Yard Crane Scheduling to Minimize Total Weighted Vessel Loading Time in Container Terminals

Huang, Shell Ying; Li, Ya

2017

Huang, S. Y. & Li, Y. (2017). Yard crane scheduling to minimize total weighted vessel loading time in container terminals. *Flexible Services and Manufacturing Journal*, 29(3-4), 689–720.

<https://hdl.handle.net/10356/86084>

<https://doi.org/10.1007/s10696-017-9299-1>

© 2017 Springer Science+Business Media. This is the author created version of a work that has been peer reviewed and accepted for publication by *Flexible Services and Manufacturing Journal*, Springer Science+Business Media. It incorporates referee's comments but changes resulting from the publishing process, such as copyediting, structural formatting, may not be reflected in this document. The published version is available at: [<https://doi.org/10.1007/s10696-017-9299-1>].

Downloaded on 26 Apr 2023 18:33:22 SGT

Yard Crane Scheduling to Minimize Total Weighted Vessel Loading Time in Container Terminals

Shell Ying Huang and Ya Li

School of Computer Science and Engineering,
Nanyang Technological University,
Singapore 639798.

{assyhuang, LIYA}@ntu.edu.sg

+65 67904932

ABSTRACT

One of the most important objectives for container terminals is to minimize the total weighted turnaround time for vessels. Minimizing total weighted vessel loading time directly helps minimize total weighted vessel turnaround time. In this paper we consider the yard crane (YC) scheduling problem in a multi-vessel loading process. We propose a YC scheduling system that minimizes the total weighted maximum tardiness of loading jobs. Our experiments show that this is significantly more effective in the minimization of the total weighted vessel loading time than the existing YC scheduling methods that minimize job waiting time or makespan. Our scheduling system also has the ability to dynamically optimize the YC operations with the consideration of the maximum tardiness already incurred to each vessel earlier in the loading process. To evaluate against our YC scheduling system, an optimal algorithm to minimize makespan (useful if there is YCs shortage) and an optimal algorithm to minimize total weighted job tardiness (common in manufacturing) are also proposed.

Keywords: Container Terminal, Yard crane scheduling, Vessel turnaround time, Vessel loading time optimization

1. Introduction

One of the most important objectives for container terminals is to reduce vessel turnaround time (Steenken et al. 2004). For high-capacity terminals that serve vessels of different sizes from many shipping lines, minimizing the total weighted turnaround time for vessels is often the most important objective. This is because terminal operators may treat different shipping lines with different priorities, and because it is more important to shorten the turnaround time for big ocean-crossing vessels than for small regional feeders.

The turnaround time of a vessel is the vessel berthing time for unloading and loading of containers. Minimization of a vessel's turnaround time means minimization of the waiting time of the Quay Cranes (QCs) when they are loading/unloading containers for vessels. Yard Crane (YC) operations which are well synchronized with the QCs is a critical factor to achieve this. Figure 1 shows part of the quay and storage yard of a terminal where QCs, vehicles and YCs are handling containers. QCs unload containers from the vessel and load containers onto the vessel at the quayside. After the job lists of the QCs have been scheduled, the continuous working of the QCs will achieve the shortest vessel turnaround time. Vehicles transport containers between QCs and YCs. YCs transfer containers between vehicles and container blocks in the storage yard. In this pipeline of container handling operations, a QC's operation schedule decides when a YC should handle a container (Wu et al., 2015). Lee et al. (2009) argues that YC schedule should be synchronized with the QC schedule. This synchronization is to ensure that the QCs will have their waiting time minimized and work continuously.

The YC scheduling problem has been studied either as a standalone problem or as part of an integrated scheduling problem where other type(s) of equipment or storage allocation are also considered. As may be seen from the next section, current standalone YC scheduling solutions cannot guarantee optimal synchronization of YCs with the activities of the QC. Poor coordination between different types of equipment is the main reason for efficiency loss in container handling (Lau and Zhao, 2008). Integrated scheduling of multiple types of equipment has been studied by a number of people in order to produce well-coordinated operation plans. However, such scheduling systems are very complex and requires substantial amount of computational time to generate an operation plan. In Chen et al. (2013), it is

reported that about 30-40 minutes is required to compute a scheduling plan for two and half hours of operations for 4-5 QCs. It is not known whether the scheduling algorithms are able to scale up for high-capacity terminals.

Figure 1: Part of the quay and storage yard of a container terminal.

In most container terminals, a ship processes her unloading operations before loading operations and plans the loading operations or unloading operations, respectively (Zeng and Yang, 2009). Vessel loading operations are more difficult to synchronize because the sequence of the containers to be loaded by a QC has to be respected. YC delays in the loading process and in the unloading process have different impacts on QCs and vehicles (Li et al., 2012, Wu et al. 2015). So some studies focus on the vessel loading process only, for examples, Meersmans and Wagelmans (2001) and Zeng and Yang (2009). For most studies that include both vessel loading and unloading, for examples, Chen et al. (2007) and Lau and Zhao (2008), it is also assumed that vessel unloading precedes vessel loading and each YC handles either vessel loading or vessel unloading but not both. We study the multi-YC scheduling problem for the multi-vessel loading process in this paper. In Section 5, we explain how to include unloading operations in YC scheduling.

We propose a standalone multiple-YC scheduling system with an objective of minimizing the total weighted tardiness to the loading processes of a number of vessels. By tardiness to the loading process of a vessel, we mean the actual length of the loading process minus the scheduled length of the loading process as decided by the QC schedules of the vessel. This objective directly helps reduce the total weighted vessel turnaround time for a terminal. Our system has the simplicity of a standalone YC scheduling system but directly and explicitly synchronizes YC operations with the QCs'. To the best of our knowledge, this is the first paper that performs standalone YC scheduling to minimize delays to the completion of vessel loading which is not the same as to minimize the total job tardiness. It is also the first paper to perform YC scheduling in order to reduce the total weighted vessel loading time in a multi-vessel loading process of a container terminal. The challenge of YC scheduling in a multi-vessel loading process is because each YC may be loading for more than one vessel, the scheduling decision of one YC affects the other YCs' loading for the same vessel. These other YCs will affect other more YCs. We will explain this later.

Other contributions of the paper are the proposed complete and optimal algorithms for scheduling a single YC to minimize makespan or to minimize total weighted tardiness of jobs. These two algorithms are developed mainly to compare their performance with the multi-YC scheduling system to minimize the total weighted tardiness to the loading processes of vessels. However, they would be useful in some situations. For example, when the number of YCs in the terminal is low compared with other types of equipment like vehicles or QCs, then scheduling YCs to minimize makespan may offer opportunities for faster YC re-

deployment to other blocks (Huang et al. 2011). The existing single YC scheduling algorithms compute job sequences to minimize vehicle waiting time, not makespan.

The rest of the paper is structured as follows. Firstly, we review the related studies in Section 2. The YC scheduling algorithms and the YC scheduling system that will plan all jobs for all YCs for the multi-vessel loading process are presented in Section 3. The experimental evaluations are presented in Section 4. Section 5 discusses how to extend the YC scheduling system to YCs handling unloading operations or handling a mixture of loading and unloading operations. Conclusions are drawn in the last section.

2. Related Work

Yard crane scheduling may be studied alone where container jobs are sequenced based on job information and a certain objective function. Yard crane scheduling may also be studied in an integrated scheduling problem involving YCs and other types of equipment or together with storage allocation. The YC scheduling problems studied can be different because of the different layouts of the yard blocks (side/end loading, in other words, Asian/European layouts), the number and types of YCs deployed in a block, the number of yard blocks considered. Carlo et al. (2014) has a recent, comprehensive review of the studies on storage yard operations.

2.1 Scheduling YCs Alone

When scheduling YCs alone with side loading blocks, some works considered a single YC handling containers in a block for *vessel loading* only. Kim and Kim (1999) studied the YC scheduling problem where the load plan for the QC and the bay plan at the yard block for the loading containers are given. A Mixed Integer Programming (MIP) model is proposed to minimize the total gantry movement time of the YC. The solutions focus on the sequence of bay visits and the number of containers to be picked up at each bay while the individual container pick-up sequence within a specific bay is left to the crane operator. Later, Kim and Kim (2003) and Kim et al. (2004) extended the study of this problem by comparing exact optimization, a beam search heuristic and a Genetic Algorithm (GA). Narasimhan and Palekar (2002) proved that the yard crane scheduling problem is NP-complete and proposed exact and heuristic algorithms after formulating it as an integer program.

For a single YC supporting both *loading and unloading* of vessels in a side loading yard block, Ng and Mak (2005a, 2005b) developed branch-and-bound heuristics to schedule a given set of jobs with different ready times. Their objective is to minimize the total vehicle waiting time. Kumar and Omkar (2008) used particle swarm optimization with genetic algorithm operators to schedule jobs with different ready times to minimize total vehicle waiting time. Guo et al. (2011) applied A* search to compute optimal job sequence based on job ready times to minimize total vehicle waiting time.

Some works considered 2 or more YCs handling containers for *vessel loading* in side loading yard blocks. Jung and Kim (2006) considered 2 YCs working in one yard block to support vessels loadings with a GA and a Simulated Annealing (SA) algorithm to minimize makespan. Lee et al. (2007) considered 2 YCs working in 2 non-overlapping zones to serve vessel loading operations of a QC with a SA algorithm to minimize makespan. Chen and Langevin (2011) studied the problem of scheduling multiple YCs where a YC may move among blocks in the same row and up to 2 YCs may work in a block. Vehicle ready time is not considered. A genetic algorithm and a tabu search algorithm are used to minimize the makespan of the YCs.

Ng (2005) studied the problem of scheduling multiple YCs to handle both *vessel loading and unloading* jobs with different ready times within a yard zone with MIP and heuristics.

Froyland et al. (2008) presented an integer programming approach to schedule multiple gantry cranes for loading and unloading trains in a container terminal. Li et al. (2009) presented a discrete MILP model to schedule multiple YCs to minimize a linear combination of loading job earliness and tardiness and unloading job tardiness. Li et al. later (2012) presented a continuous time MILP model for the same problem. Wu et al., (2015) proposed a cluster-reassigning algorithm also for the same problem. Guo and Huang (2012) proposed space and time partitioning methods to manage the workload among multiple YCs working in a row of yard blocks to minimize total vehicle waiting time. Cao et al. (2008) considered Double-Rail-Mounted gantry (DRMG) crane systems where two YCs can pass through each other along a row of blocks. They used a combined greedy and SA algorithm to minimize makespan. Huang et al. (2015) proposed a job dispatching algorithm for Twin-yard-cranes with side loading based on the non-zero-sum game approach.

For end loading blocks, there are a number of studies published. Since end loading yard layout is not the focus of this paper, we will not go into details of these studies. Readers may refer to Saanen and van Valkengoed (2005), Dorndorf and Schneider (2010), Park et al. (2010), Stahlbock and Voss (2010), Vis and Carlo (2010), Gharehgozli et al. (2014), Gharehgozli et al. (2015), Briskorn and Angeloudis (2016), Hu et al. (2016), Briskorn et al. (2016), Kim et al. (2016), Ahmt et al. (2016), Speer and Fischer (2017). Many of these works compute a container storage and retrieval sequence for YCs but Kim et al. (2016) and Ahmt et al. (2016) proposed solutions for a different objective. They try to minimize the number of container moves due to rehandling containers when a YC retrieves containers according to a given sequence.

As can be seen from above, standalone YC scheduling usually tries to either minimize total (average) vehicle waiting time or minimize makespan for the YCs. Minimization of the total vehicle waiting time at yard blocks may result in some vehicles returning to the quayside earlier than they are needed while others are late for the QCs. The late vehicles incur QC waiting time. Minimizing makespan for a YC optimizes the YC productivity, but optimal makespan of a YC may also cause long delays to some vehicles. Therefore, scheduling YCs alone with the objectives to minimizing vehicle waiting time or YC makespan may not produce a YC operation plan well-synchronized with the QCs. Huang et al. (2014) presented an algorithm MMT-RBA* for YC scheduling to minimize the maximum job tardiness of a vessel in order to minimize the vessel's turnaround time. MMT-RBA* uses the Recursive Backtracking algorithm with a bound as in A* search to Minimize Maximum Tardiness of jobs. They concluded that minimizing maximum tardiness is more effective in minimizing one vessel's turnaround time than minimizing makespan or minimizing vehicle waiting time. However, the relationship between the maximum tardiness of both loading and *unloading* jobs of a vessel caused by YCs and the vessel's turnaround time is not well established.

2.2 Integrated Scheduling

Integrated scheduling of container handling operations may cover all three main types of equipment, namely, QCs, vehicles and YCs. It may also cover just two types of equipment, for example, QCs and vehicles, or vehicles and YCs. Or it may perform vehicle scheduling and storage allocation together. Since our focus is on YC scheduling, we only review works on integrated scheduling that include YCs.

A number of studies consider vessel loading operations only. Meersmans and Wagelmans (2001) presented a Branch-and-Bound algorithm and a Beam Search heuristic to schedule QCs, vehicles and YCs in an end loading yard layout for automated terminals. The objective is to minimize the makespan of a set of container loading operations. Zeng and Yang (2009) proposed a simulation and optimization method to schedule loading operations for QCs,

vehicles and YCs. Simple dispatching rules are used to generate initial solutions. GA is used to improve solution qualities. A Neural Network model is used to filter out bad solutions to reduce the simulation time in evaluating solution qualities. Cao et al. (2010) presented an integrated scheduling model for YCs and vehicles. They proposed a general Benders cut-based method and a combinatorial Benders cut-based method to minimize the makespan of retrieving a set of containers from storage yard to the QCs. However, containers are delivered to the QCs in any order decided by the scheduling method.

Other integrated scheduling works consider both loading and unloading operations. Chen et al. (2007) proposed a Tabu search algorithm to find scheduling solutions for QCs, vehicles and YCs to minimize the makespan for the loading *or* unloading of a set of ships. For one ship, the scheduling of unloading operations and that of loading operations are considered two independent problems. Each YC handles either loading containers or unloading containers but not both. Lau and Zhao (2008) defined a mixed integer programming model for the integrated scheduling of QCs, vehicles and YCs in an automated container terminal with end loading yard blocks. A multi-layer genetic algorithm was proposed to find optimal or near-optimal solution to minimize delay to QC operations, the total travelling time of vehicles and total gantry time of YCs. Chen et al. (2013) formulated the QC, vehicle and YC scheduling problem as a constraint programming model. A three-stage iterative algorithm is proposed to minimize the makespan of a set of operations in the vessel loading and unloading processes.

There is no doubt that integrated scheduling is able to generate better-synchronized scheduling plans for efficient terminal operations, not just for an efficient subprocess. However, in Zeng and Yang (2009), it takes more than 2300 seconds to schedule 400 containers on 4 QCs, 10 YCs and 16 vehicles. In Chen et al. (2013), it takes more than 2600 seconds to schedule 400 containers on 5 QCs, 20 YCs and 30 vehicles. Note that four to six QCs are usually allocated to serve one medium to large sized vessel. In our largest experiment, we tested 48 YCs loading 9792 containers for 32 QCs which serve 8 vessels.

2.3 General Machine Scheduling Problems

The YC scheduling problem is one type of machine scheduling problems. YC gantry times between two jobs in a job sequence are the machine setup times which are sequence dependent. Vehicle arrival times at yard blocks are the job ready times. YC loading/unloading times are the job processing times. The objective is usually to minimize total job completion time (which is equivalent to total job waiting time), or to minimize total tardiness, or to minimize maximum lateness. To the best of our knowledge, there is no report on any problem which is to minimize maximum tardiness, as we do in this paper.

Lenstra et al. (1977), Graham et al. (1979) and Pinedo (2016) discussed the complexity of a number of machine scheduling problems. Allahverdi et al. (2008) presented a survey on scheduling problems with setup times. Apart from mathematical programming approaches, the other approaches used to solve these problems are heuristic, meta-heuristic or branch-and-bound search.

3. Solving the YC Scheduling Problem for Multi-vessel Loading

3.1. The Problem

We consider a container terminal with an Asian yard layout, i.e. side-loading yard blocks. There is space for vehicles to move at the side of a yard block such that vehicles can be sequenced according to the YC's planned working sequence regardless of the vehicles' arrival order. It is assumed that there are enough vehicles to satisfy the transport demand and vehicle traffic in the terminal is smooth.

All the YCs in the storage yard are identical. Each YC works in a yard zone which is part of a yard block and does not overlap with the yard zone of another YC. Thus a yard block can be one yard zone with one YC or two non-overlapping zones with 2 YCs. The partition of the storage yard into yard zones and YC deployment to the non-overlapping yard zones can be done by a separate algorithm but is out of the scope of this paper. This separate algorithm can be executed in between the executions of the YC scheduling algorithm presented in this paper. In other words, the YC scheduling algorithm plans the job sequence for each YC and each YC performs the operations accordingly. When the need to re-partition the yard and redeploy (some or all) YCs, decisions will be made by a separate algorithm. After this, the YC scheduling algorithm can continue the job sequence planning for each YC in its (new) zone. This means that the operation of a YC does not block or hinder that of another YC. In the rest of the paper, we use the terms a yard zone and a yard block interchangeably.

Just as in Chen et al. (2007), the scheduling of loading and unloading operations of a vessel is considered separate problems and we consider the simultaneous loading operations of a number of vessels. Each vessel has a number of QCs allocated to it. We assume that the distribution of the workloads among these QCs is even. This is justified by the objective of QC scheduling which is to minimize vessel turnaround time. This objective is achieved by an even workload distribution among the QCs. Therefore all QCs serving a vessel are scheduled to complete loading at the same time.

Each QC works for one vessel and loads containers fetched by vehicles from several YCs. Also as in Chen et al. (2007) the YCs in our problem do not mix loading and unloading operations. Each YC serves vehicles which take containers to different QCs, possibly working for different vessels. So each QC may be served indirectly (through vehicles) by several YCs and each YC may serve indirectly a different set of QCs/vessels, i.e. a many-to-many relationship between YCs and QCs. For example, YC1 in Figure 1 may be loading containers for QC1 and QC3 which serve vessel 1 and vessel 2 respectively. YC1 and YC2 may be both loading containers for QC1. At the same time, YC1 and YC3 may be both loading containers for QC3.

Each vessel has a weight which represents the priority a terminal operator assigns to the vessel's operations.

Each loading container for YCs has a deadline which is based on the QC schedule. As in Lee and Chen (2010) travelling time of a QC between two ship bays is ignored. So during a vessel loading process, if container i is to be loaded after container j by a QC, the scheduled arrival time of vehicle/container i at its QC is the scheduled arrival time of vehicle/container j plus QC's time to load j . This is also the method used in Lau and Zhao (2008). The time a vehicle should leave the yard block with the container is the time the vehicle should arrive at a QC minus the expected travel time from the yard block to the QC (we assume no traffic congestions). This is the deadline of the vehicle job for the YC. It should be noted that the containers a YC handles may come from vehicles serving different QCs possibly for different vessels. Therefore the deadlines for these jobs are independent from each other.

Each vehicle has a predicted arrival time at its yard block. Based on the time a container is needed by a QC, a suitable vehicle will be allocated by the vehicle dispatching system to retrieve it from its yard block. The arrival time of the vehicle at the yard block depends on when and where the allocated vehicle finishes its previous job. The vehicle arrival time at the yard block can be predicted from the time the vehicle is expected to finish its previous job and move towards the yard block and the expected travel time of the vehicle. We generate a random value which is correlated with the job's deadline to model the job/vehicle arrival time at a yard block.

Each loading job has a slot location (yard bay) of the container which is the location of the stack the container will be retrieved by a YC as shown in Figure 2. The jobs that come to a

yard block will have randomly generated slot numbers. Other recent studies using randomized container locations include, for example, Zeng and Yang (2009). The container is assumed to be on the top of the stack specified by the slot number. So reshuffling time is not considered. This means the time taken for a YC to move a container from a container stack to a vehicle can be approximated by a constant. This assumption is the same as in Ng (2005) and Cao et al. (2008). The no-reshuffling assumption represents the scenario where the containers are the ones that move between vessels and storage yard. Because vessel arrival times can be better predicted than the arrival times of external trucks, the storage and retrieval times of containers can be better predicted than that of import/export containers carried by external trucks. Due to this, the container allocation system, together with block relocation procedures and pre-marshalling, is able to prepare the containers before vessel loading operations. As a result, most of the times containers are on top of the stack with no reshuffling need when their storage/retrieval operation comes. For example, in Yu and Qi (2013), overnight remarshalling is used to prepare storage stacks for subsequent vessel operations. YC gantry movement time between two slots is calculated based on a constant gantry speed and the distance between the two slots.

Figure 2: Top view of a yard zone.

Based on the above, the following notations are used by the YC scheduling problem. Because each container has one and only one designated yard block, we will not explicitly indicate the yard block of the container in the notations.

- a_{ih} the arrival time of loading job(vehicle) i for vessel h at the yard zone of a YC.
- p the time taken by a YC to load a container onto the vehicle which is a constant.
- d_{ih} the deadline of job i for vessel h at a YC.
- v_h the weight of vessel h .
- m_{ij} the time taken for a YC to move from the slot position of job i to that of job j . m_{0j} denotes the YC gantry time from its position at the start of the job sequence to the position of job j .
- N the number of loading vessels

The YC scheduling system which manages the multiple YCs involved in the multi-vessel loading process is to minimize the total weighted vessel loading time, that is,

$$\min \sum_{h=1}^N v_h L_h \quad (1)$$

where v_h is the weight of vessel h and L_h is the total time taken to load vessel h .

Given the QC schedules of the loading vessels, we show that the above objective is equivalent to the minimization of total weighted maximum tardiness of vessel loading jobs:

$$\min \sum_{h=1}^N v_h (\max_i (T_{ih})). \quad (2)$$

where T_{ih} is job i 's tardiness with respect to the job's deadline.

Our objective is to minimize total weighted vessel loading time given in (1). A vessel's (actual) loading time is the loading time required as specified by the QC schedules plus the

delays to the completion of the QC schedules. First, consider one QC of one vessel h . Suppose a QC has n containers numbered 1 to n in its loading sequence which is computed by the QC scheduling system of the terminal.

Lemma: Consider a sequence of n vehicles scheduled to come to one QC to deliver containers. Each vehicle has a scheduled time to reach its QC in order to synchronize with the QC schedule. Let $\{T_{1h}, T_{2h}, \dots, T_{nh}\}$ be the tardiness of the vehicles in reaching the QC (some tardiness is zero when a vehicle is not late). The delay to the QC in completing this sequence of jobs is $\max(T_{1h}, T_{2h}, \dots, T_{nh})$.

Proof by induction:

If there is only one job ($n = 1$), the delay to the QC in completing the sequence is T_{1h} . Assume that the delay to the QC in completing a sequence of k jobs is $\max(T_{1h}, T_{2h}, \dots, T_{kh})$ and the next job has a delay of $T_{k+1,h}$. If $T_{k+1,h} \leq \max(T_{1h}, T_{2h}, \dots, T_{kh})$, the vehicle arrives at the QC before (or at the time) the QC completes the k^{th} loading operation. There is no delay in the $(k+1)^{th}$ job. So the delay of this QC after the $(k+1)^{th}$ loading operation is $\max(T_{1h}, T_{2h}, \dots, T_{kh})$ which is $\max(T_{1h}, T_{2h}, \dots, T_{k+1,h})$. If $T_{k+1,h} > \max(T_{1h}, T_{2h}, \dots, T_{kh})$, the QC will need to wait for the vehicle for a time period of $T_{k+1,h} - \max(T_{1h}, T_{2h}, \dots, T_{kh})$. The delay of this QC after the $(k+1)^{th}$ loading operation will become $T_{k+1,h}$. So the delay of this QC after the $(k+1)^{th}$ loading operation is also $\max(T_{1h}, T_{2h}, \dots, T_{k+1,h})$. Therefore the delay to the QC in completing its sequence of n jobs is $\max(T_{1h}, T_{2h}, \dots, T_{nh})$.

When a number of QCs are allocated to load one vessel, the QCs are scheduled to complete their loading operations at the same time. It follows that the longest delay among the QCs will be the delay to the completion of the vessel loading. It also follows that the longest delay among all container jobs of a vessel is the delay to the completion of the loading of the vessel.

Corollary 1: The delay to the completion of the loading of a vessel will be $\max(T_{ih})$ where i is a container to be loaded onto vessel h .

Since there are enough vehicles to satisfy the transport demand and vehicle traffic in the terminal is smooth, the tardiness of vehicles T_{ih} will be due to the tardiness of jobs at their yard blocks caused by the YCs. Therefore, we can consider T_{ih} as the job tardiness at YCs and the objective of YC scheduling for multi-vessel loading as expressed by (1) is equivalent to that expressed by (2).

In (2) the loading containers of each vessel h will come from a set of YCs. All these YCs may affect the final value of $\max(T_{ih})$. However, a YC may be loading for different vessels at the same time so affecting the final values of $\max(T_{ih})$ for different h at the same time. Each of the YCs plays a part in producing the best value for (2). However it is not possible to plan all the YC operations in all the yard blocks for all the loading vessels at once. The loading process of each vessel typically takes a number of hours to complete. Due to the dynamic nature of terminal operations, it is not possible to know all the vehicle arrival times at the yard blocks at the beginning of a vessel loading process. So planning has to be done in stages along the time line. We divide the global optimization problem into a number of sub-problems.

3.2. Dividing the YC Scheduling Problem for Multi-vessel Loading into Sub-problems

The division of the global problem is first carried out along the time dimension. Dividing the time line into fixed planning windows either in terms of the length of time or the number of jobs is a simple solution. However, there is a drawback. For any two jobs in the same yard block, the shorter the time interval there is between their arrival times/deadlines, the more likely they will have impact on each other's position in a YC planning sequence. If planning is done in fixed windows, the impact of the planning results between two jobs in different windows will be ignored. This may produce sub-optimal solutions. Therefore, we

perform the planning of YC operations by rolling windows as time progresses and information about the container jobs becomes available.

Within one planning window, a number of yard blocks/YCs will supply containers to the loading vessels. The sets of yard blocks for different vessels are likely to overlap. Global planning of all the blocks is needed to achieve global optimum even within one planning window. However as the existing works on integrated scheduling have shown (Section 2.2), it is computationally very expensive to plan for all blocks together unless the number of blocks in a terminal is small. Given that the movement of one YC do not block or hinder that of another YC, we propose to plan YC schedules block by block in each planning window.

The loading jobs of a vessel h generally come from a number of yard blocks and are planned in a number of planning windows. Let t be the number of planning windows and b the number of blocks. The jobs in this vessel will be involved in $t \times b$ invocations of the YC scheduling algorithm, each invocation for one unique combination of window and block. We will number the invocation of the YC scheduling algorithm by $j = 1, 2, \dots$, each for a particular combination of window and block.

Corollary 2: The delay to the vessel h at the end of the j^{th} invocation of the scheduling algorithm, $D_{h,j}$, $j = 1, \dots, t \times b$, will be $\max(D_{h,(j-1)}, \max_i(T_{ih}))$ with $D_{h,0} = 0$, i is a container of vessel h to be loaded in the current planning window from the current yard block.

Corollary 2 follows from Corollary 1. In general, for each vessel h ,

$$D_{h,j} = \begin{cases} D_{h,(j-1)}, & \text{if the current block does not have containers for vessel } h, \\ \max(D_{h,(j-1)}, \max_i(T_{ih})), & \text{otherwise} \end{cases} \quad (3)$$

with $D_{h,0} = 0$. The objective of the j^{th} scheduling is

$$\min \left(\sum_{h=1}^N v_h D_{h,j} \right) \quad (4)$$

$D_{h,j}$ is affected by $D_{h,(j-1)}$, which is affected by $D_{h,(j-2)}$ and so on. However it is not computationally feasible to try the $y!$ possible planning orders for y yard blocks of a large container terminal to get the best result for the current planning window. We investigate three ways of arranging the order of the yard blocks for the planning tasks in the same window.

- a) **Weighted EDD order (WEDD).** This is an intuitive greedy approach. It means to first sort the yard blocks by the sum of the weights of vessels which have container operations in the respective blocks. Break tie by using earliest deadline of the operations in the block; This is to give priority to blocks that are handling containers for higher weighted vessels or needed by YCs earlier.
- b) **Earliest completion time order (ECT).** It is also an intuitive greedy approach. Sort the blocks in increasing order of the completion times of last operations of the blocks in the previous planning window. This is a very practical method as the scheduling algorithm is applied when YCs are completing their current scheduling plans and expect to receive their next plans.
- c) **Random order (RAN).** This is to test how sensitive the planning results are to the block planning order.

3.3. YC Scheduling to Minimize Total Weighted Maximum Tardiness (MTWMT)

The YC scheduling algorithm is to sequence the n vehicle jobs in a planning window for a YC/yard block. Each vehicle job is associated with a loading vessel. Each vessel has a weight v_h ($h = 1, 2, \dots, N$). Before the execution of the algorithm for the j^{th} time which is for a particular planning window and a particular YC block, the maximum tardiness from the operations already planned up to the $(j-1)^{th}$ time are known and given as $D_{h,(j-1)}$ for each vessel h . The jobs to be planned have their predicted job arrival times a_{ih} ($i = 1, 2, \dots, n$) and deadlines d_{ih} ($i = 1, 2, \dots, n$). The YC gantry times m_{ik} ($i = 0, 2, \dots, n$; $k = 1, 2, \dots, n$) are the

expected YC movement time from job i 's location to job k 's. The problem can be formulated as an integer program.

Parameters:

All those defined in Section 3.1 and

- n the number of jobs to be planned.
- $D_{h,(j-1)}$ the maximum tardiness of vessel h resulted from the $(j-1)^{th}$ scheduling. D_{h0} is set to 0.
- M a very large positive integer.
- N_j The number of loading vessels served in the j^{th} scheduling by a YC

Decision variables:

- C_{ih} the completion time of job i of vessel h by a YC, $i = 1, 2, \dots, n$. Each job is associated with a particular vessel h .

$$X_{ih,kh'} = \begin{cases} 1 & \text{if job } i \text{ of vessel } h \text{ is handled before job } k \text{ of vessel } h' \\ 0 & \text{otherwise} \end{cases}$$

$$D_{hj} = \max(D_{h,(j-1)}, \max_i(\max(0, C_{ih} - d_{ih})))$$

the maximum tardiness of vessel h
Note that for each j^{th} scheduling, D_{hj} is a decision variable while $D_{h,(j-1)}$ is a constant resulted from the $(j-1)^{th}$ scheduling

Objective function for the j^{th} scheduling:

$$\min \left(\sum_{h=1}^{N_j} v_h D_{hj} \right) \quad (5)$$

Subject to

$$C_{ih} \geq a_{ih} + p \quad (i = 1, \dots, n, h = \text{the vessel that } i \text{ is associated with}) \quad (6)$$

$$C_{kh'} - C_{ih} \geq m_{ik} + p - (1 - X_{ih,kh'})M \quad (i, k = 1, \dots, n, i \neq k, \\ h, h' = \text{vessels that } i, k \text{ are associated with respectively}) \quad (7)$$

$$X_{ih,kh'} + X_{kh',ih} = 1 \quad (i, k = 1, \dots, n, i \neq k, \\ h, h' = \text{vessels that } i, k \text{ are associated with respectively}) \quad (8)$$

$$X_{ih,kh'} \in \{0, 1\} \quad (i, k = 1, \dots, n, i \neq k, \\ h, h' = \text{vessels that } i, k \text{ are associated with respectively}) \quad (9)$$

$$D_{hj} \geq D_{h,(j-1)} \quad (h = 1, \dots, \text{no. of loading vessels}) \quad (10)$$

$$D_{hj} \geq C_{ih} - d_{ih} \quad (i = 1, \dots, n, \\ h = 1, \dots, \text{no. of loading vessels}) \quad (11)$$

Objective function (5) is basically the same as (4), with the exclusion of the vessels not affected by the j^{th} scheduling. Constraint (6) means the job completion time is greater than or equal to the job arrival time plus the job processing time. Constraint (7) gives the relationship between a job's completion time and that of its successors. Constraint (8) ensures the correctness of precedence between any two jobs. Constraint (9) specifies that $X_{ih,kh'}$ is either 1 or 0. Constraint (10) ensures that the maximum tardiness of each vessel from the j^{th} scheduling is greater than or equal to that from the $(j-1)^{th}$ scheduling. Constraint (11) ensures that the maximum tardiness of each vessel from the j^{th} scheduling is not less than the tardiness that any job may bring by the planned sequence.

Note that $D_{hj} = \max(D_{h,(j-1)}, \max_i(\max(0, C_{ih} - d_{ih}))) = \max_i(\max(0, C_{ih} - d_{ih} - D_{h,(j-1)})) + D_{h,(j-1)}$. Note also that $d_{ih} + D_{h,(j-1)}$ is the deadline for container i which, if satisfied, will not further delay vessel h . As the YC scheduling system plans the operations window by window

and block by block, this model has the ability to dynamically optimize the YC operations with the consideration of the maximum tardiness already incurred to each vessel h in the form of $D_{h,(j-1)}$. The scheduling algorithm will balance the use of YC resources to minimize further increases to the current maximum tardiness of the vessels to achieve the optimal value for the objective function (4).

Using the conventional three-field notation, this scheduling problem is $1 | p_i = p, ST_{sd}, r_i | \sum w_h T_{h,max}$. It is a single machine job scheduling problem with constant processing time, sequence dependent setup times and job ready times to minimize the total weighted maximum tardiness for families of jobs. The partition of jobs into families is not related to setup times as in the usual scheduling problems. It is a partition of the set of jobs into subsets based on the origin of the jobs (which vessel a container is for). As far as we know, this is the first study on this type of problems.

For such a scheduling problem, there are $n!$ possible solutions in total. The scheduling problem can be transformed into a tree-search problem. The root of the tree is the start node before the first job is selected. Each path from the start node to a leaf node in the tree represents a complete sequence of n jobs. The algorithm conducts a Backtracking tree search with a bound so that an optimal solution can be found with little computational time. For example, $n = 4$. The backtracking search algorithm quickly finds the first solution by visiting only 4 nodes in the depth-first manner in Figure 3 (the leftmost path). It would then backtrack to examine other solutions. If a better solution is encountered during the backtracking-then-search process, the knowledge of current best solution will be updated accordingly. Therefore, the search is complete and optimal. To avoid traversing the entire tree to find the optimal solution, we employ a bound to prune a no-good tree branch as early as possible, similar to the A* search. The pruned tree branches are shown by the dotted ovals in Figure 3. With our admissible bound which never overestimates, the backtracking search cuts away branches in the search tree without missing an optimal solution.

Figure 3: Example search for the optimal sequence for 4 jobs.

The pseudocode for the method is given in Figure 4. The algorithm MTWMT (short for Minimize Total Weighted Maximum Tardiness) will take a list J of jobs and a list $MaxT$ as input. The list $MaxT$ stores the current maximum tardiness to vessels, i.e. $D_{h,(j-1)}$ for all loading vessels at the start of the algorithm. The scheduling plan is computed by the recursive method B. After an optimal scheduling plan is obtained, $MaxT$ is updated to hold $D_{h,j}$ for the set of vessels which will be used in the next scheduling. Since each job is for one vessel only, we use C_i , a_i and d_i to represent C_{ih} , a_{ih} and d_{ih} respectively.

In method B of Figure 4, each job J_i in list J , will be considered as a potential next job for the partially planned sequence plannedJ and evaluated. The evaluation examines the tardiness of J_i (line 6 and line 7) and what is the resulting tardiness for J_i 's vessel (line 11). It also evaluates the lower bound of the job tardiness of the remaining jobs (line 13). The lower

bound tardiness LBT_j is the minimum for each job that is not planned yet. It is the job tardiness if it were served after job J_i . It follows that $MaxT2_h$ as computed by line 14 is the lower bound of the maximum tardiness for vessel h among the jobs not planned yet. $WeightedT$ as computed by line 16 and line 17 is the lower bound of the total weighted maximum tardiness of all vessels for all job sequences where the planned partial sequence is the prefix. If it shows that adding J_i to plannedJ is a no-good sequence because the lower bound of the resulting complete job sequences will not be better than the current best sequence, this search path is terminated (line 18). The method will then backtrack to consider another job. Otherwise J_i is removed from J and appended to plannedJ, followed by a call to B to plan for the remaining jobs ($J - \{J_i\}$) in line 22. To accelerate the search process, instead of choosing the next job randomly, J_i is selected in ascending order of job deadlines in line 5 because of the sorting in line 2. This increases the likelihood of discovering a good job sequence at the beginning stages of the search process and a better threshold to prune inferior branches later, significantly increasing the efficiency of the search for an optimal solution.

```
// Names in italic are input parameters to the algorithm
J = set of jobs to be scheduled for a YC in a planning window
MaxT = {MT1, MT2, ..., MTk}    // This stores the current maximum tardiness of k vessels
MTWMT (J, MaxT) {
1  plannedJ =  $\phi$ ; optimalJ =  $\phi$ ; CurSmallest =  $\infty$ ;
2  sort J into an ascending order of job deadlines;
3  B(J, plannedJ, MaxT);
4  MaxT = optimalMaxT;    //MaxT to be passed to the next YC's scheduling
}

B(J, plannedJ, MaxT) {
5  FOR each job  $J_i$  in J // Select  $J_i$  as the job to serve after jobs in plannedJ;
6     $C_i = \max(C_i + m_{ij}, a_i) + p$ ;    // job  $J_i$  is the last job in plannedJ
7     $T_i = \max(0, C_i - d_i)$ ;
8    FOR each vessel h
9       $MaxT1_h = MaxT_h$ ;
10      $MaxT2_h = 0$ ;
11      $MaxT1_{h'} = \max(MaxT1_{h'}, T_i)$  //  $J_i$  belongs to vessel  $h'$ ;
12     FOR each job  $J_j$  in J,  $i \neq j$     // for each job not planned yet
13        $LBT_j = \max(0, \max(C_i + m_{ij}, a_j) + p - d_j)$ ;    //  $LBT_j$  is the LB of job  $J_j$ 's tardiness
14        $MaxT2_h = \max(MaxT2_h, LBT_j)$  if  $J_j$  belongs to vessel h;
15     WeightedT = 0;
16     FOR each vessel h
17       WeightedT = WeightedT +  $v_h * \max(MaxT1_h, MaxT2_h)$ ;    //  $v_h$  is the weight of vessel h
18     IF WeightedT < CurSmallest    // when the condition is false, this search path ends
19       Remove  $J_i$  from J;
20       Append  $J_i$  to plannedJ;
21       IF J is not empty
22         B(J, plannedJ, MaxT1);
23       ELSE
24         CurSmallest = WeightedT;
25         optimalJ = plannedJ;    //Store the current best list
26         optimalMaxT = MaxT1;
}
```

Figure 4: Pseudocode of the MTWMT algorithm

Because the lower bound LBT_j and therefore WeightedT do not overestimate, no optimal solution will be pruned in our backtracking search. This will guarantee that MTWMT will

find a job sequence that minimizes the total weighted maximum tardiness of vessels among all sequences of the n jobs.

Note that when the YC scheduling system tries to find a solution that minimizes (4) block by block and planning window by planning window, it will be able to minimize the maximum tardiness of the jobs from a vessel of less weight without increasing the total weighted maximum tardiness of the jobs from vessels of higher weights. For example, there are two vessels with weights v_1 and v_2 where $v_1 > v_2$. If there are two planning sequences with the same tardiness for vessel 1 and different tardiness for vessel 2, the scheduling algorithm will choose the sequence which incurs less tardiness for vessel 2. Our experimental results in Section 4 confirm this.

3.4. The YC Scheduling Algorithms for Two Other Objective Functions

To evaluate the effectiveness of minimizing total weighted maximum tardiness for minimizing total weighted vessel loading time, we design two algorithms with different objectives to compare against the MTWMT algorithm. We present the MMS algorithm to Minimize MakeSpan in Section 3.4.1 and the MTWT algorithm to Minimize Total Weighted Tardiness for jobs in Section 3.4.2. Both these two algorithms use the same backtracking approach to search for an optimal solution for its objective function as the MTWMT algorithm. This means both algorithms are complete and optimal if there is no time constraint. Both algorithms use a suitable bound to improve search efficiency.

3.4.1. Minimize MakeSpan (MMS)

To minimize makespan is a common objective in YC scheduling. If optimizing YC productivity is the main concern of a container terminal, minimizing YC makespan will be able to achieve the optimal productivity of individual yard cranes. The objective function is to minimize the completion time of the last YC operation.

The input to this algorithm is the same as those for MTWMT except that the algorithm will not consider the deadlines of the jobs and the vessel weights. The algorithm in Figure 5 conducts a backtracking search of the tree is to find a path from the start node to a leaf node which minimizes the completion time of the last job in the scheduled job sequence.

In method B of Figure 5, each job J_i in list J will be considered as a potential next job for the partially planned sequence plannedJ and evaluated. The evaluation first computes the completion time of J_i (line 5). If job J_i is the only job in J which means it is the last job, the current best solution is updated when necessary (line 8 and line 9). Otherwise line 11 to line 21 computes LB1, the lower bound of the total time needed for the remaining unplanned jobs by relaxing the constraint of job arrival times. It is the total job processing time plus the minimum total gantry time that all the unplanned jobs will incur. The minimum total gantry time happens if each job is ready when the YC serves from the leftmost job to the rightmost job (or vice versa) after job J_i . Any other job sequence will not have less total gantry time. Therefore line 22 returns the lower bound to the completion time of the job sequence by relaxing the arrival times of the unplanned jobs. The code in line 23 to line 26 computes LB2, the minimum completion time of the job sequence without considering the YC gantry time between the unplanned jobs. In this case, the sequence with minimum YC idle time will return the earliest completion time of all jobs. Thus LB in line 27 that is the maximum of LB1 and LB2 can never overestimate the total time span to complete all jobs. It follows that no optimal solution will be pruned when the current search path is abandoned in our backtracking search by the test in line 28. Since each job is for one vessel only, we use C_i , a_i and d_i to represent C_{ih} , a_{ih} and d_{ih} respectively.

To accelerate the search process of the algorithm, in line 2 of MMS in Figure 5 we arrange the job list J into an order by SCJF (Smallest Completion time Job First). Intuitively, if the YC serves the job with the smallest completion time first, it is likely to be a good solution for completing all the jobs in the shortest time.

Since MMS algorithm plans the operations of each yard block completely independent of other yard blocks, we run the algorithm from the first block to the last block in each rolling planning window, not using the planning orders in Section 3.2.

```

// Names in italic are input parameters to the algorithm
 $J = \{J_1, J_2, \dots, J_n\}$ 
MMS ( $J$ ) {
1  plannedJ =  $\phi$ ; optimalJ =  $\phi$ ; CurSmallest =  $\infty$ ;
2  sort  $J$  into the smallest completion time first order;
3  B( $J$ , plannedJ);
}

B( $J$ , plannedJ) {
4  FOR each job  $J_i$  in  $J$  // Select  $J_i$  as the job to serve after jobs in plannedJ;
5       $C_i = \max(C_i + m_{ij}, a_i) + p$ ; // job  $J_i$  is the last job in plannedJ
6      IF  $J_i$  is the only job in  $J$ 
7          IF  $C_i < \text{CurSmallest}$ 
8              CurSmallest =  $C_i$ ;
9              optimalJ = plannedJ; //Store the current best job sequence
10     ELSE // more jobs to plan
11         LeftEnd = slot position of the leftmost job in  $J - \{J_i\}$  in the block;
12         RightEnd = slot position of the rightmost job in  $J - \{J_i\}$  in the block;
13          $r = |J| - 1$ ; // no. of jobs not planned yet
14         IF no job in  $J - \{J_i\}$  is on the left of  $J_i$ 
15             LB1 =  $r * p$  + gantry time from slot position of  $J_i$  to RightEnd;
16         ELSE IF no job in  $J - \{J_i\}$  is on the right of  $J_i$ 
17             LB1 =  $r * p$  + gantry time from slot position of  $J_i$  to LeftEnd;
18         ELSE IF  $J_i$  is closer to LeftEnd
19             LB1 =  $r * p$  + gantry time from slot position of  $J_i$  to LeftEnd then from LeftEnd to RightEnd;
20         ELSE
21             LB1 =  $r * p$  + gantry time from slot position of  $J_i$  to RightEnd then from RightEnd to LeftEnd;
22         LB1 = LB1 +  $C_i$ ;
23         put jobs in  $J$  into tempJ in ascending order of job arrival times without  $J_i$ ;
24         LB2 =  $C_i$ ;
25         FOR each job  $J_j$  in tempJ
26             LB2 =  $\max(\text{LB2}, a_j) + p$ 
27         LB =  $\max(\text{LB1}, \text{LB2})$ ;
28         IF LB < CurSmallest
29             Remove  $J_i$  from  $J$ ;
30             Append  $J_i$  to plannedJ;
31         B( $J$ , plannedJ);
}

```

Figure 5: Pseudocode of the MMS algorithm

3.4.2. Minimize Total Weighted Tardiness (MTWT)

To reduce the total (weighted) job tardiness is a common objective in job scheduling problems. We propose YC scheduling algorithm MTWT to minimize total weighted job tardiness to see how it performs in minimizing total weighted vessel tardiness. The input to

this algorithm is the same as those for MTWMT. The objective of our tree search is to find a path from the start node to a leaf node (a sequence of n jobs) that minimizes the total weighted job tardiness. In this case, the vessel weight is the job weight. The algorithm is presented in Figure 6. Since each job is for one vessel only, we use C_i , a_i and d_i to represent C_{ih} , a_{ih} and d_{ih} respectively.

Line 5 to line 7 in B computes the total weighted tardiness of the partially planned job sequence when J_i is added as the next job. For each remaining job j not planned yet, the minimum tardiness is $\max(0, \max(C_i + m_{ij}, a_j) + p - d_j)$. Line 13 to line 17 computes the lower bound of the total weighted tardiness of all job sequences where the planned partial sequence including J_i is the prefix. It is obvious that this lower bound never overestimates. So no optimal solution will be pruned when the current search path is abandoned in our backtracking search by the test in line 18.

To accelerate the search process of the algorithm, in line 2 of MTWT in Figure 6, we also sort the job list J into ascending order of job deadlines.

Since MTWT algorithm plans the operations of each yard block completely independent of other yard blocks, we run the algorithm to plan from the first block to the last block in each rolling planning window, as in Section 3.4.1.

```
// Names in italic are input parameters to the algorithm
 $J = \{J_1, J_2, \dots, J_n\}$ 
MTWT ( $J$ ) {
1  plannedJ =  $\phi$ ; optimalJ =  $\phi$ ; CurSmallest =  $\infty$ ;
2  sort  $J$  into the ascending order of job deadlines;
3  B( $J$ , plannedJ);
}

B( $J$ , plannedJ) {
4  FOR each job  $J_i$  in  $J$  // Select  $J_i$  as the job to serve after jobs in plannedJ;
5       $C_i = \max(C_i + m_{ij}, a_i) + p$ ; // job  $J_i$  is the last job in plannedJ
6       $T_i = \max(0, C_i - d_i)$ ;
7       $TWT_i = TWT_i + v_h * T_i$ ; //  $h$  is the vessel of job  $i$  and  $v_h$  is the vessel weight
8      IF  $J_i$  is the only job in  $J$ 
9          IF  $TWT_i < \text{CurSmallest}$ 
10             CurSmallest =  $TWT_i$ ;
11             optimalJ = plannedJ; //Store the current best job sequence
12      ELSE // more jobs to plan
13          LB =  $TWT_i$ ;
14          FOR each job  $J_j$  in  $J$ ,  $i \neq j$ 
15               $C_j = \max(C_i + m_{ij}, a_j) + p$ ;
16               $T_j = \max(0, C_j - d_j)$ ;
17              LB = LB +  $v_h * T_j$ ;
18          IF LB < CurSmallest
19              Remove  $J_i$  from  $J$ ;
20              Append  $J_i$  to plannedJ;
21              B( $J$ , plannedJ);
}
```

Figure 6: Pseudocode of the MTWT algorithm

4. Performance Evaluation

4.1. Design of Experiments

To evaluate the performance of the proposed YC scheduling system and algorithms, simulation experiments were carried out. The YC scheduling models are programmed in C++ language under Microsoft Visual Studio 2010 using Dell Precision T3500, Windows 7 64-bit OS, Intel(R) Xeon(R) CPU with 3.2GHz and 6GB RAM.

The descriptions of the problem parameters are given in Section 3.1. Parameter settings in the experiments were obtained from real world terminal models as in past projects (Guo et al. 2007). A yard block has a size of 36 slots. The time for a YC to move one container from its stack to a vehicle, P , will take between 75 seconds and 120 seconds. All YCs are identical. We choose P such that the average YC utilization is 85% under the job arrival rate used in the experiments.

The configuration of our experiments, namely, the number of vessels in the loading process and the vessel weights, the number of QCs, the number of YCs and the number of containers to plan and load are shown in Table 1. The vessel weights are randomly assigned. The weight distribution is given in “Vessel weights” column in Table 1. For example, “1:1,2:2,1:3” means 1 vessel of weight 1, 2 vessels of weight 2 and 1 vessel of weight 3. There are 4 QCs to serve each vessel.

Table 1: Configuration of experiments.

Experiment	No. of vessels	Vessel weights	No. of QCs	No. of yard blocks/YCs	No. of containers
A	2	1:1,1:2	8	12	2448
B	4	1:1,2:2,1:3	16	24	4896
C	8	3:1,3:2,1:3,1:4	32	48	9792

First, a total of 306 container jobs are generated as the loading list of each QC of each vessel. For each job in a loading list,

- (1) using a realistic QC operation rate and incorporating a small amount of randomness, the deadline of each job is a random variable drawn from $U(162, 198)$ seconds after the deadline of its previous job for the QC;
- (2) the job arrival time at the YC (the arrival time of a vehicle to pick up the container) is set to the job deadline minus a random variable drawn from $U(102, 360)$ seconds;
- (3) the slot and row number of the container in the yard block are randomly generated.

The YC gantry time between two jobs in the same yard block is based on a gantry speed of 7.8km/hour and their slot numbers.

After the loading lists are generated, the jobs in each QC’s list will be distributed in a round robin manner to a randomly-chosen, small number of yard blocks (we set it to 3 blocks for each QC in our experiments). Each YC is limited to getting jobs from 2 QCs only. These 2 QCs may be serving the same vessel, or may be serving different vessels of the same weight or of different weights.

After the distribution of the jobs in the loading lists of QCs to the YCs, each YC will get a list of jobs for the entire multi-vessel loading process. These job lists are processed by the rolling planning window scheme because job information will not be available from the beginning in real container operations. The size of the planning window is the number of jobs the YC scheduling algorithm needs to plan each time. After computing the job sequence in one planning window, a number of jobs at the front of the sequence will roll out for the YC to execute. The remaining jobs, together with new job arrivals will be planned in the next planning window. After some preliminary experiments, we found that 10 jobs in a planning window and 6 jobs to roll out gave us the best results. So after a YC scheduling algorithm produces a 10-job sequence from one planning window for a YC, the first 6 jobs in the sequence will be executed by the YC. The next planning window will work on the 4 remaining jobs plus 6 new jobs for this YC. This results in 34 planning windows to handle all the jobs that a YC will get.

We compare the YC scheduling systems in their effectiveness to minimize total weighted vessel loading time and in computational efficiencies. These systems are

- MTWMT: the YC scheduling system based on the MTWMT algorithm for each block/YC (Section 3.3) and the WEDD block order in each planning window (Section 3.2).
- MMS: the YC scheduling system based on the MMS algorithm for each block/YC and the numerical block order in each planning window (Section 3.4.1).
- MTWT: the YC scheduling system based on the MTWT algorithm for each block/YC and the numerical block order in each planning window (Section 3.4.2).
- RBA*: the YC scheduling system is based on the algorithm RBA* (Guo et al., 2011) for each block/YC and the numerical block order in each planning window. RBA* is a YC scheduling algorithm that minimizes the total job waiting time.
- EDD: the YC scheduling system is based on the Earliest Due Date First heuristic algorithm for each block/YC and the numerical block order in each planning window.

We also compare the three planning orders for yard blocks for the MTWMT scheduling system as discussed in Section 3.2.

4.2 MTWMT vs Others

In this section, we compare the performance of MTWMT with others.

4.2.1 Effectiveness in Minimizing Total Weighted Vessel Loading Time

The performance of various systems in YC scheduling is evaluated by how well they perform in minimization of (2). From the YC scheduling results of each independent run of the experiments, the tardiness of every job is recorded. The tardiness of a vessel h is

$$VT_h = \max_{q=1..Q_h} \left(\max_{i=1..N_q} (T_i) \right) \quad (12)$$

where Q_h is the number of QCs serving vessel h , N_q is the number of jobs for QC q and T_i is the tardiness of container i . The job list of QC q is the job list we generated for q before the jobs are distributed to the YCs. The total weighted tardiness of vessels is

$$TWVT = \sum_{h=1}^N v_h VT_h \quad (13)$$

where N is the number of loading vessels. It can be seen that (13) is the same as (2). The number of independent runs is decided such that the half width of the 90% confidence interval of $TWVT$ is less than 5% of mean of $TWVT$ itself.

We present the comparison of various systems x against MTWMT. The total weighted vessel tardiness ratio of system x against MTWMT is defined as

$$TWVTR(x, MTWMT) = \frac{TWVT(x)}{TWVT(MTWMT)} \quad (14)$$

Tables 2, 3 and 4 show the total weighted vessel tardiness ($TWVT$) in the multi-vessel loading process. They are the results of experiments A, B and C respectively. In experiment A, there are 2 sets: Set 1 and Set 2 represent two random assignments of vessel weights to the two vessels (1 vessel of weight 1 and 1 vessel has weight 2). In experiment B, there are 6 sets: Sets 1-6 represent six random assignments of vessel weights to the four vessels (1 vessel of weight 1, 2 vessels of weight 2 and 1 vessel of weight 3). In experiment C, there are 10 sets: Sets 1-10 represent 10 random assignments of vessel weights to the eight vessels (3 vessel of weight 1, 3 vessels of weight 2, 1 vessel of weight 3 and 1 vessel of weight 4). Each set has multiple independent runs and the average is reported. Tables 2, 3 and 4 also present the total weighted vessel tardiness ratio ($TWVTR$) of system x against MTWMT where x is one of the other systems used to plan for the YC operations as described in Section 4.1.

Table 2: The total weighted vessel tardiness, $TWVT$ in seconds and the total weighted vessel tardiness ratio of system x against MTWMT, $TWVTR(x,MTWMT)$ for Experiment A.

Weight assignment		MTWMT	MTWT	MMS	RBA*	EDD
Set 1	$TWVT$	282.03	413.40	1331.40	1331.38	666.27
	$TWVTR(x,MTWMT)$	1.00	1.47	4.72	4.72	2.36
Set 2	$TWVT$	286.28	408.51	1327.85	1339.94	647.41
	$TWVTR(x,MTWMT)$	1.00	1.43	4.63	4.68	2.26
Average	$TWVT$	284.15	410.96	1329.62	1335.66	656.84
	$TWVTR(x,MTWMT)$	1.00	1.45	4.68	4.70	2.31

Table 3: The total weighted vessel tardiness, $TWVT$ in seconds and the total weighted vessel tardiness ratio of system x against MTWMT, $TWVTR(x,MTWMT)$ for Experiment B.

Weight assignment		MTWMT	MTWT	MMS	RBA*	EDD
Set 1	$TWVT$	724.43	975.35	3314.23	3389.54	1489.27
	$TWVTR(x,MTWMT)$	1.00	1.35	4.57	4.68	2.06
Set 2	$TWVT$	708.48	922.02	3295.21	3393.68	1471.08
	$TWVTR(x,MTWMT)$	1.00	1.30	4.65	4.79	2.08
Set 3	$TWVT$	910.37	1193.10	3386.54	3507.05	1560.72
	$TWVTR(x,MTWMT)$	1.00	1.31	3.72	3.85	1.71
Set 4	$TWVT$	892.94	1146.35	3367.53	3511.18	1542.54
	$TWVTR(x,MTWMT)$	1.00	1.28	3.77	3.93	1.73
Set 5	$TWVT$	828.05	1102.63	3340.33	3493.50	1526.59
	$TWVTR(x,MTWMT)$	1.00	1.33	4.03	4.22	1.84
Set 6	$TWVT$	798.62	1046.84	3341.43	3407.22	1505.21
	$TWVTR(x,MTWMT)$	1.00	1.31	4.18	4.27	1.88
average	$TWVT$	810.48	1064.38	3340.88	3450.36	1515.90
	$TWVTR(x,MTWMT)$	1.00	1.31	4.12	4.29	1.88

Table 4: The total weighted vessel tardiness, $TWVT$ in seconds and the total weighted vessel tardiness ratio of system x against MTWMT, $TWVTR(x,MTWMT)$ for Experiment C.

Weight assignment		MTWMT	MTWT	MMS	RBA*	EDD
Set 1	$TWVT$	1110.96	1300.95	5991.01	6162.94	2035.87
	$TWVTR(x,MTWMT)$	1.00	1.17	5.39	5.55	1.83
Set 2	$TWVT$	1116.64	1299.72	5955.31	5958.09	1959.62
	$TWVTR(x,MTWMT)$	1.00	1.16	5.33	5.34	1.75
Set 3	$TWVT$	984.28	1136.93	5804.20	5852.03	1869.34
	$TWVTR(x,MTWMT)$	1.00	1.16	5.90	5.96	1.90
Set 4	$TWVT$	992.03	1126.70	5793.19	5887.57	1816.89
	$TWVTR(x,MTWMT)$	1.00	1.14	5.84	5.93	1.83
Set 5	$TWVT$	1078.42	1311.86	5955.59	6107.86	1961.04
	$TWVTR(x,MTWMT)$	1.00	1.22	5.52	5.66	1.82
Set 6	$TWVT$	1069.44	1216.47	5775.75	5881.79	1830.46
	$TWVTR(x,MTWMT)$	1.00	1.14	5.40	5.50	1.71
Set 7	$TWVT$	1041.12	1224.53	5810.87	5947.44	1858.92
	$TWVTR(x,MTWMT)$	1.00	1.18	5.58	5.71	1.79
Set 8	$TWVT$	1034.32	1222.13	5907.10	5961.47	1909.79
	$TWVTR(x,MTWMT)$	1.00	1.18	5.71	5.76	1.85
Set 9	$TWVT$	991.12	1146.01	5949.47	5973.00	1929.51
	$TWVTR(x,MTWMT)$	1.00	1.16	6.00	6.03	1.95
Set 10	$TWVT$	1170.57	1397.49	6011.21	6170.39	2058.65
	$TWVTR(x,MTWMT)$	1.00	1.19	5.14	5.27	1.76
average	$TWVT$	1058.89	1238.28	5895.37	5990.26	1923.01
	$TWVTR(x,MTWMT)$	1.00	1.17	5.57	5.67	1.82

These results clearly show that MTWMT is most effective in minimizing the total weighted vessel tardiness. MTWT is better than EDD. The simple heuristic EDD is a comparatively better policy than MMS and RBA*. These results show that (a) including job deadlines based on QC operations in the objective function in YC scheduling helps synchronize YC operations with the QCs; (b) including vessel weights in the objective function helps minimize the total weighted vessel tardiness; (c) minimizing the total weighted *maximum* tardiness among jobs of a vessel, instead of the *total* weighted tardiness of jobs, gives the best result in minimizing the total weighted vessel tardiness. We can deduce that minimizing the total weighted job waiting time is unlikely to do better than MTWMT. Of course it is not entirely fair to MMS, RBA* and EDD because these algorithms are designed to optimize other objective functions. Minimization of makespan (in MMS) improves the YC productivity and minimization of total vehicle waiting time (in RBA*) improves vehicle productivity. These commonly used objective functions also help (indirectly) to some extent in reducing vessel loading time. But they are much less effective in *ensuring* minimum total weighted vessel tardiness in comparison. The results also show that minimization of total weighted vessel tardiness in a multi-vessel loading process can be done by a standalone YC scheduling system.

To show the degree of variations in the results, we present the various percentiles of the total weighted vessel tardiness TWVT obtained by MTWMT for experiment B in Table 5. The percentiles are calculated from 50 independent runs for each set. They show acceptable variations in the results.

We evaluate the service quality received by vessels of different weights under the different YC scheduling systems. The mean of the tardiness of vessels of a certain weight (priority) w is computed by

$$VT(w) = \frac{1}{n_w} \sum_{h=1}^{n_w} VT_h \quad (15)$$

where n_w is the number of vessels with weight w in an experiment setup. This is the average tardiness experienced by vessels of weight w through their loading operations. Vessel weight distribution used in experiments is given in Table 1.

Table 5: The various percentiles of the total weighted vessel tardiness, *TWVT* in seconds obtained by algorithm MTWMT for Experiment B.

Weight assignment	10% percentile	Median	90% percentile	mean
Set 1	577.76	715.00	896.98	724.43
Set 2	551.90	705.70	868.36	708.48
Set 3	744.59	904.07	1102.00	910.37
Set 4	721.40	892.19	1079.90	892.94
Set 5	666.90	800.50	1073.50	828.05
Set 6	642.70	791.00	969.51	798.62

MTWMT and MTWT favour vessels with a higher weight and Table 6 shows that these vessels get shorter tardiness than lower weighted vessels. From the results in the various rows in column 3 and column 4, we see that the higher the vessel weight is, the lower the average vessel tardiness is for both MTWMT and MTWT. This explicitly confirms that they allow the terminal operator to provide differential treatment to vessels of different importance to it. The difference between the results of column 3 and that of column 4 shows again that minimizing maximum tardiness is significantly more effective than minimizing total tardiness in reducing vessel tardiness. For the vessels of the highest weight, the vessel tardiness from MTWMT is about one tenth to one seventh of the tardiness from MMS or RBA* and about

one fifth to one third of the tardiness from EDD. As expected, there is no significant difference in vessel tardiness among vessels of different weights for MMS, RBA* and EDD.

If we study each row in Table 6, we see that for vessels of any weight level, MTWMT is the best in minimizing the vessel tardiness, including vessels of the lowest weight level. MTWT comes second in most cases except for vessels of weight 1 in Experiment B where EDD is the second best. This shows that MTWMT is able to produce the lowest vessel tardiness for vessels of all weight levels, not sacrificing less important vessels to favour more important vessels. This confirms that while MTWMT tries to minimize the total weighted maximum tardiness of vessel jobs across yard blocks and planning windows, it does not just minimize the tardiness of the highest weighted vessels, the tardiness of lower weighted vessels are also kept as low as possible. For the vessels of the lowest weight, the vessel tardiness from MTWMT is about one third to one half of the tardiness from MMS or RBA* and about 8% to 20% better than the tardiness from EDD.

Table 6: The average tardiness of vessels of various weights, $VT(w)$ in seconds computed by (15).

	Vessel weight, w	MTWMT	MTWT	MMS	RBA*	EDD
Experiment A (2 vessels)	1	202.19	220.68	443.21	443.82	218.95
	2	40.98	95.14	443.21	445.92	218.95
Experiment B (4 vessels)	1	175.09	223.93	421.46	432.42	192.33
	2	110.80	138.49	413.76	430.17	186.64
	3	64.06	95.50	421.46	432.42	192.33
Experiment C (8 vessels)	1	101.62	117.11	371.80	373.36	121.77
	2	77.69	86.15	371.02	378.30	123.10
	3	46.13	57.47	365.71	361.09	114.94
	4	37.37	49.41	364.18	379.27	118.57

4.2.2 Computational Costs of YC Scheduling Algorithms

Each experiment has 34 planning windows to complete the multi-vessel loading process. Experiments A, B and C respectively have 12, 24 and 48 blocks to plan in each planning window. In Table 7a, row “total CPU time” presents the total computational time when a scheduling algorithm is run for all the yard blocks and all the planning windows in Experiments A, B, C. The average, the minimum and the maximum computational times are obtained from a few hundred runs. It can be seen that in our backtracking search algorithms which theoretically can degenerate into exhaustive search, the maximum computational times from a few hundred independent runs are not much longer than the average computational times. This means the bounds we use to prune the search space are very effective. The total CPU time in the independent runs of experiment C to plan for 48 YCs and 9792 containers has a minimum of 336 seconds and a maximum of 447 seconds with an average of 351 seconds.

Row “CPU time per planning window” in Table 7a shows the average computational time to plan for all yard blocks in one planning window. Row “CPU time per block-planning window” shows the average computational time to plan for one yard block in one planning window. Also shown are the ratios of the computational time of various algorithms to that of MTWMT in row “x/MTWMT”.

For all the algorithms, computing the optimal job sequence for one YC in one planning window takes less than 3 seconds on average. Computing the optimal job sequences for all the YCs in one planning window takes approximately between 3 seconds (MTWMT and MTWT) and 30 seconds (MMS) in Experiment B. They take approximately between 10 seconds (MTWMT and MTWT) and 120 seconds (MMS) in Experiment C. This means that

these algorithms are efficient to be used in real-time operation planning. Planning for the next window may start a few minutes before the end of the execution of the current YC plans. When YCs complete the execution of the current plans, the new plans are ready for them to continue operation.

Table 7a also shows that EDD consumes the least amount of computational time, which is expected, since it performs very little computation.

Table 7a: The computational time in seconds for Experiments A, B, C.

		MTWMT	MTWT	MMS	RBA*	EDD
Total CPU time (seconds)	A	average	23.87	22.17	190.47	40.11
		min, max	21.74, 26.29	21.14, 23.68	153.35, 231.63	37.81, 43.12
	B	average	96.05	99.74	1001.56	203.63
		min, max	88.2, 104.8	95.0, 105.9	809.24, 1267.28	192.8, 218.7
	C	average	350.58	358.42	4055.76	774.41
		min, max	336.4, 447.4	348.2, 377.4	3689.2, 4419.4	755.1, 817.4
CPU time per planning window (seconds)	A	0.702	0.652	5.602	1.180	0.277
	B	2.825	2.934	29.46	5.989	0.618
	C	10.311	10.542	119.287	22.777	1.448
CPU time per block-planning window (seconds)	A	0.059	0.054	0.467	0.098	0.023
	B	0.118	0.122	1.228	0.250	0.026
	C	0.215	0.220	2.485	0.475	0.030
x/MTWMT	A	1	0.93	7.98	1.68	0.39
	B	1	1.03	10.43	2.12	0.22
	C	1	1.02	11.57	2.21	0.14

Table 7b: The average number of nodes evaluated during tree search in Experiments A, B, C.

		MTWMT	MTWT	MMS	RBA*
Total no. of nodes	A	42635	46809	536068	104897
	B	84780	93897	1083471	209925
	C	169294.6	179589.2	2168135	406174.8
x/MTWMT	A	1	1.10	12.57	2.46
	B	1	1.11	12.78	2.48
	C	1	1.06	12.81	2.40

For MTWMT and MTWT they have similar computational costs and are much faster than MMS and RBA*. Table 7b shows supporting evidence for the efficiency of MTWMT and MTWT. In Table 7b, the “total No. of nodes” presents the average total number of nodes explored in the backtracking tree search when the each scheduling algorithm is run for all the YCs and all the 34 planning windows. The ratios of the number of nodes explored by the various algorithms to that of MTWMT are presented as “x/MTWMT”. Table 7b shows that MTWMT and MTWT explore much fewer number of nodes before finding the optimal solution than RBA* and MMS do. One reason is these two algorithms first evaluate a path (the leftmost path) of the tree in Figure 3 which is a job sequence in job deadline order. This sequence is a good solution for minimizing total weighted maximum tardiness for MTWMT or minimizing total weighted tardiness for MTWT. This solution’s objective function value is used in pruning other branches of the tree. In the remaining backtracking search, the search procedure will also explore alternative tree paths in increasing deadline order. This means tree paths which are neighbouring paths to the leftmost path in Figure 3 are examined first. In other words, solutions which make minor adjustments to the job sequence in deadline order are examined first. This part of the search space is where the optimal solution is likely to be and the other parts of the search space will mostly be pruned by the bound we use in the algorithms MTWMT and MTWT. Another reason for the short computational time is the small size of the planning window. Ten jobs are planned each time, which corresponds to 20 to 30 minutes of YC operations. Note that planning in small windows in our case does not

mean short-sightedness because of the use of rolling windows. The last four jobs in a planned job sequence in the current window will be replanned together with six new jobs in the next window.

4.3 Three Block Orders - WEDD, ECT and RAN

As discussed in Section 3.2, the different sequences of executing MTWMT on yard blocks in one planning window will produce different optimization results. We use Experiment B to examine the impact of the block execution order of MTWMT on the objective function results obtained. As mentioned before, six sets of randomly chosen vessel weights are assigned to the 4 vessels. Fifty to seventy independent runs are conducted for each set. Table 8 presents the mean and the 90% confidence intervals of the total weighted vessel tardiness and Table 9 the computational time for the three execution orders WEDD, ECT and RAN described in Section 3.2.

Table 8: The total weighted vessel tardiness, $TWVT$ in seconds when planning using MTWMT with different planning orders of yard blocks, with 90% confidence intervals.

	WEDD	ECT	RAN
Set 1	724.43±30.18	719.94±29.18	725.38±29.71
Set 2	708.48±29.44	696.11±27.75	698.44±28.40
Set 3	910.37±28.97	921.34±31.19	917.89±26.92
Set 4	892.94±31.54	894.93±31.62	900.88±33.29
Set 5	828.05±35.25	829.42±34.91	846.55±36.22
Set 6	798.62±31.92	786.00±31.14	787.22±30.88
Average	810.48±56.21	807.96±61.27	812.73±61.04

Table 9: The computational time in seconds when planning using MTWMT with different planning orders of yard blocks.

	WEDD	ECT	RAN
Set 1	96.05	93.09	94.42
Set 2	95.16	93.62	94.48
Set 3	96.53	94.73	93.43
Set 4	96.39	94.58	93.73
Set 5	96.12	94.04	93.57
Set 6	96.04	94.46	93.56
Average	96.05	94.09	93.87

Table 8 confirms that the different execution orders return results which are not identical. This is because that different execution orders among the blocks in the same planning window will lead to different $D_{h,(j-I)}$ in (10) or different MaxT in Figure 4 when a block is planned. However, Table 8 also shows that the different execution orders return results which are not statistically different. When the maximum tardiness of a vessel is increased by a job sequence of any block (line 11 in Figure 4) and that sequence eventually becomes the optimal solution (line 18 and line 25 in Figure 4), it affects the planning of all other blocks that have containers to be loaded to the same vessel. In fact, among the multiple blocks, the planning sequences are inter-dependent in a many-to-many relationship, not a one-to-one relationship. So there is not a universal optimal order for the blocks to do planning by the YC scheduling algorithm MTWMT. Table 9 shows that the different execution orders consume similar amounts of computational time.

This is a very good characteristic of the proposed yard crane planning system. It means that as vessels start and end their loading operations at different times and yard cranes complete their scheduling plans at different times, MTWMT's execution can be synchronized

with the completion of each yard crane's execution of a job sequence without the explicit notion of planning windows and a deliberate block planning order. The $D_{h,(j-1)}$ in (10) or MaxT in Figure 4 will be the current maximum tardiness of vessels incurred by the planning already done. MTWMT is able to ensure a final good result even though it cannot guarantee the optimal result for the whole loading process of all vessels.

5. Discussions

We have shown that minimization of the maximum tardiness caused by YCs produces the smallest vessel tardiness in the loading process. To minimize the whole vessel turnaround time, the unloading process should also be considered. We agree with Li et al. (2009, 2012) and Wu et al. (2015) that storage and retrieval operations at YCs should have different objectives. In their works, the objective is to minimize retrieval (i.e. vessel loading operations) earliness and tardiness and storage (i.e. vessel unloading operations) tardiness. In our opinion, maximum tardiness caused by YCs, rather than total tardiness, in the loading process delays the completion of loading a vessel. On the other hand, delays caused by YCs in the unloading process do not necessarily affect QC operations. They affect yard operations and how fast vehicles can be ready for the next transport jobs. Therefore minimization of the maximum tardiness caused by the YCs does not necessarily minimize vessel unloading time.

To help vehicles finish their transport jobs and be available for next jobs in the shortest time will help the smooth container flows for both vessel loading and unloading. Therefore the objective function to minimize total job waiting time will be the best objective function for the unloading process. As stated by Ng (2005) and Li et al. (2012), for unloading operations, minimization of total job waiting time is equivalent to minimization of total completion time. If we set the job deadline to be the job arrival time, which is what Li et al. (2009, 2012) did, minimization of total job waiting time is also equivalent to minimization of total tardiness.

So if a YC is doing loading operations only, minimization of total weighted maximum tardiness should be the objective function. If a YC is doing unloading operations only, minimization of total tardiness as just suggested should be the objective function. The algorithm MTWT in Figure 6 can be used after removing the vessel weights. If a YC has loading and unloading jobs in a planning window, a linear combination as follows will be a suitable objective function for YC scheduling.

$$\min_{\text{job sequences}} \left(\sum_{h=1}^{N_v} v_h L_h + \sum_{i=1}^{N_u} T_i \right) \quad (16)$$

where N_v is the number of loading vessels and N_u is the number of unloading jobs. The vessel weights v_h will be set higher than 1 such that all loading operations will have a higher weight than unloading operations. Note that all unloading operations have a weight of 1. Minor changes to the algorithm in Figure 4 will be able to compute the YC job sequence for this objective function.

6. Conclusions

We have proposed a standalone YC scheduling system which synchronizes the operations of multiple YCs with the QCs involved in the multi-vessel loading process. The objective of the YC scheduling system is to minimize the total weighted vessel tardiness which translates to the total weighted vessel loading time. It includes a complete and optimal algorithm MTWMT to minimize the total weighted maximum job tardiness. The YC scheduling system for multiple YCs has the ability to dynamically optimize future operations with the consideration of the delays already incurred to each vessel in earlier operations. This YC scheduling system is able to minimize total weighted vessel loading time much better than the

commonly used objectives in YC scheduling, namely, minimization of YC makespan and minimization of total vehicle waiting time. Our YC scheduling system also performs better than the scheduling algorithm to minimize total weighted job tardiness and the Earliest Due Date First heuristic. Minimizing total weighted vessel loading time directly help minimize total weighted vessel turnaround time, one of the most important objective of operation management in container terminals.

Our YC scheduling system based on MTWMT is able to provide differential treatment to vessels of different weight levels, the higher the weight level, the shorter tardiness they get in vessel loading. It is also able to produce the shortest tardiness, compared with existing YC scheduling algorithms, for vessels of every weight level, not just the vessels with the highest weight level. This shows that the usual objectives of minimizing total waiting time, total job tardiness, which treat all jobs equally and intend to optimize for all, are actually less helpful to all. It means that when allocating resources to operations, the objective should not be to try to complete every operation in a process as soon as possible. It should be to try to minimize further delay to the process.

The computational time required by MTWMT is only inferior to the simple heuristic EDD algorithm, making it suitable for YC planning in real terminal operations.

To facilitate the evaluation against MTWMT, a complete and optimal algorithm MMS to minimize makespan and a complete and optimal algorithm MTWT to minimize total weighted job tardiness are also proposed. MMS will be useful if YC productivity is the main concern in a container terminal. MTWT will be useful for YCs involved in unloading operations only as discussed in the last section.

The limitation of this study is when a QC scheduling system is not able to produce a schedule where all QCs have the same completion time. Sometimes a QC scheduling system deliberately makes a schedule where the QC(s) at one or both ends of the vessel will finish operations earlier and move to other vessels. Then minimizing the maximum tardiness among all containers of a vessel may not have the same effect as in the current study. Some extensions to the MTWMT algorithm in the principle of minimizing further delays to vessel loading completion may be studied in future. Another limitation is this study only covers the loading process, the discussions in the last section offer suggestions to both loading and unloading operations.

REFERENCES

- Ahmt J, Sigtenbjerggaard JS, Lusby RM, Larsen J and Ryan D (2016), A new approach to the Container Positioning Problem, *Flexible Services & Manufacturing Journal* 28:617–643.
- Allahverdi A, Ng CT, Cheng TCE and Kovalyov MY (2008), A survey of scheduling problems with setup times or costs, *European Journal of Operational Research* 187: 985–1032.
- Briskorn D and Angeloudis P (2016), Scheduling co-operating stacking cranes with pre-determined container sequences, *Discrete Applied Mathematics* 201:70-85.
- Briskorn D, Emde S and Boysen N (2016), Cooperative twin-crane scheduling, *Discrete Applied Mathematics* 211:40-57.
- Cao JX, Lee DH, Chen JH and Shi Q (2010), The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods, *Transportation Research Part E* 46:344–353.
- Cao Z, Lee DH, Meng Q (2008), Deployment strategies of double-rail-mounted gantry crane systems for loading outbound containers in container terminals, *International Journal of Production Economics* 115:221–228.

- Carlo HJ, Vis IFA, Roodbergen KJ (2014), Storage yard operations in container terminals: Literature overview, trends, and research directions, *European Journal of Operational Research* 235(2):412-430.
- Chen JH, Lee DH, Goh, M (2014), An effective mathematical formulation for the unidirectional cluster-based quay crane scheduling problem, *European Journal of Operational Research* 232:198-208.
- Chen L, Bostel N, Dejax P, Cai J and 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.
- Chen L and Langevin A (2011), Multiple yard cranes scheduling for loading operations in a container terminal, *Engineering Optimization*, 43(11):1205-1221.
- Chen L, Langevin A and Lu Z (2013), Integrated scheduling of crane handling and truck transportation in a maritime container terminal, *European Journal of Operational Research* 225:142-152.
- Dorndorf U and Schneider F (2010), Scheduling automated triple cross-over stacking cranes in a container yard, *OR Spectrum* 32(3):617-632.
- Froyland G, Koch T, Megow N, Duane E, Wren H (2008), Optimizing the landside operation of a container terminal, *OR Spectrum* 30(1):53-75.
- Gharehgozli AH, Yu Y, de Koster R, Udding JT (2014), An exact method for scheduling a yard crane, *European Journal of Operational Research* 235(2):431-447.
- Gharehgozli AH, Laporte G, Yu Y, de Koster R (2015), Scheduling twin yard cranes in a container block, *Transportation Science* 9(3):706-719.
- Graham RL, Lawler EL, Lenstra JK, Kan AHGR (1979), Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics* 5: 287-326.
- Guo X, Huang SY, Hsu WJ, Low MYH, Chan TH, Liu JH (2007), Vehicle Dispatching with real time location information in container terminals, In *Proceedings of the European Modeling and Simulation Symposium*.
- Guo X, Huang SY, Hsu WJ, Low MYH (2011), Dynamic Yard Crane Dispatching in Container Terminals with Predicted Vehicle Arrival Information, *Advanced Engineering Informatics* 25(3):472-484.
- Guo X, Huang SY (2012), Dynamic Space and Time Partitioning for Yard Crane Workload Management in Container Terminals, *Transportation Science* 46(1):134-148.
- Hu ZH, Sheu JB, Luo JX (2016), Sequencing twin automated stacking cranes in a block at automated container terminal, *Transportation Research Part C* 69:208-227.
- Huang SY, Guo X and Tay TC. (2011), An Improved Least Cost Heuristic for Dynamic Yard Crane Deployment in Container Terminals. In the 7th Annual IEEE Conference on Automation Science and Engineering (CASE 2011): 84-89. Trieste, Italy: Institute of Electrical and Electronics Engineers, Inc.
- Huang SY, Tay ZC (2013), A Framework for Automated Real Time Management of Container Handling Equipment. The 2013 International Conference on Logistics and Maritime Systems.
- Huang SY, Li Y, Guo X (2014), Yard Crane Dispatching to Minimize Vessel Turnaround Times in Container Terminals. The 2014 Winter Simulation Conference.
- Huang SY, Li Y, Fan X (2015), TwinCrane-ATCRSS-Game: Job Dispatching with Lookahead for Twin Yard Cranes. The 5th International Conference on Logistics and Maritime Systems.
- Jung SH, Kim KH (2006), Load scheduling for multiple quay cranes in port container terminals, *Journal of Intelligent Manufacturing* 17:479-492.

- Kemme N (2010), RMG crane scheduling and stacking, In J. W. Bose (eds), *Handbook of Terminal Planning*, Springer.
- Kim KH, Kang JS, Ryu KR (2004), A beam search algorithm for the load sequencing of outbound containers in port container terminals, *OR Spectrum* 26:93–116.
- Kim KM, Kim KY (1999), An optimal routing algorithm for a transfer crane in port container terminals, *Transportation Science* 33(1):17–33.
- Kim KY, Kim KH (2003), Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals, *Naval Research Logistics* 50:498-514.
- Kim Y, Kim T and Lee H (2016), Heuristic algorithm for retrieving containers, *Computers & Industrial Engineering* 101:352–360.
- Kumar MM, Omkar SN (2008), Optimization of yard crane scheduling using particle swarm optimization with genetic algorithm operators (PSOGAO), *Journal of Scientific & Industrial Research* 67:335-339.
- Lau HYK, Zhao Y (2008), Integrated scheduling of handling equipment at automated container terminals, *International Journal of Production Economics* 112:665-682.
- Lee DH, Cao Z, Meng Q (2007), Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm, *International Journal of Production Economics* 107:115-124.
- Lee DH, Cao JX, Shi Q and Chen JH (2009), A heuristic algorithm for yard truck scheduling and storage allocation problems, *Transportation Research Part E* 45:810-820.
- Lee DH, Chen JH (2010), An improved approach for quay crane scheduling with non-crossing constraints, *Engineering Optimization* 42(1):1-15.
- Lenstra JK, Kan AHGR, Brucker P (1977), Complexity of machine scheduling problems, *Annals of Discrete Mathematics* 1: 343-362.
- Li W, Wu Y, Petering M, Goh M, d. Souza R (2009), Discrete time model and algorithms for container yard crane scheduling, *European Journal of Operational Research* 198:165-172.
- Li W, Goh M, Wu Y, Petering M, d. Souza R, Wu YC (2012), A continuous time model for multiple yard crane scheduling with last minute job arrivals, *International Journal of Production Economics* 136:332-343.
- Meersmans PJM and Wagelmans APM (2001), Effective algorithms for integrated scheduling of handling equipment at automated container terminals, Erasmus Research Institute of Management (ERIM), ERIM Report Series Reference No. ERS-2001-36-LIS.
- Narasimhan A, Palekar US (2002), Analysis and Algorithm for the Transtainer Routing Problem in Container Port Operation, *Transportation Science* 36(1):63-78.
- Ng WC (2005), Crane scheduling in container yards with intercrane interference, *European Journal of Operational Research* 164:64-78.
- Ng WC, Mak KL (2005a), An effective heuristic for scheduling a yard crane to handle jobs with different ready times, *Engineering Optimization* 37(8):867-877.
- Ng WC, Mak KL (2005b), Yard crane scheduling in port container terminals, *Applied Mathematical Modelling* 29(3):263-276.
- Park T, Choe R, Ok SM, Ryu KR (2010), Real-time scheduling for twin RMGs in an automated container yard, *OR Spectrum* 32:593-615.
- Petering MEH, Wu Y, Li W, Goh M, d. Souza R (2009), Development and simulation analysis of real-time yard crane control systems for seaport container transshipment terminals, *OR Spectrum* 31:801-835.
- Pinedo ML, *Scheduling: theory, algorithms, and systems* (2016), Fifth edition, Springer.
- Saenen Y, van Valkengoed M (2005), Comparison of three automated stacking alternatives by means of simulation, *Proceedings of the winter simulation conference 2005*, 1567–1576.
- Speer U and Fischer K (2017), Scheduling of Different Automated Yard Crane Systems at Container Terminals, *Transportation Science* 51(1):305-324.

- Stahlbock R, Voss S (2010), Efficiency consideration for sequencing and scheduling of double-rail-mounted gantry cranes at maritime container terminals, *International Journal of Shipping and Transport Logistics* 2(1):95-123.
- Steenken D, Voss S, Stahlbock R (2004), Container terminal operation and operations research – a classification and literature review, *OR Spectrum* 26:3-49.
- Vis IF, Carlo HJ (2010), Sequencing two cooperating automated stacking cranes in a container terminal, *Transportation Science* 44(2):169-182.
- Wu Y, Li W, Petering MEH, Goh M, de Souza R (2015), Scheduling multiple yard cranes with crane interference and safety distance requirement, *Transportation Science* 49(4):990-1005.
- Yu M, Qi X (2013), Storage space allocation models for inbound containers in an automatic container terminal. *European Journal of Operational Research* 226(1):32–45.
- Zeng Q, Yang Z (2009), Integrating simulation and optimization to schedule loading operations in container terminals, *Computers & Operations Research* 36(6):1935–1944.