tag{12}$$

Moves located at the first and last SEP slots

Moves located at the first SEP slots (see Fig. 3) can only be assigned to the first YC, due to inter-crane interference. Similarly, moves at the last SEP slots can only be assigned to the last YC. This is enforced as follows:

$$W_{mc} = 1, \quad \forall m \in LJ_m,\ c = 1,$$
$$W_{mc} = 1, \quad \forall m \in HJ_m,\ c = NC.$$

Set LJ_m is defined as

$$LJ_m = \{m \in M : \text{slot}_m \leqslant SEP\}.$$

LJ_m includes moves located at the first SEP slots (see Fig. 3). Similarly, HJ_m is defined as

$$HJ_m = \{m \in M : slot_m > (NSL\text{—}SEP)\},$$

where NSL is the total number of slots. HJ_m includes moves located at the last SEP slots (see Fig. 3).

As the storage moves can only be handled after their target times, we can fix some X_{mt} to 0:

$$X_{mt} = 0, \quad \forall m, t \in XS_{mt},$$

where set XS_{mt} denotes time intervals t which are earlier than the target time of storage move m. It is defined as

$$XS_{mt} = \{m \in S_m, t \in T : IntLen * (Ord_t - 1) - Tgt_m < 0\}.$$

3.4. Results of DMIP1

The number of YCs in each block varies from 1 to 4. Thirty test scenarios with 20–32 moves in 40-slot blocks and thirty test scenarios with 20–32 moves in 60-slot blocks are randomly generated. ILOG CPLEX 9.0 is used to solve all the models developed in this paper on a Pentium 1.6GHz computer. The MILP relative optimality gap of CPLEX is set to 0 (optcr = 0.0). No priority branching for the binary variables is used (PriorOpt = 0). All other settings follow directly from system default.

An illustrative example (*EX1*), which contains 2 YCs, 60 slots and 32 jobs, is used to generate the results for *DMIP1*. Table 1 contains the dataset for *EX1*. The start time of the scheduling horizon is 168 minute. Using the formulations described in Section 3, *DMIP1* involves 1291 binary variables (49 time intervals), 4813 single variables and 84,212 single equations. The solution time is 8604.7 seconds. The total PM waiting time is 106.647 minutes.

We note that in the MILP model developed in Ng (2005), three tri-index binary variables, X_{mnc}, Y_{cSLt}, W_{mct} are defined which involves $32^{*}32^{*}2+2^{*}60^{*}49+32^{*}2^{*}49=11{,}064$ binary variables in a 49-time interval horizon. Furthermore, to increase the accuracy, each interval is defined to be the time YCs required to travel a slot in Ng's model. On average, an YC takes 4 seconds to gantry one slot. Thus, in a 2-hour time window, there are 1800 time intervals. The number of binary variables involved in Ng's model is now $32^{*}32^{*}2+2^{*}60^{*}1800+32^{*}2^{*}1800=333{,}248$, which is considerably larger than *DMIP1*.

Table 1
Data for the illustrative example (*EX1*)

Moves	Target time (minute)	$slot_m$	Type*
m1	270.56	2	S
m2	261.317	3	S
m3	226.336	6	S
m4	198.957	8	S
m5	168.451	9	S
m6	214.004	11	S
m7	272.078	13	S
m8	217.108	15	S
m9	200.291	16	S
m10	250.959	20	S
m11	171.535	22	S
m12	204.811	23	S
m13	172.899	24	S
m14	260.132	26	S
m15	204.783	27	S
m16	262.883	31	S
m17	192.807	32	S
m18	261.393	33	S
m19	212.644	36	S
m20	261.324	37	S
m21	290.849	37	R
m22	220.24	42	R
m23	246.347	43	R
m24	174.536	44	R
m25	230.677	47	R
m26	207.904	47	R
m27	217.201	51	R
m28	290.617	52	R
m29	221.771	54	R
m30	180.469	58	R
m31	204.882	59	R
m32	224.63	60	R

* R: retrieval move; S: storage move.

Table 2
Results of *DMIP1*

Scenario	Solution time, seconds		
	Mean	Maximum	Minimum
40 slots	1917.5	50,208.2	0.6
60 slots	7007.3	146,374.7	2.5
Pure storage moves	450.5	4815.7	0.6
Storage + retrieval	7137.1	146,374.7	0.7

The results of *DMIP1* are shown in Table 2. Due to the isolated cases of extremely large maximum solution times, the mean solution time to solve 40-slot blocks with 2 YCs is 1, 917.5 seconds. For 60-slot blocks, the mean solution time is 7007.3 seconds. *DMIP1* can yield a good solution in 450.5 seconds on average for scenarios with pure storage moves. This is better than the average of 10 minutes to solve smaller problems with only 10 pure storage jobs in Ng (2005). Note that the solution time may increase exponentially when the number of jobs increases from 10 to 20–32. Generally, *DMIP1* takes much longer (about 7137.1 seconds) to obtain a good solution when retrieval moves are involved in the scenarios.

4. Model integrated with heuristics (DMIP2)

Though the number of binary variables has been reduced significantly by defining only two bi-index binary variables, W_{mc} and X_{mt}, in *DMIP1* compared to defining tri-index binary variables (see Ng, 2005), the size of the model still increases significantly as the number of moves increases. We propose a heuristic to narrow the search space in *DMIP2*. We observe that, for both storage and retrieval moves, the larger the difference between the job finish time of a move and its target time, the larger the total PM waiting time. Since the total PM waiting time is to be minimized, the job finish time is arranged around its corresponding target time as near as possible in an optimal solution. We can reasonably assume that the job finish time of each move is placed inside a certain range (job handling range) around its target time. To apply the above heuristic into the model, we first define set TS_{mt} for storage moves as follows:

$$TS_{mt}=\{m\in S_m,t\in T: IntLen*(Ord_t-1)-Tgt_m \geqslant 0\cap IntLen*(Ord_t-1)-Tgt_m<SU*IntLen\}.$$

That is, the time intervals between the target time and $SU^{*}IntLen$ above the target time of move m belongs to set TS_{mt}. Similarly, we define set TR_{mt} for retrieval moves and TSR_{mt} for both of them.

$$TR_{mt}=\{m\in R_m,t\in T: Tgt_m-IntLen*(Ord_t-1) \geqslant -RL*IntLen\cap Tgt_m-IntLen*(Ord_t-1)<RU*IntLen\},$$

$$TSR_{mt}=TS_{mt}\cup TR_{mt}.$$

We force move m to be handled within TSR_{mt} by

$$X_{mt}=0,\quad \forall m,\ t\notin TSR_{mt};\quad YY_{mct}=0,\quad \forall m,\ t\notin TSR_{mt}.$$

SU, *RL* and *RU* are parameters used to define the job handling range. Their values can be adjusted according to the density of the planned job arrivals. If a lot of moves arrive within a short period of time, the YCs will be quite busy. Hence it is possible that some moves may be scheduled far from their target times. We set the values of *SU*, *RL* and *RU* higher in this case. *SU*, *RL* and *RU* are all set to value of 8.0, which is conservative and satisfies all the scenarios tested in this paper. Some constraints should be modified when applied to set TSR_{mt}. For example, constraint (9) is defined within TSR_{mt}:

$$\sum\nolimits_{n\in PLJ_{mn}} YY_{n,c+1,t}\leqslant 1-YY_{mct},\quad \forall c\in C,\ \forall t\in T,\ \forall m\in M,\ c<NC,\ NC>1,\ \forall m,\ t\in TSR_{mt}. \tag{9'}$$

Similarly, constraints (8a)–(8c), (10)–(12) should also be defined within TSR_{mt}.

4.1. Results of DMIP2

For *EX1* described in Section 3.4, *DMIP2* involves 398 binary variables, 4548 single variables and 30,572 single equations. The model size is significantly reduced compared to *DMIP1*. *EX1* was

Table 3
Results of *DMIP2*

Scenario	Solution time, seconds			Gap, %
	Mean	Maximum	Minimum	$(TCT - TCT_{LB})/TCT_{LB}$
40 slots	208.3	5390.6	0.2	NA
60 slots	1703.5	36,842.6	0.4	NA
Pure storage moves	51.9	924.0	0.2	5.1
Storage + retrieval	1558.6	36,842.6	0.3	NA

solved in 676.5 seconds with a total PM waiting time of 106.647 minutes.

In Table 3, all scenarios are solved to optimality by setting the MIP relative optimality gap to zero. The mean solution time needed to solve 40-slot blocks is 208.3 seconds. For 60-slot blocks, the mean solution time is 1703.5 seconds. Generally, scenarios can be solved in several seconds. The average solution time is lengthened by 2–3 scenarios with very long solution times. To compare the results with the literature, we calculate the lower bounds (TCT_{LB} in Table 3) for the scenarios with pure storage moves by applying the evaluation procedure proposed by Ng (2005). The average/maximum/minimum *TCT* gaps for pure storage scenarios from our model are 5.1%/14.4%/2.4%. That is, even with additional constraints (YC safety distance, simultaneous storage/retrieval moves) considered in our paper, the solution quality is still better than the current literature (7.3% gap achieved in Ng, 2005).

We note that *DMIP2* generally takes a much longer time (average of 1558.6 seconds) to obtain a good solution when retrieval moves are involved while *DMIP2* can yield a good solution in about 51.9 seconds for scenarios with pure storage moves, making it much more efficient than the model developed by Ng (2005). This may be because retrieval moves can be either handled before and after their target times while storage moves can only be handled after.

5. Rolling-horizon algorithm for DMIP2 (DMIP3)

Though *DMIP2* has reduced the model size and the solution time significantly, it is unstable and sometimes still requires very long solution time for some scenarios. As observed in Section 4, the job finish time of a move should be scheduled as near as possible around its target time. In other words, a move *n* which arrives much later than move *m* is very likely to finish later than move *m*, otherwise a large gap between the job finish time and target time will be incurred. This gives us the opportunity to divide the jobs into many groups. Moves with near job target times are grouped together and their handling sequence can be interchanged. All moves in a higher group (moves with bigger job target times) will be handled later than a lower group (moves with smaller job target times). For example in Fig. 4, all scheduled job handling times of moves in group *H* are bigger than moves in group *M* and *L*. Thus, we can divide the problem into several sub-problems; each sub-problem involves moves of only one group. Because each sub-problem involving fewer moves can be solved quickly, the entire YC scheduling problem can be solved quickly.

In Fig. 4, we observe that, the moves in group *M* which have big target times and moves in group *H* which have small target times are actually near jobs. Their handling sequence can possibly also be interchanged. Thus, when implementing the algorithm, all sub-problems are generated in a rolling-horizon fashion. In Fig. 5, the first sub-problem includes moves *m1*–*m5*. After solving this small sub-problem, two moves (*m1* and *m2*) with the smallest handling times are fixed. In the second sub-problem, moves *m3*–*m5* are also included because their handling sequence may possibly be interchanged with the two new added moves (*m6* and *m7*). In sub-problem 2, if several retrieval moves have very near target times, it is possible to insert one or several of these moves into the empty space among the jobs fixed. For example, if moves *m3*–*m5* are retrieval jobs and their target times are almost the same, we may need to insert one of these moves into the space between moves m1 and m2 which have been fixed in sub-problem 1. After solving sub-problem 2, we can fix two more moves and add two new moves into the sub-problem 3. The algorithm stops when all moves have been fixed.

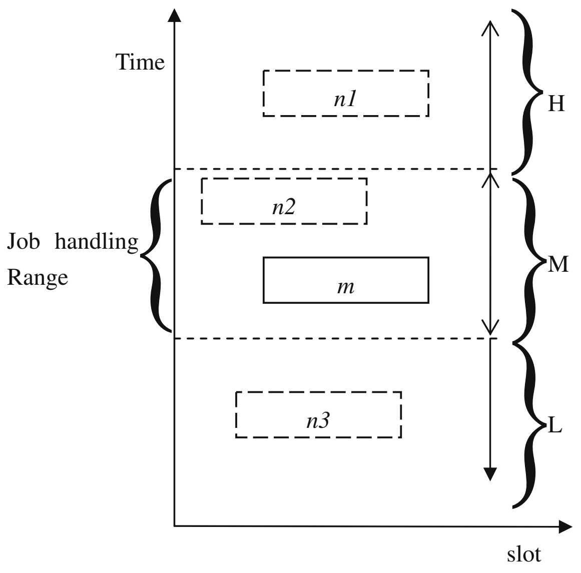

Fig. 4. Definition of neighboring moves.

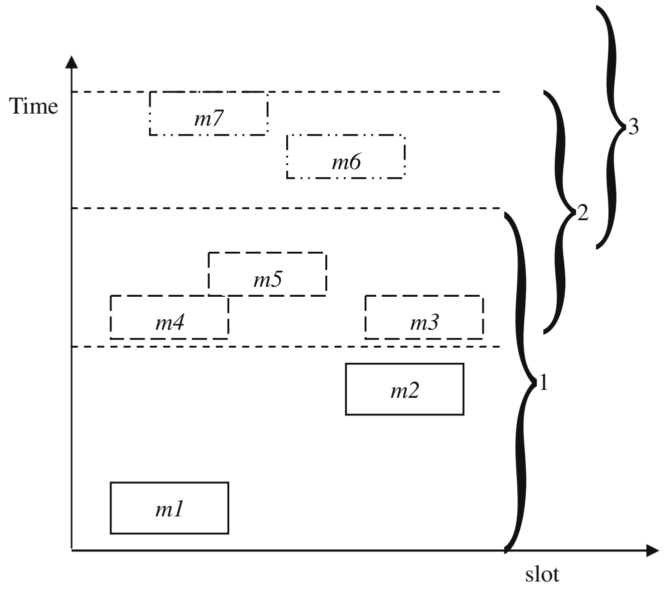

Fig. 5. Definition of rolling sub-problems.

5.1. Constraints and sets used in the rolling-horizon algorithm

To implement the rolling-horizon algorithm, some sets and constraints in Sections 3 and 4 need to be modified. We first define set PM_m to include moves involved in a sub-problem. Then we redefine sets FJ_{mn}, PLJ_{mn}, HJ_{mn}, LJ_{mn} and TSR_{mt} under set PM_m. For example, set FJ_{mn} is redefined as

$FJ_{mn} = \{m, n \in M : |slot_n - slot_m| > TVL \wedge m \neq n \wedge PM_m \wedge PM_n\}$.

Note that in the above definition, only "$\wedge PM_m \wedge PM_n$" is added from the previous definition.

Related constraints should be modified under set PM_m in *DMIP3*. For example, constraint (2) is replaced by (2′):

$$\sum_{c \in C} W_{mc} = 1, \quad \forall m \in PM_m,\ NC > 1. \tag{2'}$$

We allow unscheduled free jobs (i.e., PM_m) to be inserted among the existing set of fixed jobs (i.e., FxJ_m) in the rolling-horizon algorithm. Several constraints should also be added to define the relationship between free moves and fixed moves. These constraints are similar to (4), (9)–(12) except that they are defined under sets PM_m and FxJ_m.

5.2. Flow diagram of rolling-horizon algorithm

Fig. 6 shows the flow diagram of the rolling-horizon algorithm. First, the *include* files are read first to define the whole problem (number of moves, target time, job type, slot position, YC number, etc.). Then we set the value of parameters *JNPI* (=7) and *SU, RL, RU* (=8). *JNPI* can be adjusted according to the tradeoff between the solution quality and time. The solution quality increases as *JNPI*, the solution time of each iteration and hence the total solution time may increase. Sets FxJ_m, $TsFin_m$, $WFin_{mc}$ $XFin_{mt}$, $YYFin_{mct}$ are set to empty at the beginning. All moves are sorted according to their target times and saved in set $Order_m$. Set PM_m includes moves involved in an iteration. At the beginning, PM_m includes the first *JNPI* moves with the smallest target times. With PM_m defined, sets FJ_{mn}, PLJ_{mn}, HJ_{mn}, LJ_{mn} and TSR_{mt} are updated. *DMIP3* is then solved to obtain the schedule for the first *JNPI* moves. This usually takes less than 0.1 second to obtain an optimal solution because the size of *DMIP3* is very small at each iteration. Two moves with the smallest job handling times are fixed and added into the set FxJ_m from the result of *DMIP3*. Their job finishing times and handling YCs are recorded into sets $TsFin_m$ and $WFin_{mc}$, respectively. The *RE* (retrieval earliness), *RL* (retrieval lateness), *STL* (storage lateness) are also recorded. Next, we check whether there are moves left in the set $Order_m$. If NO, the algorithm terminates. Otherwise, the set PM_m is updated by deleting moves fixed and adding new moves from $Order_m$. With updated PM_m, sets NJ_{mn}, PLJ_{mn} etc. are updated accordingly and the algorithm continues until all moves in $Order_m$ are scheduled.

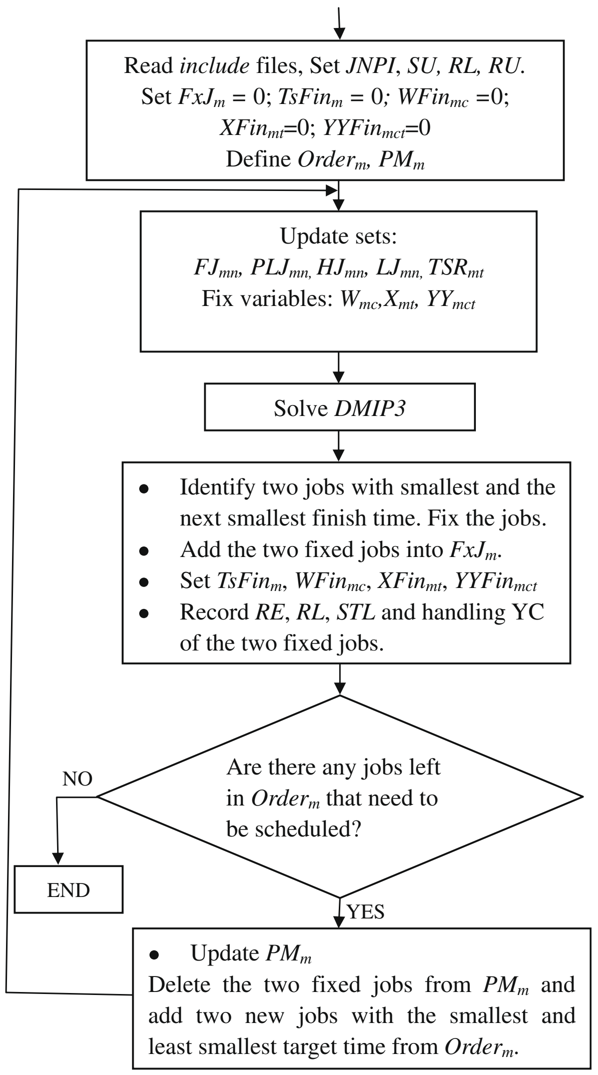
Read *include* files, Set *JNPI*, *SU, RL, RU*.
Set FxJ_m = 0; $TsFin_m$ = 0; $WFin_{mc}$ =0; $XFin_{mt}$=0; $YYFin_{mct}$=0
Define $Order_m$, PM_m
Update sets: FJ_{mn}, PLJ_{mn}, HJ_{mn}, LJ_{mn}, TSR_{mt}
Fix variables: W_{mc}, X_{mt}, YY_{mct}
Solve *DMIP3*
• Identify two jobs with smallest and the next smallest finish time. Fix the jobs.
• Add the two fixed jobs into FxJ_m.
• Set $TsFin_m$, $WFin_{mc}$, $XFin_{mt}$, $YYFin_{mct}$
• Record *RE*, *RL*, *STL* and handling YC of the two fixed jobs.
Are there any jobs left in $Order_m$ that need to be scheduled?
NO
END
YES
• Update PM_m
Delete the two fixed jobs from PM_m and add two new jobs with the smallest and least smallest target time from $Order_m$.

Fig. 6. Rolling-horizon algorithm.

5.3. Results of DMIP3

The results of the model with rolling-horizon algorithm are shown in Table 4. For *EX1* described in Section 3.4, *DMIP3* involves 74 binary variables, 617 single variables and 1606 single equations at the first iteration. *DMIP3* obtains the solution for *EX1* in 3.1 seconds with a total PM waiting time of 106.647 minutes which is also optimal. *DMIP3* obtains a near optimal solution in 3.9 seconds on average for all scenarios tested. This is much faster than the solution time (444.72 seconds on average) required in the paper of Ng and Mak (2005) solving for 25 moves in a block. *DMIP3* is very robust because the maximum solution time is only 9.5 seconds. By applying the lower bound evaluation procedure proposed by Ng (2005), the average/maximum/minimum *TCT* gaps for pure storage scenarios are 5.2%/16.6%/3.0%. Again, this is smaller than the current literature. Because TCT_{LB} is usually 1–3% lower than the optimal *TCT*, the actual average *TCT* gap for pure storage scenarios is less than 5.2%. A noticeable feature of *DMIP3* is that it finds optimal solutions for all scenarios with pure storage moves in only 3.0 seconds on average. From the 60 scenarios tested, the average/maximum/minimum ratio of the optimal *DMIP3* objective to the optimal *DMIP1* objective is 2.6%/26.2%/0.0%. We note that *DMIP3* obtained optimal solutions for 49 out of the 60 tested scenarios.

Although the decrease in size is accompanied by an increase in the number of sub-problems to solve, the total solution time still reduces dramatically. The results suggest that it is much better to solve a larger number of sub-problems than to solve the MILP model as a whole. As the number of jobs involved at each iteration is set to a small value, the model size of each iteration is very small. As the total problem size increases, the total solution time only increases linearly in n, i.e. $O\left(\frac{n}{JFPI} * e^{JNPI}\right)$. With *JNPI* much smaller than *n*, the exponential term, e^{JNPI}, is much smaller compared to e^n.

Table 4
Results of *DMIP3*

Scenario	Solution time, seconds			Gap, %
	Mean	Maximum	Minimum	$(TCT-TCT_{LB})/TCT_{LB}$
40 slots	3.6	9.5	0.7	NA
60 slots	4.3	7.5	1.6	NA
Pure storage moves	3.0	6.4	0.7	5.2
Storage + retrieval	4.6	9.5	1.7	NA

6. Conclusion

This paper develops efficient models for container YC work schedules. By applying heuristics and a rolling-horizon algorithm, we show that the model size is greatly reduced systematically and the solution time is shortened from days to seconds. The algorithm yields higher solution quality in a very short time compared to other heuristics used in the literature. The proposed formulations and algorithms can be extended to problems with time windows and space constraints such as quay crane scheduling or berth allocation (Wang and Lim, 2007). For future research, we suggest developing robust YC scheduling models by taking into account uncertain PM arrival times. Another future research development can consider the effect of landside conditions.

Acknowledgement

We thank the two anonymous referees for their comments and suggestions which have helped to improve this paper.

Appendix I. Definition of sets and parameters

The notations used in the mathematical formulation are as follows:

(a) Indices

c, c'	yard cranes, c, $c' = 1,2,\ldots,C$
m, n	container moves, m, $n = 1,2,\ldots,M$
SL	slot number, $SL = 1,2,\ldots,TSL$
t	time interval

(b) Sets

C	number of YCs working in the CY
FJ_{mn}	moves n that are TVL slots away from move m
FxJ_m	jobs that have been fixed after the current iteration and all previous iterations
HJ_m	includes moves located at the last SEP slots in CY
LJ_m	includes moves located at the first SEP slots in CY
M	set of container moves to be scheduled
$Order_m$	all jobs sorted by their target times
PLJ_{mn}	moves that locate on the left side of move m and moves that locate at the right side of move m but their slot difference against move m are less than SEP
PM_m	moves involved in a sub-problem of rolling-horizon algorithm
R_m	set of yard retrieval moves
S_m	set of yard storage moves (the total number of items in R_m and S_m is M)
T	set of time intervals scheduled
TSL	number of slots in the CY
XS_{mt}	time intervals t which are earlier than the target time of storage move m
TSR_{mt}	time intervals t which belong to job handling range of storage or retrieval move

(c) Parameters

$IntLen$	length of time interval
SU, RU, RL	parameter used to defined the job handling range
$JNPI$	job number per iteration
$MaxH$	the maximum target time of moves.
NC	total number of YCs scheduled
NSL	total number of slots in the CY
NT	total number of time intervals in set T
SEP	minimum difference in slot numbers allowed for two YCs at the same time. SEP is assumed to be 8 slots in this paper
$slot_m$	slot number where move m takes place. Without loss of generality, we assume that if $m < n$, then $slot_m \leqslant slot_n$
TVL	maximum number of slots an YC can gantry in half minute (8 is used here)
Tgt_m	target time for move m in the CY. For retrieval moves, this is the latest job start time that meets the deadline set by the QCs. For storage moves, this is the earliest job start time following the release of the job in the CY
$TsFin_m$	job start time (Ts_m) of moves that have been fixed
W_{re}	weight assigned to total retrieval earliness in the objective function
W_{sd}	weight assigned to total storage delay in the objective function
W_{rd}	weight assigned to total retrieval delay in the objective function
$WFin_{mc}$	the value of W_{mc} for moves that have been fixed
$XFin_{mt}$	the value of X_{mt} for moves that have been fixed
$YYFin_{mct}$	the value of YY_{mct} for moves that have been fixed

(d) Variables

RE_m	amount of retrieval earliness for move m
RL_m	amount of retrieval lateness for move m
STL_m	amount of storage lateness for move m
TC	the linear combination of retrieval earliness and storage and retrieval delays. It is the objective value to be minimized
W_{mc}	0–1, binary variable to denote if container move m is assigned to YC c
X_{mt}	0–1 variable to denote if move m is scheduled to take place during time interval t
YY_{mct}	0–1, continuous variable to denote if YC c handles moves m at time interval t

References

Bish, E.K., 2003. A multiple-crane-constrained scheduling problem in a container terminal. European Journal of Operational Research 144, 83–107.
Chen, L., Bostel, N., Dejax, P., Cai, J., Xi, L., 2007. A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. European Journal of Operational Research 181, 40–58.
Cheung, R.K., Li, C., Lin, W., 2002. Interblock crane deployment in container terminals. Transportation Science 36, 79–93.
Froyland, G., Koch, T., Megow, N., Duane, E., Wren, H., 2008. Optimizing the landside operation of a container terminal. OR Spectrum 30, 53–75.
Henwood R., 2006. The practitioner's definitive guide: Seafreight forwarding, SNP Reference.
Hopp, W.J., Spearman, M.L., 2000. Factory Physics: Foundations of Manufacturing Management. Irwin/McGraw-Hill.
Kim, K.H., Kim, K.Y., 1999. An optimal routing algorithm for a transfer crane in port container terminals. Transportation Science 33, 17–33.
Kim, K.Y., Kim, K.H., 2003. Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals. Naval Research Logistics 50, 498–514.
Kim, K.H., Park, Y., 2004. A crane scheduling method for port container terminals. European Journal of Operational Research 156, 752–768.
Li, J., Leung, S., Wu, Y., Liu, K., 2007. Allocation of empty containers between multiports. European Journal of Operational Research 182, 400–412.
Lim, A., Rodrigues, B., Xu, Z., 2007. A m-parallel crane scheduling problem with a non-crossing constraint. Naval Research Logistics 54, 115–127.
Narasimhan, A., Palekar, U.S., 2002. Analysis and algorithms for the transtainer routing problem in container port operations. Transportation Science 36, 63–78.
Ng, W.C., 2005. Crane scheduling in container yards with inter-crane interference. European Journal of Operational Research 164, 64–78.
Ng, W.C., Mak, K.L., 2005. Yard crane scheduling in port container terminals. Applied Mathematical Modelling 29, 263–275.
Ng, W.C., Tsang, W.S., 2005. Scheduling yard crane in a port container terminal using genetic algorithm. In: The First International Conference on Transportation Logistics (T-Log 2005) 27–29 July, Singapore.
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, I.F.A., Koster, R.D., 2003. Transshipment of containers at a container terminal: An overview. European Journal of Operational Research 147, 1–16.
Wang, F., Lim, A., 2007. A stochastic beam search for the berth allocation problem. Decision Support Systems 42, 2186–2196.
Zhang, C., Wan, Y., Liu, J., Linn, R.J., 2002. Dynamic crane deployment in container storage yards. Transportation Research B 36, 537–555.