**Submitted to *Transportation Science***
**manuscript TS-2013-0176.R1**

# Scheduling Multiple Yard Cranes with Crane Interference and Safety Distance Requirement

Yong Wu*
Department of International Business and Asian Studies, Griffith University, Gold Coast Campus, QLD 4222 Australia.

Wenkai Li
Graduate School of International Management, International University of Japan, Japan.

Matthew E. H. Petering
Industrial and Manufacturing Engineering Department, University of Wisconsin-Milwaukee, USA.

Mark Goh
NUS Business School and The Logistics Institute – Asia Pacific, National University of Singapore,
& School of Business IT & Logistics and Platform Technologies Research Institute, RMIT University, Melbourne, VIC 3000 Australia.

Robert de Souza
The Logistics Institute – Asia Pacific, National University of Singapore, 21 Heng Mui Keng Terrace, #04-01, Singapore 119613.

Container terminals require robust scheduling algorithms for yard cranes to optimally determine the sequence of storage and retrieval operations in yard blocks for higher container terminal performance. This paper investigates the multiple yard crane scheduling problem within a generic yard block and considers the operational restrictions such as the crane non-crossing constraint and models the crane travel time realistically. Further, the fact that any two adjacent cranes must keep an operational safety distance is also taken into consideration. These physical constraints limit the mobility of yard cranes and greatly render the scheduling difficulty for such pieces of equipment.

This paper proposes a clustering-reassigning approach, which fully considers all the operational constraints in practice. The complexity of the approach is $o(n^3)$, where $n$ is the number of container moves to be scheduled, makes it suitable for real time scheduling. Numerical experiments and benchmark with a continuous time based mixed integer linear programming (MILP) model indicate that the clustering-reassigning approach can provide satisfactory near optimal solutions for different sets of test cases in a real time scheduling context.

*Key words*: crane scheduling; heuristics; container yard; crane interference; safety distance

## 1. Introduction

Container terminals serve as important nodes in global supply networks. Together with the increasing global trade, the associated container volume around the globe has been increasing over the years and is predicted to continue grow in the future. In order to meet the demand of increasing

* Corresponding author. Phone: +61-7-5552-9116, Email: yong.wu@griffith.edu.au.

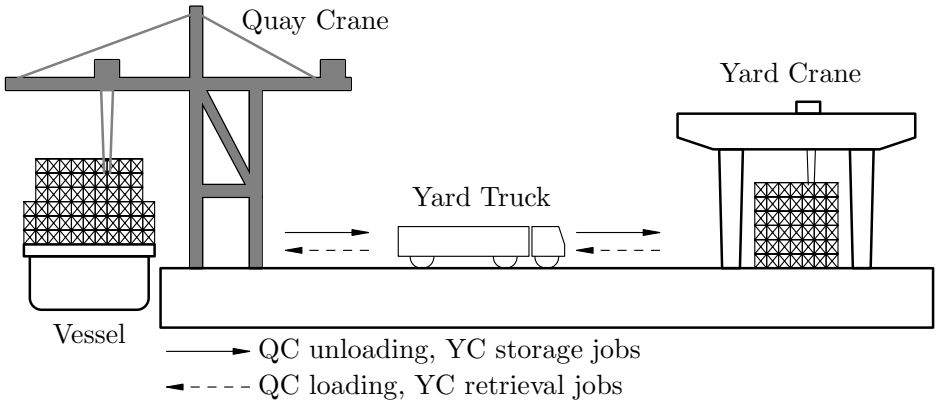


**Figure 1** **Typical container flows in terminal operations.**

container volume, the capacity of container ships has been steadily increasing as well. For example, the number of container ships with a capacity over 10,000 TEUs was 35 in July 2009, 105 in August 2011, and the number has jumped to 180 in June 2013 (Alphaliner 2009, 2011, 2013). With a similar trend, the average ship capacity for major shipping liners also increased over the years. For example, the world top 10 container fleets' average capacity shifted from 3,539 TEU/ship in July 2009, to 3,933 TEU/ship in August 2011, and to 4,432 TEU/ship in June 2013 (Alphaliner 2009, 2011, 2013). The increasing container volume around the world coupled with larger and larger container ships exert a lot of pressure on the operational efficiency of container terminals.

Container terminals, which are an essential part of container ports, play an important role in global maritime shipping industry by serving as a multi-modal transportation interface. A container yard inside a container terminal is basically a place for temporary storage of incoming and outgoing containers to facilitate the container flow. There are different types of equipment employed at a container terminal across the quay side and the yard: typical equipment includes quay cranes, yard cranes or straddle carriers, yard trucks or freight trains when the terminal has rail connections and accessibility. Quay cranes are responsible for loading and unloading containers to and from container liners; yard cranes are responsible for storing and retrieving containers into and out of container yard blocks; and yard trucks are used to bridge container movement between quay cranes and yard cranes, as well as transporting containers into and out of the container terminal.

Some container terminals, such as Port of Hong Kong and PSA Corporation Singapore Terminal, serve as transshipment facilities. In these terminals, most of the incoming containers, if not all, will stay within the terminal and wait for transshipment to other container ships later. Such terminals play an indispensable role in facilitating international trade as they allow the reconfiguration of container flows, which is a prerequisite to the deployment of large container ships. Figure 1 illustrates typical container flows in a transshipment container terminal. There are two major flows of containers between the yard and the quay side: quay crane unloading, yard crane storage

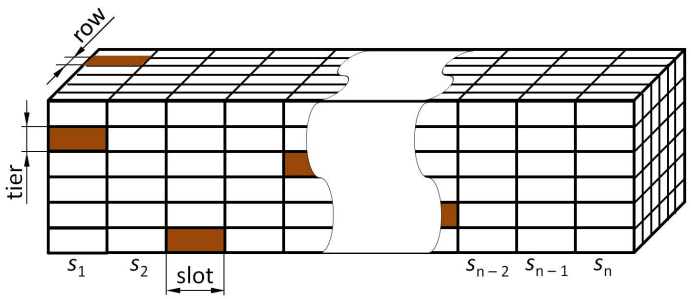


**Figure 2 A container block and its slot, row, and tier.**

container flow, and yard crane retrieval, quay crane loading container flow. Due to the nature of transshipment, the two flows usually happen at the same time for any particular container ship and therefore the two flows need to be coordinated carefully to serve container liners. Furthermore, efficient supply network operations require those containers to be handled as quickly as, and stay as short as possible in these transshipment terminals. Improving the operational efficiency of a container terminal will not only increase the utilization of some expensive pieces of equipment, such as container ships, quay cranes and yard cranes; but also boost the terminal's competitiveness and achieve higher throughput without additional investments.

Normally, a container yard is divided into multiple contiguous rectangular blocks. Figure 2 illustrates a typical yard block, where the block is divided into 'slots' along its length, 'rows' along its width, and 'tiers' along its height. Therefore, a container can be stored at an exact place in the container yard by specifying the block number, followed by slot, row and tier numbers, respectively. A 'slot' in Figure 2 could refer to up to 36 containers as there are 6 rows and 6 tiers in this case. When a yard crane needs to handle a container movement, the yard crane should travel to the slot number associated with the job to initiate the handling. Hence, the slot numbers are sufficient for the dispatch of yard cranes.

At a container terminal, it is common to use the quay crane work rate as the key performance indicator since it directly affects the turnaround time of vessels and the terminal's throughput. As quay cranes and yard cranes are linked by yard trucks, the times of both yard crane retrieval and storage jobs can be determined by the quay crane loading and unloading sequences, respectively. At the quay side, quay crane unloading moves ('move' and 'job' will be used interchangeably in this paper) can be handled as long as there are empty yard trucks available for picking up the containers. Quay crane loading jobs, however, need to handle the containers according to the pre-determined loading sequences; therefore, sometimes quay cranes have to wait for the right containers to come. These quay crane loading containers correspond to yard retrieval jobs. As a consequence, any delay in yard retrieval jobs will result in quay crane waiting, which should be avoided if possible as they are directly linked with the terminal's key performance indicators. It is therefore reasonable to

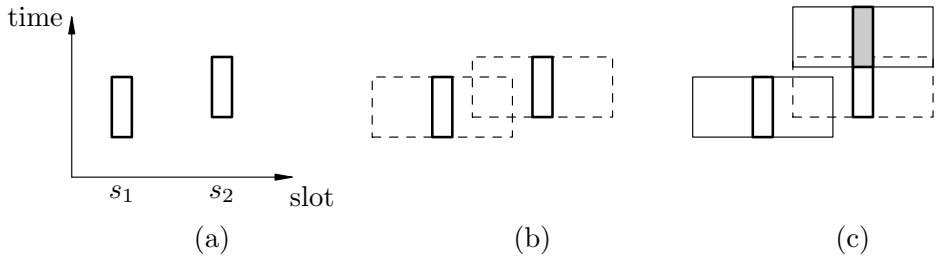


**Figure 3 Crane interference for non-overlapping jobs.**

request that quay crane loading jobs (e.g., yard crane retrieval jobs in the yard) should arrive at the quay side just in time or slightly ahead of quay crane schedules so that no quay crane disruption will occur. At the same time, it should be noted that the earlier a yard retrieval job is handled, the longer the yard truck will wait at the quay side (or nearby) to be served and might cause quay side congestion. Thus, we can conclude that while the quay crane disruptions (i.e., lateness of yard crane retrieval jobs) are undesirable, we also need to minimize the earliness of yard crane retrieval jobs. On the other hand, for yard storage jobs, the earliest time they can be handled is the time the yard trucks arrive at the destination yard slots with the right containers.

The establishment of ideal container handling times in the yard, in accordance with the quay crane loading and unloading sequence, makes it clear that the yard crane operations should target minimizing the deviation between the actual handling times and the ideal handling times. Considering the fact that any delay in the yard retrieval (i.e. for quay crane loading) jobs will directly affect quay crane schedules, there is a need to prioritize the yard retrieval jobs.

This paper investigates the scheduling of multiple yard cranes within one yard block while considering practical operational constraints. Yard cranes are bulky, slow-moving and expensive pieces of equipment. For yard blocks with frequent storage and retrieval operations, usually more than one yard crane will be employed. The physical features of yard cranes eliminate the possibility of one crane crossing over another in the same yard block for most container terminals. In addition, to ensure operational safety, certain distance (safety distance requirement, usually measured in terms of number of slots) must be kept between any pair of yard cranes operating at the same block. For example, assuming a safety distance of 8 slots and the slots are numbered from left to right and starting from $s_1$, as indicated in Figure 2, then a yard crane which is operating at slot $s_9$ will prohibit other yard cranes to access slots $s_1$ to $s_{17}$ as these slots fall into the safety distance requirement of this yard crane.

The yard crane safety distance requirement not only limits the workable area of yard cranes, but also increases the difficulty of the multiple yard crane scheduling problem. Figure 3 illustrates the impact of safety distance for two "non-overlapping" jobs: the "slot" axis specifies the locations (slots) of the jobs, and the "time" axis indicates the start time and duration of each job. The two

jobs in Figure 3a are not at the same slot. When considering the safety distance, the "space" a crane "holds" is much wider than the yard crane itself. The dashed rectangles in Figure 3b indicate the imaginary "spaces" the cranes should hold when considering the safety distance requirement. Hence, when scheduling the two jobs, one of the jobs must be postponed to ensure that there is no intersection between the dashed rectangles (i.e. to enforce the safety separation requirement) and enough time for one crane to gantry away without affecting the other one to avoid crane interference, which eventually might lead to the schedule generated in Figure 3c where one job has been postponed to the grey area and the solid rectangles indicate the actual handling operations.

The crane interference and safety distance requirement demand the scheduling algorithm not only to consider the job target times, proper prioritization of yard retrieval jobs, crane availability, but also the spatial requirement for multiple cranes in the same block and the crane travel time between consecutive jobs. We propose a clustering-reassigning approach for the problem and realistically factor in the actual crane travel time between jobs and the operational constraints. The algorithm is adaptive to any number of yard cranes, provides instant quality solutions for the multiple yard crane scheduling problem and therefore can be used for real time yard crane dispatching. In addition, it can be used to quickly assess the workloads of different yard blocks to facilitate the decision making on the inter-block yard crane assignment problem.

The remainder of this paper is organized as follows: Section 2 briefly reviews related work on existing yard crane scheduling methods; Section 3 presents the clustering-reassigning algorithm, followed by numerical experiments and algorithm performance analysis in Section 4 and the concluding Section 5.

## 2. Related Work

The uniqueness of container terminal operations management is receiving more and more attention due to the increasing importance of marine transportation. The most important objective for a container terminal is to increase its throughput, or in particular, to decrease the turnaround times of vessels (Steenken et al. 2004, Vis and de Koster 2003, Blazewicz et al. 2011). It is obvious that lower turnaround time depends heavily on the effectiveness and efficiency of allocating and scheduling key resources, such as berths, quay cranes, yard cranes and yard trucks, etc.

Several approaches have been proposed for scheduling cranes among yard blocks in container terminals, e.g., Chung et al. (2002), Linn et al. (2003) and Zhang et al. (2002). Those approaches focus on the problem of allocating yard cranes to different yard blocks in the entire terminal, rather than generating an operational schedule for any particular yard crane in a yard block and therefore is not the focus of this paper.

Many papers considered the scheduling of a single yard crane where containers are grouped and the crane must retrieve containers from specified groups according to a fixed sequence, without due

dates or release times, while minimizing travel distance. Since containers belonging to a specific group may be stored in multiple locations, both the yard crane route and number of containers picked up at each slot are decisions to be made (Kim and Kim 1997, 1999). For example, Kim and Kim (1999) studied the problem of routing a single yard crane to support the loading operation of a vessel. Ng and Mak (2005a,b) addressed the problem of scheduling a single yard crane carrying out a given set of container moves with different ready times and minimized the total job waiting times as the objective. A heuristic approach was developed in Ng and Mak (2005a); and its performance was benchmarked with a lower bounding algorithm. A branch and bound algorithm was then proposed in Ng and Mak (2005b). Guo et al. (2011) noted that the problem of yard crane scheduling is NP-hard and hence adopted a three level hierarchical procedure for yard crane operation management, in which the yard crane dispatching in individual zones happens at level 3. They then focused on the single yard crane dispatching and proposed two heuristic algorithms to minimize the average vehicle delay at the yard. A further study by Guo and Huang (2012) focused on the level 2 decision making, and proposed a time partition algorithm and a space partition algorithm to arrange the yard crane workload in a single row of yard blocks. Gharehgozli et al. (2014) adopted a continuous time integer programming model to tackle the single yard crane scheduling problem with both storage and retrieval jobs in a single container block. A two-phase solution method, which utilized the intrinsic properties of the model, was adopted to optimally solve the problem.

As it is common to observe more than one yard crane working in a yard block, studies on the scheduling of multiple yard cranes are essential for improving yard block operational efficiencies. Ng (2005) presented an integer programming model, a dynamic programming-based heuristic, and a lower bound to solve the problem of scheduling multiple yard cranes. While the crane interference was considered, there was no consideration of the safety distance requirement and no differentiation of the storage and retrieval jobs. Lee et al. (2007) considered the scheduling problem of two-transtainer systems. The container bay visit sequence and the number of containers to be picked up at each visit of the two transtainers were determined. A simulated annealing algorithm was developed to solve the proposed mathematical model but no detailed yard crane work schedules were generated. Cao et al. (2008) formulated a discrete time integer model and designed a combined scheduling heuristic for the double rail mounted gantry crane scheduling problem which involves different crane interferences. In this case, the two cranes are allowed to cross each other as their heights and widths are different, which is distinct from the problem we consider in this paper. Vis and Carlo. (2010) investigated the single block twin-crane scheduling problem, i.e., the same problem as in Cao et al. (2008), but with both storage and retrieval jobs. They constructed a mathematical model to minimize the make-span for both cranes and introduced an algorithm to

derive a lower bound for the make-span. A simulated annealing based heuristic was then proposed to solve the problem.

Most of the algorithms applied on the yard crane scheduling problem, such as Kim et al. (2004), Ng (2005) and Ng and Mak (2005b), only consider one type of jobs, presumably because it is easier to develop algorithms for homogeneous moves. Recently, Cao et al. (2010) solved an integrated yard truck and yard crane model for retrieval jobs using Benders' decomposition but they did not consider yard crane interference. Li et al. (2012), together with Li et al. (2009), addressed all the important operational constraints in a transshipment container terminal and provided high quality solutions with the help of mathematical optimization packages. Specifically, Li et al. (2009) developed discrete time mixed integer linear programming (MILP) models, which realistically considered the safety distance requirement and the impact of yard crane interference. Due to the number of variables involved, a rolling-horizon algorithm was applied to improve the solution quality and decrease solution time. Li et al. (2012) further improved the model by developing an efficient continuous-time MILP model and the solution quality and speed were greatly increased compared to the discrete-time MILP models.

This paper focuses on the same multiple yard crane scheduling problem in Li et al. (2009) and Li et al. (2012), where the problem has been investigated via a mathematical programming lens, and further extends the work through a heuristic approach. More specifically, the contributions of this paper can be summarized as follows:

1. *Significant improvement on the solution speed.* The heuristic approach proposed in this paper has a polynomial running time, which enables it to provide instant solutions. In contrast, the continuous time model in Li et al. (2012), though achieves significant improvements in terms of solution quality and speed over the discrete time models in Li et al. (2009), still requires noticeable running time (ranges from a few seconds to about twenty seconds), even with the aid of the rolling horizon technique. While this appears to be acceptable at individual yard block level, it might be difficult for the terminal operator to have a quick assessment of all the yard blocks in order to make higher level of decisions.

2. *Significant reduction in the solution time variability.* The heuristic approach helps address the solution time variability issue in a similar way. Stable solution time guarantees accurate estimation of the running time of a decision making module in the complex terminal operations management system. This is essentially important when the operator is dealing with on-the-fly scheduling and routing of various pieces of equipment, as the integer programming approaches usually suffer from solution variability, especially when dealing with large-size problems.

3. *Excellent scalability.* The proposed algorithm scales well when multiple yard blocks are considered simultaneously. This will help address the problem of scheduling yard cranes within a

particular yard zone, which is at a higher level than the yard blocks. Consequently, the three-level decision making process mentioned in Guo et al. (2011) could potentially become a two-level decision making process. The mathematical models, on the other hand, will be heavily affected as the number of container moves increases as both the numbers of decision variables and auxiliary variables will increase accordingly.

4. *Elimination of algorithm parameters.* The proposed algorithm completely eliminates the requirement of parameters; the only information required is the number of cranes to be deployed and the jobs to be handled. On the other hand, in the mixed integer programming approaches in Li et al. (2009) and Li et al. (2012), some parameters, such as *JNPI* and *JHR* in Li et al. (2012), need to be set or adjusted during the run, which subsequently require mathematical programming expertise.

5. *Independence of an optimization software package.* Approaches using mixed integer programming require the support of optimization software, such as CPLEX license. Commercial use of such licenses might be a cost factor that terminal operators need to consider.

6. *Provision of interfaces to the existing systems and other studies.* The use of a heuristic approach also makes it possible (and easier) to integrate with existing planning systems. As for the mathematical programming models, apart from the commercial licenses required, integrating commercial optimization software into a terminal planning system might be a cumbersome process and incur additional implementation and maintenance costs. By contrast, the heuristic approach can be easily implemented and integrated into existing planning systems. Furthermore, it is easy to combine the proposed heuristic solution with other studies, such as the one in Petering et al. (2009), to assess its performance. In addition, the heuristic approach can be used as a building block to assess different scenarios and situations at the terminal level, such as container storage planning, and inter-block yard crane cross gantry planning, where an accurate workload estimation is necessary. A workload estimation that is purely based on the number of jobs in a particular block might be a misleading indicator as these jobs might be clustered together and therefore the target yard block could not make use of additional yard cranes.

## 3. The Multiple Crane Scheduling Algorithm

For the multiple yard crane scheduling problem, usually there are many jobs to be handled within a planning horizon. As discussed earlier, the objective is to minimize the deviation between the actual handling time and the ideal handling time, while eliminate the retrieval job delays at the same time. We introduce the following notations to facilitate the description of the problem and the proposed heuristic algorithm:

Sets:

$C$: set of yard cranes.

$J$: set of container moves, indexed by $j$. The set $J$ can be divided into storage job set $J_S$ and retrieval job set $J_R$, based on the job types. Note that depending on the scheduling results, the retrieval job set $J_R$ can be further divided into a subset for jobs with retrieval earliness, $J_{RE}$, and a subset for jobs with retrieval lateness, $J_{RL}$.

Parameters:

$N_c$: total number of yard cranes working in the yard block; let $c_i$ denote yard crane $i$, $i = 1, 2, \ldots, N_c$;

$D_s$: safety distance in terms of number of slots allowed between two yard cranes. $D_s$ is assumed to be 8 slots in this paper;

$\check{T}_j$: planned/target job starting time for move $j$;

$H$: yard crane handling time of a container move; handling time is usually 2–4 minutes (Ng and Mak 2005b). $H$ is assumed to be three minutes in this paper;

$s_j$: slot number where move $j$ takes place, without loss of generality, we assume the numbering of slots starts from the left side of a yard block;

$G_{jj'}$: time for a yard crane to gantry from $s_j$ to next job at slot $s_{j'}$ after handles job $j$. The actual crane gantry time for one job to the next one is dependent on the location of the next job and therefore it will change if the schedules change. In this paper, $G_{jj'}$ is allocated realistically through considering factors such as acceleration and deceleration.

$s_{\max}$, $s_{\min}$: the maximum, and minimum slot numbers in a cluster, respectively; note that $s_{\max}$ and $s_{\min}$ will change if the jobs assigned to one cluster change;

Variables:

$h_{c_i j}$: binary variable indicating which crane $c_i$ handles which job $j$;

$T_j$: scheduled job starting time of container move $j$.

The multiple yard crane scheduling problem is in fact a multi-objective optimization problem, which was formulated into a single objective problem in Li et al. (2009, 2012) by assigning different weights to the retrieval lateness, retrieval earliness and storage lateness. Mathematically, the objectives can be expressed as:

$$\min \sum_{j \in J_{RL}} \left(T_j - \check{T}_j\right) \tag{1}$$

$$\min \operatorname{card}(J_{RL}) \tag{2}$$

$$\min \sum_{j \in J_{RE}} \left(\check{T}_j - T_j\right) \tag{3}$$

$$\min \sum_{j \in J_S} \left(T_j - \check{T}_j\right) \tag{4}$$

Objective function (1) tries to minimize the total retrieval lateness for all jobs that are scheduled later than the target time, while objective function (2) aims at reducing the cardinality of the set $J_{RL}$, i.e., the number of jobs that incur retrieval lateness. Objective functions (3) and (4) minimize the total retrieval earliness and storage lateness. Apparently, the following Equation (5) is also required to ensure that each job is handled by one and only one yard crane:

$$\sum_{c_i \in C} h_{c_i j} = 1, \quad \forall j \tag{5}$$

Obviously, other auxiliary variables and constraints are needed to ensure that the safety distance and crane interference are properly considered, and each crane can handle one job at a time, etc. However, as the focus of this paper is on the heuristic algorithm development, we will not discuss them here. Mathematical representation of these constraints can be found in Li et al. (2009, 2012).

One possible approach to tackle the multiple yard crane schedule problem is via concurrent scheduling, i.e., schedule jobs for all yard cranes at a particular time and always make sure that they meet all the operational constraints. Decisions need to be made at the beginning and end of a job and as time proceeds, all the jobs can be assigned to different yard crane and therefore generate a feasible schedule. The tricky part of this approach is the amount of retrieval earliness that should be assigned to a retrieval job in order to avoid a retrieval lateness, if possible. Therefore, this approach might need to proceed in a trial-and-error fashion to determine the appropriate amount of retrieval earliness and hence make it difficult to implement in reality. However, it should be mentioned that this approach might work well for homogenous container moves, especially for pure storage jobs.

On the other side, a divide and conquer approach could be utilized, where each yard crane is dealt with individually. Jobs could be allocated to different yard cranes and schedules can be generated for each of them. Scheduling for single yard crane eliminates the problem of safety distance requirement and yard crane interference and therefore, significantly reduces the complexity of the problem. When all the individual schedules are generated and put together, the overall feasibility of these schedules should be checked, i.e., to ensure no conflict of operational constraints. If there is a violation, necessary remedies, such as switching the orders of jobs, postponing a job, or re-allocation of a job to other cranes, need to be used which usually results in compromises. In addition, while it is relatively easy to generate individual schedules, resolving a conflict has an undesirable effect of chain reaction among all yard cranes and might make the overall feasible schedule hard to reach.

Realizing the drawbacks of both the concurrent scheduling and divide and conquer approaches, we focus on the operational characteristics of the multiple yard crane scheduling problem. From

the fact that the yard cranes are constrained by the safety distance and the order of cranes along the yard block is fixed, we can see that the service range of one particular crane at a specific time is actually constrained by its neighboring cranes. Therefore, it seems intuitive that each crane handles a relatively concentrated cluster of container moves; which not only determines the relative positions of yard cranes, but also reduces the distance yard cranes travel between consecutive jobs.

The clustering-reassigning approach follows this intuition. The clustering part of the algorithm clusters the job set into several subsets based on the locations (slots) of the jobs, where the number of subsets is equal to the number of cranes to be scheduled. Subsequently, each cluster is assigned to a crane and schedules are generated within each cluster. In this way, the cranes are guaranteed not to cross over each other when operating. However, due to the safety distance requirement, such schedules may not be feasible even when the clusters are not overlapping with each other. We devise a novel approach to deal with this problem if the initial schedules are interfering with each other. The reassigning part of the algorithm tries to reassign jobs among cranes and accepts new solutions when a better performance can be achieved.

Figure 4 schematically demonstrates the idea underlies the clustering-reassigning heuristic, where the horizontal axis refers to slots and the vertical axis refers to time. Imagine there are two yard cranes to be scheduled and each of them is assigned to half of the block: one is responsible for the left half and the other for the right half. Jobs at different times are plotted to show their potential impacts on the other crane. Again we adopt the representation in Figure 3 and plot $D_s/2$ slots on each side of the job to form a dashed rectangle to indicate the 'imaginary' size of a yard crane when interference is considered. It can be observed that jobs 6 and 7 are the rightmost and leftmost jobs for the left and the right cranes, respectively. For jobs in the right half of the block, if they are $D_s$ slots away from job 6, e.g., job 10, there is no way they can be affected by the left crane. For jobs within $D_s$ slots of job 6, namely jobs 7, 8, and 9, there will be crane interference if they are scheduled at a time overlaps with the duration of job 6. Similarly, jobs 5, 6 and 7 might be affected by the right crane, while jobs 1, 2 and 3 will never be affected by the right crane.

The important conclusion from Figure 4 is that once those "jobs could be affected by the other crane" are properly handled, the rest jobs would be relatively flexible to move along their corresponding crane schedules. This observation inspires the design of the clustering-reassigning approach.

It is obvious from Section 1 that yard storage operations can only be handled after the target start times. In the meantime, yard retrieval operations should be treated in such a way that they will just meet the quay crane working schedules, or with some retrieval earliness in order to avoid any retrieval lateness, as Equations (1) to (4) suggest. Assume the job set $J$, which includes the subset $J_R$ for retrieval jobs, and the subset $J_S$ for storage jobs, to be processed in the next planning

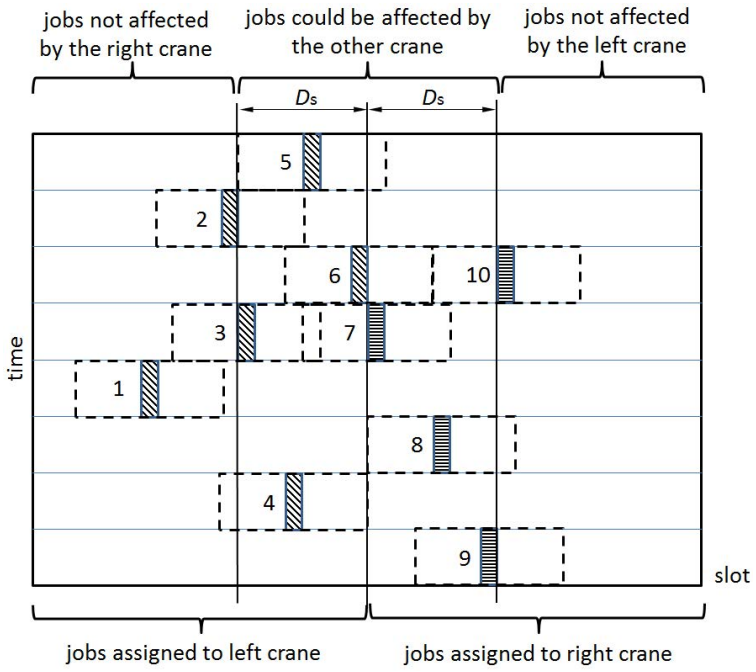


**Figure 4 The underlying concept of the clustering-reassignment heuristic.**

horizon has been given with target start times derived from quay crane schedules. Also assume trucks will always arrive on time for storage jobs and trucks are always available for retrieval jobs when requested. Before we proceed to describe the clustering-reassigning algorithm in detail, the following concepts are introduced.

*Operational continuity requirement* controls job starting times of consecutive container moves handled by the same yard crane. For two consecutive moves $j$ and $j'$ handled by the same yard crane, without loss of generality, we can assume move $j'$ happens after move $j$. Then job $j'$ can only start after the job finishing time of move $j$, $T_j + H$, plus the yard crane gantry time $G_{jj'}$ which allows yard crane to travel from slot $s_j$ to slot $s_{j'}$. This is referred as the operational continuity requirement.

*Earliest possible start time* refers to the earliest time any job can actually be handled. In the context of retrieval operations, this normally refers to the time '0', i.e., the start time of the planning horizon; for storage jobs, it is the target time the container arrives the slot it will be stored.

*Conflict-prone jobs* are defined as these container moves which might cause schedule conflict between any two adjacent yard cranes (i.e. clusters). For two clusters of jobs which will be handled by two adjacent yard cranes, the maximum slot number $s_{\max}$ in the left cluster (denote as $s^L_{\max}$) and the minimum slot number $s_{\min}$ in right cluster (denote as $s^R_{\min}$) can be identified, respectively. Those jobs with slot number greater than $s^R_{\min} - D_s$ in the left cluster and those with slot number less than $s^L_{\max} + D_s$ in the right cluster are defined as conflict-prone jobs since they are likely

to conflict the safety distance requirement. Those "jobs could be affected by the other crane" in Figure 4 are examples of conflict-prone jobs.

*Storage lateness* is the amount of time (delay) between the actual handling time and the targeted handling time for a storage job. Since storage job can only be handled after its target time, the minimum storage lateness is 0.

*Retrieval earliness* is the amount of time ahead of a retrieval job's targeted handling time. Retrieval earliness ensures the retrieval jobs will always be able to meet the quay cranes' schedules by arranging yard truck to arrive the quay side earlier. However, the amount of retrieval earliness should be kept as small as possible to minimize the waiting time of yard truck at the quay side.

*Retrieval lateness* is the amount of time (delay) between the actual handling time and the targeted handling time for a retrieval job. Retrieval lateness will directly affect the quay cranes' schedules and therefore should be avoided when possible.

We start to introduce the clustering-reassigning approach in detail from Algorithm 1 which makes schedule for a particular job cluster. To divide the jobs into clusters, the $K$-means clustering algorithm (Tou and Gonzalez 1974) is adopted since we already know the number of clusters (cranes). Within a job cluster, if the operations include only storage jobs or only retrieval jobs, the schedules are relatively easy to construct since all jobs are homogeneous. Here we assume that both types of operations exist in the cluster. The initial order of operations is determined according to their target start times $\check{T}_j$ and the jobs are inserted into a linear list $L$ starting with the earliest target start time job. As Algorithm 1 shows, retrieval operations are scheduled first from the end of the list $L$. We try to schedule the latest retrieval job to be handled just at its target start time, and schedule the rest retrieval jobs based on their target start times and the already scheduled retrieval operations in the list retrospectively. In this way, the retrieval operations are expected to be scheduled either just on time or with minimum retrieval earliness without incurring any retrieval lateness. After all retrieval operations are scheduled, the scheduled time of the first retrieval operation is checked. If the time is earlier than the earliest possible start time, the schedule needs to be adjusted so that the first retrieval job begins at the earliest possible start time, and the other retrieval operations satisfy the operational continuity requirement.

During this stage, although the order of jobs is not changed, the presence of storage jobs is actually ignored until all retrieval operations are scheduled. Once the start times of retrieval jobs are fixed, the storage operations are scheduled one by one from the beginning to the end of the list. The earliest storage job is picked and its start time is set according to the previous job finish time in the list. If there is enough space for this move to be handled, the start time is fixed; otherwise, it is switched with the next retrieval operation iteratively until it can be scheduled. The underlying concept is that we need to avoid retrieval lateness as much as possible.

**Algorithm 1** Generate a schedule for a particular cluster

**Require:** jobs assigned to the cluster, list $L$;

1: insert all jobs into list $L$ according to their target start times;
2: schedule the start time $T_j$ of the latest retrieval job $j$ in list $L$ as $\check{T}_j$;
3: **while** not reaching the beginning of list $L$ **do**
4: get the latest unscheduled retrieval job $j$;
5: **if** $\check{T}_j + H + G_{jj'} <$ start time of next retrieval job $j'$ **then**
6: schedule $T_j$ as $\check{T}_j$;
7: **else**
8: schedule $T_j$ so that $T_j + H + G_{jj'} =$ start time of next retrieval job $j'$;
9: **end if**
10: **end while**
11: **if** the first retrieval job start time is earlier than possible earliest start time **then**
12: adjust the retrieval jobs so that the first one can be done just at the earliest possible start time, and make the subsequent retrieval operations meet the operational continuity requirement;
13: **end if**
14: **while** there are unscheduled storage operations **do**
15: pick the earliest unscheduled storage operation $j$ in list $L$;
16: set the start time based on the previous job in the list;
17: **if** $T_j + H + G_{jj'} <$ start time of next retrieval job $j'$ **then**
18: keep the start time unchanged;
19: **else**
20: switch job $j$ with the next retrieval job $j'$, set it to be unscheduled, and adjust the schedule accordingly;
21: **end if**
22: **end while**

When putting all schedules generated by Algorithm 1 for all clusters together, it is very likely that there might be conflicts between schedules due to the safety distance requirement. Algorithm 2 describes the mechanism to resolve the conflicts. If the initial schedules are infeasible, those that are conflict-prone jobs are taken out and inserted into a temporary list $L_t$ where their scheduled start times are used to determine the order they enter the list $L_t$; the remaining moves are re-scheduled within each cluster and the resulting schedules are guaranteed to be feasible since no

**Algorithm 2** Resolving conflicts between schedules

**Require:** safety distance in terms of slots $D_s$; temporary list $L_t$;

clustering the jobs into $N_c$ clusters;
assign the clusters to cranes based on the location (order) of the clusters and call Algorithm 1 to generate the initial schedules within each cluster;
**if** the initial schedules are feasible **then**
  return;
**else**
  **for** each pair of adjacent clusters **do**
    identify and mark the conflict-prone jobs in the left cluster and right cluster, respectively;
    take out the marked moves and insert them into a temporary list $L_t$;
  **end for**
  **for** each crane schedule **do**
    call Algorithm 1 to re-generate schedules for each cluster;
  **end for**
  **while** $L_t \neq \varnothing$ **do**
    take the first job in the list $L_t$;
    insert the job back to its original cluster;
    postpone those in the original cluster to ensure the newly inserted one is scheduled on time;
    revise the earliest possible start time in $L_t$ based on the newly inserted job's finish time;
  **end while**
**end if**

conflict can happen at this time. Then jobs in the list $L_t$ are re-inserted back into its original cluster individually if there is enough time for the job to be handled; otherwise, the operations handled after that move are postponed so that an exact time slot can be available for the newly inserted job. The finish time of the move is then used to update the possible start time for all jobs remaining in the list $L_t$. This operation is repeated until the list $L_t$ is depleted. The basic idea of this conflict resolving mechanism is that those jobs that are not in the list $L_t$ are relatively free to be moved within the schedule since they do not affect the feasibility of all schedules, while those in the list $L_t$ are relatively inflexible because they are conflict-prone.

The reassigning part of the proposed multiple yard crane scheduling algorithm attempts to reassign jobs from one crane to its adjacent cranes' job lists and generates respective schedules, as shown in Algorithm 3. The criteria of $s_j > (s_{\max} + s_{\min})/2$ and $s_j < (s_{\max} + s_{\min})/2$ are used to

specify whether the move $j$ at slot $s_j$ should be considered for reassignment to left cluster and right cluster, respectively. Of course, other reassigning criteria can also be used subject to user specification. We use the sum of retrieval earliness, retrieval lateness and storage lateness as the performance indicator, i.e., $\sum_{j\in J_{RL}}\left(T_j-\check{T}_j\right)+\sum_{j\in J_{RE}}\left(\check{T}_j-T_j\right)+\sum_{j\in J_S}\left(T_j-\check{T}_j\right)$. Again, this can be easily changed subject to user specification. The new schedules are accepted if they generate better performance; otherwise, the original schedules are restored. The reassigning procedure tries to reassign jobs to cranes' right neighbors from leftmost crane to the right and then reverse the reassigning process from rightmost crane to the leftmost.

To illustrate the operating mechanism of Algorithms 1 to 3, we use a scheduling example for a rather congested set of jobs in a 40-slot yard block for two yard cranes. Figures 5 to 7 show the scheduling results of Algorithms 1, 2 and the reassigning Algorithm 3, respectively. Each job is represented as a rectangular box in the figures, where the width of the box equals to $D_s+1$ slots and the job is assumed to be located at the center slot of the box, the height of the box is the duration required to handle the move. Therefore, when two boxes are adjacent to each other, the distance between the two jobs is exactly the minimum distance allowed; any intersection of boxes indicates the cranes are interfering with each other and thus infeasible. The grey boxes with dashed lines around them are the original operations targeted to be handled; colored boxes with solid lines around them are the actual schedules, where each color represents one crane's schedule. Within each box, the number '0' or '1' at the left side indicates whether the job is handled by 'crane 0' or 'crane 1', the number in the middle of the box is the job number, and the letter 'S' indicates the operation is a yard storage one. The blank area in each solid box (if any) is the travel time that a yard crane needs to gantry to next job.

Figure 5 depicts the scheduling results for each cluster without considering the crane interference. It can be noticed that although individual schedules of the two cranes are feasible, they are interfering with each other when putting together: the boxes represent job 9 and job 15, job 12 and job 14 are intersecting. The conflict resolving mechanism described in Algorithm 2 helps generate the schedules in Figure 6 which resolves the problem of interference. It is noted that the job 16 and job 25 have been switched and job 12 has been postponed in order to generate feasible schedules. Figure 7 shows the results of reassigning. It can be observed that job 14 has been reassigned from crane 1's list to crane 0's list.

The time complexity of the clustering-reassigning algorithm can be analyzed following (Sipser 1997, ch. 7). The overall approach is in polynomial order of $n$, where $n$ is the number of jobs to be scheduled, i.e., the cardinality of the job set $J$. Specifically, the complexity of clustering of jobs using the $K$-means approach is in the order of $O(n)$ as we can safely assume that $N_c$ is far less than $n=\text{card}(J)$. For Algorithm 1, the scheduling of retrieval jobs is again in the order of $O(n)$

**Algorithm 3** Reassigning

**Require:** schedules generated based on Algorithm 2; number of yard cranes $N_c$;

**for** $i = 1$ up to $N_c - 1$ **do**
  **for** each job $j$ in the list of crane $C_i$ **do**
    **if** $s_j > (s_{\max} + s_{\min})/2$ **then**
      reassign the job $j$ to $C_{i+1}$;
      re-schedule all crane job lists;
      **if** better performance can be achieved **then**
        keep the reassignment results;
      **else**
        restore to the schedules before reassignment;
      **end if**
    **end if**
  **end for**
**end for**
**for** $i = N_c$ down to 2 **do**
  **for** each job $j$ in the list of crane $C_i$ **do**
    **if** $s_j < (s_{\max} + s_{\min})/2$ **then**
      reassign the job $j$ to $C_{i-1}$;
      re-schedule all crane job lists;
      **if** better performance can be achieved **then**
        keep the reassignment results;
      **else**
        restore to the schedules before reassignment;
      **end if**
    **end if**
  **end for**
**end for**

as retrieval jobs are handled in a linear way from the end of the list. The storage jobs, on the other hand, might increase the complexity to $O(n^2)$ as a storage job might need to switch places with retrieval jobs and therefore increases the complexity. The conflict resolving Algorithm 2 again has the complexity of $O(n^2)$ as when conflict-prone jobs are re-insert back to original lists, the jobs after the re-insertion point are pushed back, which is a linear operation. Therefore, the whole

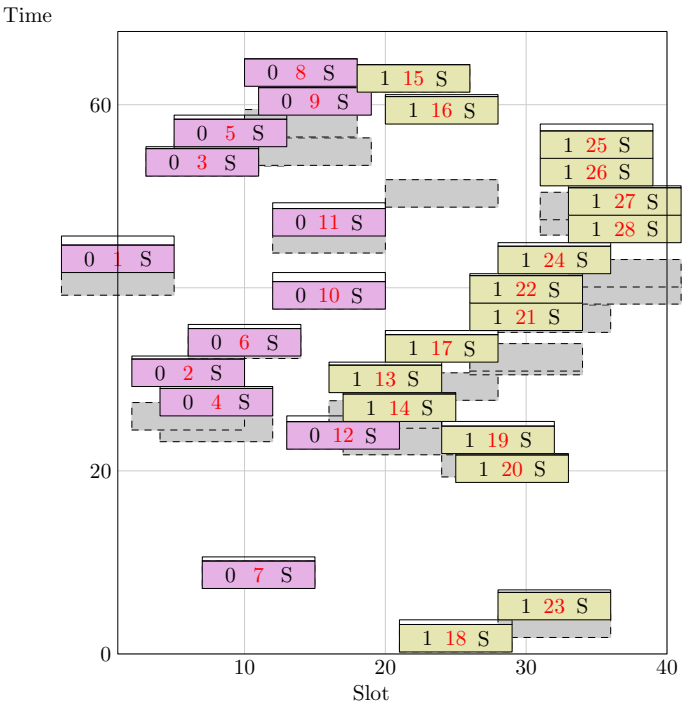


**Figure 5** **The initial schedules generated based on cluster information (Algorithm 1)**

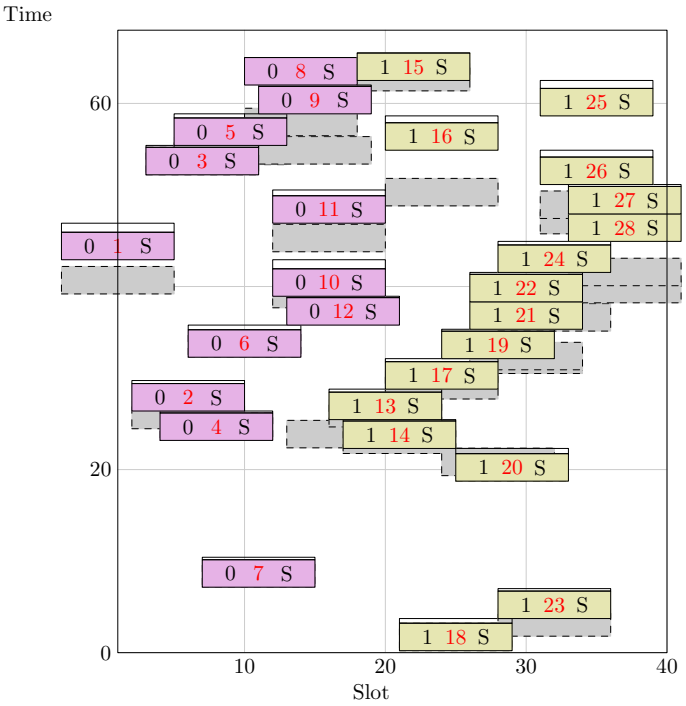


**Figure 6** **Feasible schedules generated by Algorithm 2**

clustering part of the algorithm has the complexity of $O(n^2) + N_c O(n^2) + O(n) \sim O(n^2)$, where $N_c$ is the number of cranes to be scheduled. In the reassigning Algorithm 3, each job has the possibility to be reassigned to its left or right neighboring cranes and the whole schedules are re-generated;

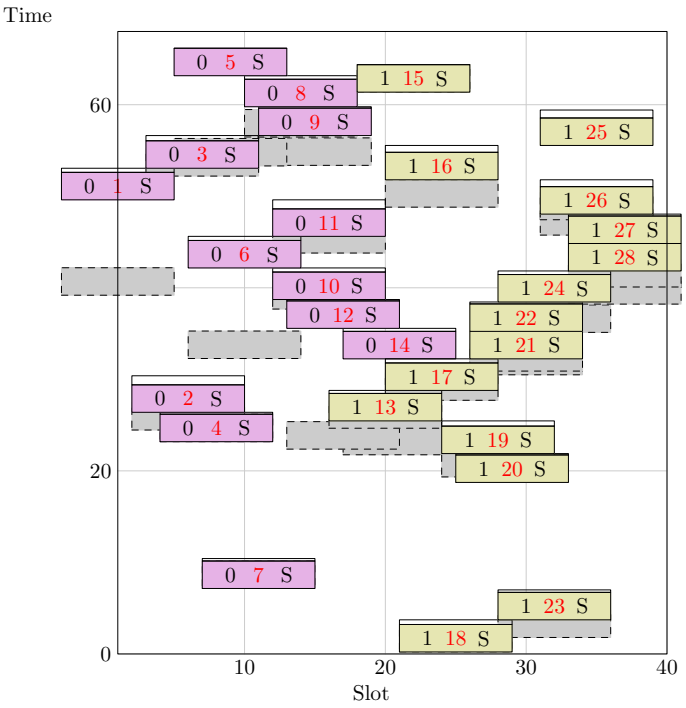


**Figure 7 Final schedules after reassigning (Algorithm 3)**

therefore, the complexity of the clustering-reassigning algorithm would be $O(n^2)n \sim O(n^3)$. It should be noted that not every job is tried to be reassigned; hence, the complexity can be further written as $o(n^3)$. This makes the algorithm suitable for online scheduling.

# 4. Numerical Experiments

## 4.1. Performance measurement

It is rather tricky to use performance measurement for yard crane scheduling when both retrieval and storages jobs are presented. For pure storage container handling jobs, it seems natural to use the total yard truck waiting time as a performance indicator since this measures the time effectiveness of handling of storage jobs. On the contrary, for pure retrieval jobs, while minimizing the total retrieval earliness is one of the objectives, another more important objective is to avoid retrieval lateness and the number of jobs to be delayed. It then will be possible that, in order to avoid a retrieval delay, a relatively large retrieval earliness might occur. Therefore, using the total yard truck waiting time is not going to be an appropriate performance measurement indicator. How to measure the performance of scheduling algorithms becomes critical for comparison with other approaches.

There are a few performance measurement indicators deployed in the literature. Ng (2005) used the total job completion time,

$$\mathrm{TCT} = \sum_{i \in J} (T_i + H),$$

**Table 1 Summary of total yard truck waiting time in minutes**

| Schedule | Total yard truck waiting time | Status |
|---|---|---|
| Figure 5 | 95.9 | Infeasible |
| Figure 6 | 110.8 | Feasible |
| Figure 7 | 103.6 | Feasible |

which is similar to measure total yard truck waiting time but avoids the problem of yard truck waiting time of 0 when making comparison. Table 1 describes the total amount of yard truck waiting time for the three schedules generated at different stages by the clustering-reassigning approach for the illustrative example. Since the jobs are all storage ones, the total yard truck waiting time shows the summation of handling time deviation from scheduled time to target time. Figure 5 holds the least amount of total yard truck waiting time but the schedules generated are infeasible. After the interference is resolved, the total amount of yard truck waiting time increases to 110.8 minutes and then decreases to 103.6 minutes after the reassigning operation. It is reasonable that the final total yard truck waiting time is greater than that in Figure 5 since compromise need to be made among cranes when resolving conflicts. However, this measurement is only applicable to pure storage jobs. If it is applied to retrieval jobs, then getting the retrieval jobs done as early as possible will improve the performance if such performance measurement is used.

Li et al. (2009, 2012) investigated the same research problem as of this paper, and therefore can be used for benchmarking purpose. The linear combination of storage lateness, retrieval earliness and retrieval lateness was used as the objective of the MILP model; where the retrieval lateness was given a higher weight. It is clear that the eventual results will depend on the weight selection for test cases with retrieval jobs as it is hard to set a definite weight for retrieval lateness. The total completion time (TCT) was used in Li et al. (2012) to report the results. To maintain consistency, we adopt the same performance measurement when possible.

### 4.2. Test cases

We conduct numerical experiments on 90 test cases, which were generated based on the simulation study by Petering et al. (2009). The majority of these test problems have been used in Li et al. (2009) and Li et al. (2012), which facilitate performance comparison. Different yard operation cases are considered: pure storage, pure retrieval and mixed storage/retrieval jobs. Both 40-slot and 60-slot yards are examined. The workload of different cases is also controlled by specifying the number of jobs to be handled in each case. For each case, the target handling time and the slot number of each job are given. Table 2 summarizes all the test cases and then classifies these test cases in terms of pure storage, pure retrieval, and mixed storage and retrieval operations. Overall, we have 90 test cases, 44 of them are 40-slots cases, and 46 are 60-slot cases. Altogether there are 48 pure storage test cases, 13 pure retrieval cases, and 29 mixed storage/retrieval cases. In this

**Table 2 Summary and classification of test cases**

| Num. of cases | Block Dim. | Pure Storage | Pure Retrieval | Mixed | Workload | Jobs per case |
|---|---|---|---|---|---|---|
| 30 | 40-slot | 11 | 5 | 14 | Light | 19 – 33 |
| 14 | 40-slot | 11 | 1 | 2 | Heavy | 41 – 69 |
| 32 | 60-slot | 15 | 6 | 11 | Light | 19 – 32 |
| 14 | 60-slot | 11 | 1 | 2 | Heavy | 50 – 66 |
| 90 Total | | 48 | 13 | 29 | | |

paper, each test case is coded with its block dimension (40 or 60), workload (L for light and H for heavy), case type (S for pure storage, R for pure retrieval, and M for mixed) and a case number. For example, 40-L-M03 refers to a 40-slot, light workload, mixed retrieval/stroage test case, which is numbered as '3'.

### 4.3. Comparison with the MILP model

The MILP model developed in Li et al. (2012) is used as the benchmark and solved with GAMS/CPLEX 12.6.1; the clustering-reassigning approach is coded in C++. Both approaches are tested on Windows XP with a 2.2 GHz Pentium processor and 2GB RAM.

Table 3 reports the total completion time (TCT) for the clustering-reassigning approach, and the MILP model, respectively. The solution time required by the MILP model is also listed. The performance gap, as a percentage, is calculated using the MILP model as the benchmark, i.e., $\text{Gap} = (\text{TCT}_{\text{Clustering-Reassigning}} - \text{TCT}_{\text{MILP}})/\text{TCT}_{\text{MILP}}$. It can be observed that the performance difference between the clustering-reassigning approach and the MILP model developed in Li et al. (2012) is very narrow. There are 6 cases where both approaches provide exactly the same results, and 33 out of the 48 cases are able to maintain a 1.0% performance gap with the MILP model.

Table 4 summarizes the performance of the clustering-reassigning approach on the pure storage test cases compared with the results provided by the MILP model in Li et al. (2012). For light workload test cases in 40-slot blocks, the maximum performance gap is 0.39% while the minimum performance gap is 0.0% which indicates that the clustering-reassigning algorithm achieves similar results as the MILP model does, the average performance gap is 0.13%. The similar trend can be observed for the 60-slot blocks with light workload as well, with a maximum performance gap of 2.62% and minimum −0.16% (which implies that sometimes the clustering-reassigning approach provides slight better results than the MILP does). The average performance gap for light workload 60-slot test cases is 0.59%. When the workload gets heavier, the clustering-reassigning algorithm's performance is moderately affected. For 40-slot test cases, the maximum performance gap is 3.38%, with an average gap of 1.35%; for 60-slot test cases, the maximum performance gap is 4.25%, with an average gap of 1.36%. Over all test cases, the average performance gap is 0.83% which demonstrates the clustering-reassigning approach provides satisfactory results compared with the MILP model.

**Table 3 Performance comparison for pure storage job cases compared with the MILP model in Li et al. (2012)**

| Case | Clust.-Reassig. | MILP Model | MILP Time (s) | Gap (%) |
|---|---|---|---|---|
| 40-L-S01 | 1704.1 | 1703.3 | 1.8 | 0.05 |
| 40-L-S02 | 1841.4 | 1838.6 | 1.5 | 0.15 |
| 40-L-S03 | 1512.6 | 1512.2 | 1.3 | 0.03 |
| 40-L-S04 | 1505.1 | 1505.1 | 0.9 | 0.00 |
| 40-L-S05 | 2164.9 | 2164.8 | 2.1 | 0.00 |
| 40-L-S06 | 1353.9 | 1353.9 | 1.6 | 0.00 |
| 40-L-S07 | 1447.2 | 1445.9 | 1.5 | 0.09 |
| 40-L-S08 | 1740.7 | 1740.4 | 1.6 | 0.02 |
| 40-L-S09 | 1508.0 | 1502.6 | 2.0 | 0.36 |
| 40-L-S10 | 1823.3 | 1817.7 | 1.6 | 0.31 |
| 40-L-S11 | 1948.4 | 1940.8 | 1.6 | 0.39 |
| 40-H-S12 | 3650.3 | 3623.3 | 5.3 | 0.75 |
| 40-H-S13 | 3554.2 | 3521.9 | 4.8 | 0.92 |
| 40-H-S14 | 3746.1 | 3665.4 | 6.4 | 2.20 |
| 40-H-S15 | 3229.8 | 3201.2 | 3.9 | 0.89 |
| 40-H-S16 | 3361.5 | 3327.1 | 5.5 | 1.03 |
| 40-H-S17 | 3891.7 | 3813.9 | 5.2 | 2.04 |
| 40-H-S18 | 2978.6 | 2960.8 | 4.8 | 0.60 |
| 40-H-S19 | 2995.8 | 2986.9 | 5.0 | 0.30 |
| 40-H-S20 | 2999.4 | 2961.2 | 4.7 | 1.29 |
| 40-H-S21 | 4128.8 | 4070.7 | 6.6 | 1.43 |
| 40-H-S22 | 5155.5 | 4986.9 | 9.5 | 3.38 |
| 60-L-S01 | 2038.3 | 2018.1 | 2.4 | 1.00 |
| 60-L-S02 | 856.0 | 856.4 | 2.5 | −0.05 |
| 60-L-S03 | 1381.6 | 1383.8 | 1.6 | −0.16 |
| 60-L-S04 | 2095.6 | 2086.4 | 3.4 | 0.44 |
| 60-L-S05 | 1773.5 | 1757.4 | 2.0 | 0.92 |
| 60-L-S06 | 1900.4 | 1876.6 | 2.2 | 1.27 |
| 60-L-S07 | 2176.6 | 2176.6 | 1.6 | 0.00 |
| 60-L-S08 | 1870.3 | 1870.3 | 1.7 | 0.00 |
| 60-L-S09 | 1794.4 | 1794.4 | 1.9 | 0.00 |
| 60-L-S10 | 2212.7 | 2203.2 | 2.7 | 0.43 |
| 60-L-S11 | 1662.2 | 1661.5 | 2.8 | 0.04 |
| 60-L-S12 | 2139.4 | 2135.7 | 2.5 | 0.17 |
| 60-L-S13 | 2248.9 | 2245.8 | 2.4 | 0.14 |
| 60-L-S14 | 1274.9 | 1249.6 | 2.1 | 2.02 |
| 60-L-S15 | 1850.4 | 1803.2 | 3.3 | 2.62 |
| 60-H-S16 | 4179.8 | 4163.2 | 6.3 | 0.40 |
| 60-H-S17 | 3735.5 | 3713.9 | 5.6 | 0.58 |
| 60-H-S18 | 3829.5 | 3673.5 | 6.8 | 4.25 |
| 60-H-S19 | 4319.3 | 4245.0 | 8.0 | 1.75 |
| 60-H-S20 | 4095.3 | 4082.0 | 8.4 | 0.33 |
| 60-H-S21 | 4241.0 | 4214.1 | 9.4 | 0.64 |
| 60-H-S22 | 4525.7 | 4494.8 | 7.3 | 0.69 |
| 60-H-S23 | 4057.1 | 3966.8 | 6.1 | 2.28 |
| 60-H-S24 | 4721.8 | 4659.6 | 9.3 | 1.33 |
| 60-H-S25 | 3835.1 | 3772.7 | 7.1 | 1.65 |
| 60-H-S26 | 4660.8 | 4612.3 | 9.3 | 1.05 |

**Table 4 Performance gap for pure storage cases compared with Li et al. (2012)**

| | | Performance Gap (%) | | | MILP Time (s) | | |
|---|---|---|---|---|---|---|---|
| Block Dim. | Workload | Max | Min | Avg | Max | Min | Avg |
| 40-slot | Light | 0.39 | 0.00 | 0.13 | 2.1 | 0.9 | 1.6 |
| 40-slot | Heavy | 3.38 | 0.30 | 1.35 | 9.5 | 3.9 | 5.6 |
| 60-slot | Light | 2.62 | −0.16 | 0.59 | 3.4 | 1.6 | 2.3 |
| 60-slot | Heavy | 4.25 | 0.33 | 1.36 | 9.4 | 5.6 | 7.6 |
| Overall | | 4.25 | −0.16 | 0.83 | 9.5 | 0.9 | 4.1 |

**Table 5 Performance comparison for pure retrieval job cases compared with the MILP model in Li et al. (2012)**

| | Clustering-Reassigning | | MILP Solution | | |
|---|---|---|---|---|---|
| Case | Earliness | Lateness | Earliness | Lateness | Time (s) |
| 40-L-R01 | 13.4 | 0.0 | 16.6 | 0.0 | 4.1 |
| 40-L-R02 | 14.8 | 0.0 | 6.3 | 2.9 | 4.1 |
| 40-L-R03 | 129.1 | 0.0 | 65.8 | 4.5 | 5.1 |
| 40-L-R04 | 22.2 | 0.0 | 12.7 | 0.0 | 3.4 |
| 40-L-R05 | 7.9 | 0.0 | 7.8 | 0.0 | 2.7 |
| 40-H-R06 | 110.9 | 103.2 | 77.8 | 118.7 | 19.6 |
| 60-L-R01 | 92.2 | 0.0 | 60.0 | 0.0 | 3.6 |
| 60-L-R02 | 3.5 | 0.0 | 3.5 | 0.0 | 1.7 |
| 60-L-R03 | 16.9 | 0.0 | 14.2 | 0.0 | 4.1 |
| 60-L-R04 | 18.7 | 0.0 | 12.0 | 0.0 | 4.2 |
| 60-L-R05 | 20.0 | 2.9 | 20.0 | 2.9 | 7.0 |
| 60-L-R06 | 110.1 | 0.0 | 18.2 | 0.3 | 9.2 |
| 60-H-R07 | 447.1 | 18.8 | 147.0 | 168.2 | 32.8 |

Table 5 reports the total earliness and lateness on the pure retrieval job cases for both the clustering-reassigning approach and the MILP model. As discussed earlier, the total completion time TCT is not suitable for reporting performance in this case. The MILP model's solution time is also included. The results indicate that the clustering-reassigning approach treats the retrieval lateness with reasonable priority and incurs retrieval lateness for 3 cases out of the 13 pure retrieval test cases. On the other side, the MILP model results in 6 cases of retrieval lateness out of the 13 test cases. It should be noted that the MILP solution reported here has already gone through a trial-and-error process for the retrieval lateness weight selection in the objective function in Li et al. (2012). The original weight assigned in Li et al. (2012) was 2 and this seems to be not big enough to avoid retrieval lateness in most cases. Therefore, we increase the retrieval lateness weight to 100 when we observe there is retrieval lateness, and continue to increase it to 1000 if there is still retrieval lateness. We report the best solution found out of the the three options (weight of 2, 100, and 1000), and consequently, report the total solution time required to achieve such a result. It should be noted that even by doing so, the clustering-reassigning still beats the MILP model in terms of avoiding occasions of retrieval lateness. Of course, it should be pointed out that by doing so, it usually will result in larger amount of retrieval earliness.

**Table 6 Performance comparison for mixed job cases compared with the MILP model in Li et al. (2012)**

| | Clustering-Reassigning | | | MILP Solution | | | |
|---|---|---|---|---|---|---|---|
| Case | SL | RE | RL | SL | RE | RL | Time (s) |
| 40-L-M01 | 0.0 | 7.3 | 0.0 | 0.0 | 4.3 | 0.0 | 1.1 |
| 40-L-M02 | 17.9 | 3.0 | 0.1 | 11.3 | 3.3 | 0.0 | 3.9 |
| 40-L-M03 | 1.7 | 0.0 | 0.0 | 0.5 | 0.0 | 0.0 | 1.1 |
| 40-L-M04 | 24.2 | 0.0 | 0.0 | 24.2 | 0.0 | 0.0 | 2.2 |
| 40-L-M05 | 12.7 | 46.0 | 5.8 | 12.2 | 36.6 | 0.0 | 4.0 |
| 40-L-M06 | 49.4 | 37.5 | 0.0 | 19.7 | 41.6 | 0.0 | 4.2 |
| 40-L-M07 | 10.0 | 45.0 | 0.0 | 10.0 | 45.0 | 0.0 | 4.3 |
| 40-L-M08 | 44.4 | 45.7 | 0.0 | 29.3 | 7.0 | 19.8 | 8.0 |
| 40-L-M09 | 5.4 | 26.1 | 17.2 | 8.4 | 15.8 | 13.3 | 4.7 |
| 40-L-M10 | 29.9 | 30.0 | 0.0 | 18.1 | 34.7 | 0.0 | 3.8 |
| 40-L-M11 | 9.2 | 31.1 | 0.0 | 6.4 | 15.0 | 0.0 | 3.0 |
| 40-L-M12 | 4.1 | 5.0 | 0.0 | 4.1 | 3.2 | 0.0 | 1.4 |
| 40-L-M13 | 24.0 | 17.6 | 0.0 | 15.0 | 15.9 | 0.0 | 5.4 |
| 40-L-M14 | 16.9 | 0.0 | 0.0 | 8.3 | 6.6 | 0.0 | 3.5 |
| 40-H-M15 | 476.0 | 304.2 | 0.0 | 143.9 | 83.8 | 61.1 | 21.2 |
| 40-H-M16 | 63.8 | 167.8 | 7.0 | 11.8 | 105.2 | 3.4 | 12.9 |
| 60-L-M01 | 231.6 | 159.8 | 0.0 | 5.3 | 39.7 | 8.4 | 6.3 |
| 60-L-M02 | 28.5 | 69.4 | 0.0 | 0.0 | 60.0 | 0.0 | 4.1 |
| 60-L-M03 | 68.2 | 81.8 | 0.0 | 1.0 | 64.5 | 0.0 | 4.9 |
| 60-L-M04 | 85.8 | 216.2 | 0.0 | 3.7 | 39.0 | 4.5 | 7.4 |
| 60-L-M05 | 46.8 | 0.0 | 0.0 | 42.5 | 0.0 | 0.0 | 2.1 |
| 60-L-M06 | 7.4 | 8.5 | 41.2 | 29.7 | 21.1 | 12.5 | 5.5 |
| 60-L-M07 | 50.4 | 7.6 | 18.0 | 35.0 | 7.7 | 0.0 | 5.6 |
| 60-L-M08 | 34.7 | 21.7 | 6.5 | 11.8 | 14.3 | 0.0 | 3.8 |
| 60-L-M09 | 20.0 | 0.0 | 1.0 | 7.4 | 2.5 | 0.0 | 5.1 |
| 60-L-M10 | 30.0 | 1.5 | 0.0 | 17.6 | 4.2 | 0.0 | 5.2 |
| 60-L-M11 | 47.7 | 11.5 | 0.0 | 30.5 | 2.7 | 0.0 | 6.1 |
| 60-H-M12 | 307.3 | 35.1 | 296.0 | 473.0 | 6.6 | 317.4 | 85.6 |
| 60-H-M13 | 192.3 | 48.0 | 94.7 | 229.7 | 32.5 | 54.0 | 25.9 |

Similarly, Table 6 presents total storage lateness (SL), retrieval earliness (RE), and retrieval lateness (RL) on the mixed job test cases for both the clustering-reassigning approach and the MILP model, together with the solution time required for the MILP model. The MILP solution reported here also adopts the weight selection process as mentioned above. Again, the clustering-reassigning approach seems to be able to control the retrieval lateness well, and incurs 10 cases of retrieval delay out the 29 cases tested. The MILP model provides a comparable performance in this regard and results in 9 cases of retrieval delay. It is interesting to note that the clustering-reassigning approach's intrinsic priority set for retrieval lateness can generate quite different results from the MILP model. For example, in the cases 40-L-M08 and 40-H-M15, the clustering-reassigning approach avoids retrieval delay completely; however, the MILP accepts delays (even though we test with different weights on the retrieval lateness) and tries to gain benefits from less retrieval earliness and storage lateness.

In terms of solution time, on average the MILP model needs 2∼3 seconds for light workload cases and more than 5 seconds for heavy workload cases. Sometimes it needs more than one minute. Although the performance of the MILP model has been improved significantly compared to the discrete time model proposed in Li et al. (2009), the average time required indicates that the MILP model is still not very suitable for real time scheduling, along with other disadvantages, as we discussed in Section 2. The clustering-reassigning approach, on the other hand, provides solutions instantly and the running time is not practically reportable due to its polynomial complexity. The running time of these test cases included in this paper numerically confirms our analysis on the algorithm complexity and verifies that the approach is appropriate for real time scheduling.

### 4.4. Validation via simulation

In both the MILP model and the clustering-reassigning approach, although the yard crane travel time is modeled realistically, the handling time of each job is assumed to be fixed. In order to provide feasible solutions for practical usage, the handling time for each job is assumed to be a rather conservative constant amount of time when scheduling. To further validate the performance of the clustering-reassigning approach, simulated runs are conducted to check how well the algorithm will work if the generated schedules are strictly followed. That is, each job is handled according to the sequence generated by the clustering-reassigning algorithm, but the handling time is assumed to follow a known statistical distribution from a terminal operator. The schedules generated by the clustering-reassigning approach are tested by simulating the handling times and each case is checked for 10 runs.

Table 7 compares the total yard truck waiting time, for pure storage test cases, between the generated schedules and simulation runs. The total yard truck waiting time summarizes the time difference between the actual job handling and the target job handling times. The "Max", "Min" and "Avg" in the table stand for the maximum, minimum and average percentage of the simulated runs compared with the scheduled results. It can be observed from the table that the maximum percentages of total waiting time in the pure storage, light workload cases (both 40-slot and 60-slot) are equal to the scheduled results, while for the two batches of heavy workload test cases, the percentages are 76.83% and 85.98% respectively. A similar pattern can also be examined for the average percentages: in the light workload cases, the average percentages are closer to the scheduled results; while in the heavy workload cases, the deviation of the simulation results is large from the scheduled results. This is due to the fact that under light workloads, the total waiting time is quite small even when using a conservative handling time for scheduling since the generated schedules are quite "loose". However, for heavy workload cases, the schedules are normally quite "tight"; therefore some jobs are unnecessarily postponed due to the conservative estimation of the handling time, which in turn explains the huge reduction of total waiting time in simulation runs.

**Table 7 Comparison of total yard truck waiting times between simulation runs and generated schedules**

| Case Type | Blk Dim. | Workload | Max (%) | Min (%) | Avg (%) |
|---|---|---|---|---|---|
| Pure storage | 40-slot | Light | 100.00 | 11.92 | 77.72 |
| | 40-slot | Heavy | 76.83 | 42.27 | 58.35 |
| | 60-slot | Light | 100.00 | 43.40 | 80.91 |
| | 60-slot | Heavy | 85.98 | 40.16 | 58.35 |

**Table 8 Comparison of make-span between generated schedules and simulation runs**

| | | | Scheduled (%) | | | Simulated (%) | | |
|---|---|---|---|---|---|---|---|---|
| Case Type | Blk Dim. | Workload | Max | Min | Avg | Max | Min | Avg |
| Pure storage | 40-slot | Light | 114.71 | 100.00 | 101.70 | 114.01 | 100.00 | 101.42 |
| | 40-slot | Heavy | 110.11 | 100.00 | 103.06 | 106.06 | 100.00 | 101.57 |
| | 60-slot | Light | 100.73 | 100.00 | 100.14 | 100.64 | 100.00 | 100.02 |
| | 60-slot | Heavy | 113.92 | 101.47 | 105.42 | 111.80 | 100.00 | 103.72 |
| Mixed | 40-slot | Light | 114.71 | 100.00 | 101.82 | 114.67 | 100.00 | 101.46 |
| | 40-slot | Heavy | 110.06 | 100.00 | 105.03 | 106.70 | 100.00 | 102.86 |
| | 60-slot | Light | 117.13 | 100.00 | 104.05 | 104.98 | 100.00 | 101.05 |
| | 60-slot | Heavy | 106.66 | 100.00 | 103.33 | 104.95 | 100.00 | 102.02 |

Note for pure retrieval and mixed job test cases, the conservative estimation will only make the schedules easier to follow and therefore results in jobs been handled earlier if the crane operators (in the simulation) just follow the sequence of jobs, but not pay attention to the target handling time. Our simulation confirms that if the sequence, and only the sequence is to be followed, retrieval jobs tend to be handled earlier than the time specified in the schedules generated and therefore incur higher amount of total yard truck waiting time. This behavior could be corrected if the target handling time is taking into consideration. For this reason, we omit the results about pure retrieval and mixed job test cases.

For any yard crane schedules to be practical, the crane operator would want the make-spans provided by the schedules are reliable. Therefore, we also examine the reliability of the make-span generated by the schedules. Given a set of jobs, let us define the 'ideal make-span' as the make-span which enables the handling of the last job in the set just on its target handling time. The 'scheduled' make-span is therefore defined as the make-span that empowers the handling of the last job in the schedule just on its scheduled time, and the 'simulated' make-span is the make-span reported as the last job being handled in the simulation. If all the make-spans in the simulation runs can be covered by the scheduled make-span, then obviously the scheduled make-span is highly reliable. On the other side, if all the ideal make-spans can be covered by scheduled make-spans, and which are not very far away from the ideal ones, then the scheduled make-spans are tight.

Table 8 lists the scheduled and simulated make-spans compared to the ideal ones. Examining the "Scheduled" column in Table 8, it is found that for heavy workload cases the maximum percentages of scheduled results exceed the ideal ones by about 10% to 15%. This indicates that in heavy

workload yards, sometimes delay is inevitable. It can also be noticed that for light workload 40-slot cases, the maximum percentage is also high. This may be due to the fact that in 40-slot yards, the chance of crane interference will be higher than in 60-slot cases. The minimum percentage is somewhat equal to or close to 100%, which indicates the jobs can be finished within the ideal make-span for some cases. Average percentage of scheduled results indicates that the make-spans are very close to the ideal ones in light workload cases and slightly bigger in heavy workload cases. The simulated results show the same tendency as the scheduled ones with the maximum and minimum percentages bounded by the scheduled results. The average results indicate that the algorithm can generate schedules which enable the work to be handled with a slight delay in make-span for both light and heavy workloads, i.e., between 100.0% and 104.0%, but are still bounded by the scheduled results.

In all test cases, the make-span always lies in the range generated by the clustering-reassigning algorithm. This leads us to conclude that the clustering-reassigning approach provides a reasonable solution to the multiple yard crane scheduling problem and the results can be used to guide real-life operations. In addition, the algorithm's ability to providing instant solutions is one of the essential features for real time scheduling. Furthermore, the proposed algorithm might be very useful in assessing the workload for different yard blocks and determine the inter-block yard crane movements.

## 5. Conclusion

In this paper, we investigated the single yard block multiple yard crane scheduling problem with both storage and retrieval container moves present simultaneously. The yard crane interference as well as other operational constraints are realistically modeled and tackled using a novel clustering-reassigning approach. The clustering-reassigning approach provides solutions in polynomial time in terms of number of jobs to be scheduled and is therefore an ideal candidate for real time yard crane scheduling. The proposed algorithm provides fast and high quality results and the comparison with the continuous time mixed integer programming model supported by the latest optimization software package confirms the algorithm's performance. Simulation runs further validate the robustness and stability of the algorithm.

Future research directions for this particular problem may include consideration of the uncertain/variable handling times for container storage and retrieval operation. Schedules with the capability to deal with such variability, on top of the quick re-scheduling capability, would be more practical to guide real-world operations. This will also help address the issue of the conservative estimation of handling time, which we have discussed in Section 4, that leads to unnecessary early moves of containers.

## Acknowledgments

The authors would like to thank the anonymous reviewers and the journal editors for their most constructive comments and suggestions to improve the paper.

## References

Alphaliner. 2009. Cellular fleet at 1st July 2009. `www.alphaliner.com`.

Alphaliner. 2011. Cellular fleet at 1st August 2011. `www.alphaliner.com`.

Alphaliner. 2013. Cellular fleet at 1st June 2013. `www.alphaliner.com`.

Blazewicz, J., T. C. E. Cheng, M. Machowiak, C. Oguz. 2011. Berth and quay crane allocation: a moldable task scheduling model. *J. Oper. Res. Society* **62** 1189–1197.

Cao, J. X., D.-H. Lee, J. H. Chen, Q. Shi. 2010. The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods. *Transportation Res. E* **46**(3) 344–353.

Cao, Z., D.-H. Lee, Q. Meng. 2008. Deployment strategies of double-rail-mounted gantry crane systems for loading outbound containers in container terminals. *Inter. J. Prod. Economics* **115** 221–228.

Chung, R. K., C.-L. Li, W. Lin. 2002. Interblock crane deployment in container terminals. *Transportation Sci.* **36**(1) 79–93.

Gharehgozli, A. H., Y. Yu, R. de Koster, J. T. Udding. 2014. An exact method for scheduling a yard crane. *Eur. J. Oper. Res.* **235**(2) 431–447.

Guo, X., S. Y. Huang. 2012. Dynamic space and time partitioning for yard crane workload management in container terminals. *Transportation Sci.* **46**(1) 134–148.

Guo, X., S. Y. Huang, W. J. Hsu, M. Y. H. Low. 2011. Dynamic yard crane dispatching in container terminals with predicted vehicle arrival information. *Adv. Eng. Informatics* **25**(3) 472–484.

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

Kim, K. Y., K. H. Kim. 1997. A routing algorithm for a single transfer crane to load export containers onto a containership. *Comp. & Ind. Eng.* **33**(3–4) 673–676.

Kim, K. Y., K. H. Kim. 1999. A routing algorithm for a single straddle carrier to load export containers onto a containership. *Inter. J. Prod. Economics* **59** 425–433.

Lee, D.-H., Z. Cao, Q. Meng. 2007. Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm. *Inter. J. Prod. Economics* **107** 115–124.

Li, W.-K., M. Goh, Y. Wu, M. E. H. Petering, R. de Souza, Y. C. Wu. 2012. A continuous time model for multiple yard crane scheduling with last minute job arrivals. *Inter. J. Prod. Economics* **136**(2) 332–343.

Li, W.-K., Y. Wu, M. E. H. Petering, M. Goh, R. de Souza. 2009. Discrete time model and algorithms for container yard crane scheduling. *Eur. J. Oper. Res.* **198** 165–172.

Linn, R., J.-Y. Liu, Y.-W. Wan, C. Zhang, K. G. Murty. 2003. Rubber tired gantry crane deployment for container yard operation. *Comp. & Ind. Eng.* **45** 429–442.

Ng, W. C. 2005. Crane scheduling in container yards with inter-crane interference. *Eur. J. Oper. Res.* **164** 64–78.

Ng, W. C., K. L. Mak. 2005a. An effective heuristic for scheduling a yard crane to handle jobs with different ready times. *Eng. Optim.* **37**(8) 867–877.

Ng, W. C., K. L. Mak. 2005b. Yard crane scheduling in port container terminals. *Appl. Math. Moedelling* **29** 263–276.

Petering, M. E. H., Y. Wu, W.-K. Li, M. Goh, R. de Souza. 2009. Development and simulation analysis of real-time yard crane control systems for seaport container transshipment terminals. *OR Spectrum* **31**(4) 801–835.

Sipser, M. 1997. *Introduction to the Theory of Computation*. PWS Publishing Company, Boston.

Steenken, D., S. Voß, R. Stahlbock. 2004. Container terminal operation and operations research — a classification and literature review. *OR Spectrum* **26**(1) 3–49.

Tou, J. T., R. C. Gonzalez. 1974. *Pattern Recognition Principles*. Addison-Wesley, Reading.

Vis, I. F. A., H. J. Carlo. 2010. Sequencing two cooperating automated stacking cranes in a container terminal. *Transportation Sci.* **44**(2) 169–182.

Vis, I. F. A., R. de Koster. 2003. Transshipment of containers at a container terminal: An overview. *Eur. J. Oper. Res.* **147** 1–16.

Zhang, C., Y.-W. Wan, J. Liu, R. J. Linn. 2002. Dynamic crane deployment in container storage yards. *Transportation Res. B* **36** 537–555.