A Continuous Time Model for Multiple Yard Crane Scheduling with Last Minute Job Arrivals

Wenkai Li\textsuperscript{ab*}, Mark Goh\textsuperscript{bcd}, Yong Wu\textsuperscript{be}, M. E. H. Petering\textsuperscript{bf}, R. de Souza\textsuperscript{b}, Y. C. Wu\textsuperscript{ef}

\textsuperscript{a}Graduate School of International Management, International University of Japan, Niigata 949-7277, Japan
\textsuperscript{b}The Logistics Institute-Asia Pacific, 21 Heng Mui Keng Terrace, #04-01, Singapore 119613
\textsuperscript{c}NUS Business School, National University of Singapore, 1 Business Link, Singapore 117592
\textsuperscript{d}School of Management, University of South Australia, Adelaide, 5001
\textsuperscript{e}Dept. of International Business & Asian Studies, Griffith University, Gold Coast Campus, QLD 4222 Australia
\textsuperscript{f}Industrial and Manufacturing Engineering Department, University of Wisconsin–Milwaukee, USA.
\textsuperscript{g}Dept. of Business Management, National Sun Yat-Sen University, 70, Lienhai Rd, Kaohsiung 80424, Taiwan

Keywords: Container yard operations, Continuous time, Scheduling, Rolling-horizon algorithm, Last minute jobs.

Abstract

Container terminal (CT) operations are often bottlenecked by slow YC (yard crane) movements. Efficient YC scheduling to reduce the PM waiting time is therefore critical in increasing a CT’s throughput. This paper develops an efficient continuous time MILP model for YC scheduling. The model treats realistic operational constraints such as multiple inter-crane interference, fixed YC separation distances, simultaneous container storage/retrievals, realistic YC acceleration/deceleration stages and gantry time, and require far fewer integer variables than previous work.

The model significantly improves the solution quality compared to the existing discrete time models and other heuristics found in the literature. Using heuristics and

\*Corresponding author. E-mail: lwk@iuj.ac.jp; wenkaili2002@yahoo.com. Tel: (81)257791516. Fax: (81)257791187
\dagger Y. C. Wu was supported by NSC 99-2410-H-110-053-MY3
a rolling-horizon algorithm, our model can solve actual container yard (CY) problems quickly and robustly in polynomial time. Also, to cope with the last minute container arrivals which can disrupt routine CT operations, two methods for handling these last minute job insertions are discussed and compared.

1. Introduction

1.1 Background

Increased reliance on maritime trade within Asia necessitates growth in containers, carriers, container terminals (CT), and other container port related equipment such as the yard and quay cranes and prime movers. A recent industry report mentioned that intra-Asian trade (excluding the Middle East, Indian sub-continent and Australasia) was 28.6 million TEU in 2007, or equivalent 20% of the total global trade (eyefortransport 2008). This excludes any business moving within its confines on a feeder basis which is bound for markets in Europe, the US, and South America. This volume is expected to reach 50.7 million TEU by 2013. Any operational efficiency is thus constrained by the bottlenecks residing at the various points of the maritime supply chain, particularly at the landside operations. With greater expected container volumes and port calls, some of the major ports in Asia are already operating near or exceeding their capacity such as India (Froyland et al. 2008). As nodal players in the supply chain, CT operators face an imperative to increase container flow efficiency through the container yard (CY). Sound port planning and smooth execution of the equipment flow within the CY is thus a competitive advantage for success. However,
practical constraints limit the CY from attaining its desired efficiency. Slower than desired YC movements and last minute container arrivals into a CY often impede the performance of CT operations.

We now examine some of these operating constraints and highlight how the literature has attempted to reconcile these constraints in their models. First, in a work safe environment, regulatory compliance dictate that no two yard cranes (YC) traversing on an identical bi-directional lane can cross gantry each other. This is to reduce the possibility of YC-YC collisions, which can further hamper the throughput rate. This constraint was first studied by Lim et al. (2004) and has been later well examined and modeled in the literature (e.g. Ng 2005; Zhu et al. 2006; Lim et al. 2007; Cao et al. 2008; Froyland et al. 2008; Li et al. 2009; Wen et al. 2010). Second, to prevent tail-gating, some ports impose a minimum safety distance of 160 feet between any two YC’s. This constraint affects container pick efficiency as it delays and sometimes limits access to the containers residing in the intervening slots. Li et al. (2009) discussed this issue. Third, in Asian CTs such as Singapore and Hong Kong, due to the constraint of space and for better land use, the containers for storage and retrieval are normally kept together. This naturally lengthens the retrieval time (including search time) and thus requires more planning operationally. A reasonable approach to improving the crane utilization is to minimize the dead head time in these CTs. Put simply, it is better for yard cranes to be able to put a container in its rightful slot position in the yard and then pick up another container almost immediately, to
minimize unnecessary travel time within the yard. However, to handle the “simultaneous” drop and pick of containers requires much advanced planning. For expediency, yard operators tend to limit the number of simultaneous pick and drop moves in favor of the trade-off for idle YC’s. Most of the literature (e.g. Ng 2005; Ng and Mak 2005) consider only storage moves. This paper develops an approach for generating realistic YC work schedules, taking into account the above operating constraints, i.e. inter-crane interferences, fixed YC separation distances, and simultaneous storage/retrieval jobs handling.

1.2. Literature Review

Simulation models for CT operations in the literature address various decision problems (Vis 2006; Petering et al. 2009). For instance, Vis (2006) compared the performance of manned straddle carriers (SC) and automated stacking cranes using simulation models. Petering et al. (2009) showed that appropriate yard crane dispatching policies have a major influence on a terminal’s long-run average quay crane rate using a fully-integrated, discrete event simulation model. Scheduling algorithms for multiple YC’s are also being addressed in the literature (Stahlbock et al. 2008). For instance, Kim et al. (1999) developed a mixed integer programming model to schedule the load sequence for a single SC in a port container terminal. The objective is to minimize the total travel distance of the SC by optimally determining the SC visit sequence. Lee et al. (2007) considered the scheduling problem of two-transtainer systems. The container bay visit sequence and the number of
containers picked up at each visit of the two transtainers are determined. A simulated annealing (SA) algorithm is developed to solve the proposed mathematical model. However, they do not generate detailed work schedules for the YC’s and individual containers. Kim et al. (2003) sequenced the storage and retrieval jobs for a single YC in an YC block using dynamic programming and heuristics. In addition, Lim et al. (2004) also studied quay crane scheduling with spatial and separation constraints. The problem was modeled as a bipartite graph matching. They showed that a squeaky wheel optimization with a local search approach gives good results within a short time. Zhu et al. (2006) proved that the crane scheduling problem is NP-complete. They designed a simulated annealing framework with a graph-search-based neighbourhood search to solve quay crane scheduling problems with non-crossing constraints. Near optimal solutions for instances of various sizes can be obtained. Recently, Cao et al. (2008) formulated a discrete time integer model for a double-rail-mounted gantry (DRMG) crane scheduling problem which involves different crane interferences. In their paper, two DRMG cranes can share the same handling lane but cannot work simultaneously at the same slot. Defining a one-time period as the time required for an RMG to travel the distance of a single slot, the total time periods involved in their model can be very large in a two-hour time window. A combined scheduling heuristic is designed to solve the problem. Ng (2005) has proposed a heuristic for constructing detailed work schedules for multiple YC’s in a block with inter-crane interference, albeit without retrieval moves and minimum crane-distance separation. However, most of the algorithms applied on the YC scheduling problem (Ng 2005, Ng and Mak
2005, Ng and Tsang 2005) only consider storage moves, presumably as it is easier to develop algorithms for storage moves. Recently, Cao et al. (2010) solved an integrated PM and YC model for retrieval jobs using Benders’ decomposition but they did not consider YC interference. Wen et al. (2010) developed approaches for scheduling multiple cranes travelling over a shared network of tracks in a shipbuilding environment. Inter-crane interference is the major factor affecting make-span and crane utilization. They modeled the problem as a multi-commodity flow problem with side constraints and proposed Priority/ACO (Ant Colony Optimization) heuristics to solve the problem. Chen and Lu (2012) addressed the storage location assignment problem for outbound containers. The problem is decomposed into two stages. The yard bays and the amount of locations in each yard bay are first determined by a mixed integer programming model and the exact storage location for each container is obtained with a hybrid sequence stacking algorithm in the second stage. We note that the results from Chen and Lu (2012) may be used as inputs for our paper.

1.3. Contributions

In Li et al. (2009), a discrete-time model was developed for YC scheduling by taking into account the realistic operational constraints of a large CT such as the inter-crane interference, fixed YC separation distances, and simultaneous container storage/retrievals. The solution quality and solution time using a rolling-horizon algorithm are both better than those found in the literature. Studying the same problem but using an improved model, we extend this work in this paper with the
following fivefold contributions:

1. Significantly improved solution quality.

The continuous time model significantly improves the solution quality compared to a discrete-time model solving the same dataset. Most models in the literature (Ng 2005, Petering et al. 2006, Li et al. 2009) which consider YC operational constraints are discrete-time. Discrete-time models discretize the time horizon into a number of time intervals of uniform duration. Events such as the beginning and end of a task are associated with the boundaries of these time intervals. As the actual handling times of the container moves are continuous, discrete-time MILP models can only approximate the scheduling problem. One usually needs to use a sufficiently small time interval, for example, the greatest common factor of the processing times to achieve a suitable approximation of the original problem. However, this will significantly increase the number of integer variables associated with each discrete time interval leading to an intractable model size and much longer computational time. The main limitation of a discrete-time MILP model is its discrete approximation of the continuous time horizon. Avoiding synchronizing container moves to discrete time grids, a continuous-time model schedules YC movements to their earliest possible times leading to significantly reduced PM waiting times. However, despite the advantages of a continuous time model, only a few papers model the YC scheduling problem as a continuous time model due to the difficulty in problem formulation (Ng and Mak 2005; Ng and Tsang 2005). Also, these continuous time models consider only storage
jobs and only one YC. Besides the job sequencing constraints, other operational constraints (e.g. YC interference) are ignored. A YC schedule from these continuous time model may not be directly applicable to actual YC operations. Thus, this paper develops an efficient continuous-time MILP model taking into account realistic CT operating constraints.

2). Ability to handle large size problems.
Without discretizing the time horizon, the number of binary variables is significantly reduced in the continuous-time MILP model developed in this paper. As a result, the model size is reduced and larger instances with the number of jobs ranging from 45 to 69 in a 2-hour time window can be solved quickly in this paper. This is much larger than the size of the instances handled (20-30 jobs) in Li et al. (2009). From the YC operational data collected from the CT’s, the number of jobs can range from 0 to 60 at each block (Ng 2005). Thus our model is applicable to actual YC scheduling situations.

3). Reduced solution time
Applying appropriate heuristics and extending the rolling-horizon algorithm proposed in Li et al. (2009) to the continuous time model, the solution time to the same datasets (20-30 jobs) are faster (1.6 seconds on average in this paper compared to 4.6 seconds on average in Li et al. 2009), due mainly to the reduced model size.
4). More realistic operational constraints considered.

The following constraints are considered for the first time in this paper as they have been ignored in the discrete time models (see for example Li et al. 2009).

- The minimum YC gantry time is defined for consecutive moves at different YC’s by taking into account the idle YC’s that sit in between two working YC’s and four cases of YC order and slot difference (see Section 2.2.4).

- When container moves are handled by neighboring YC’s, the definition of the corresponding set is updated so that it varies by the order number of the YC’s (see Section 2.2.5). This update takes into account the distance of the idle YC’s that sit in between two working YC’s.

- For jobs located at the leftmost/rightmost slots, besides the constraints on the first and the last YC (Li et al. 2009), additional constraints are enforced to prevent certain YC’s from handling them (see Section 2.2.11). Enforcing these implicit constraints originating from the multiple inter-crane interference, operational constraints explicitly in the model can reduce the number of binary variables to be determined and hence significantly reduce the solution time.

5). Handling last-minute jobs

Last-minute container arrivals into the CT disrupt the normal CT operations and may impede CT performance. A CT has to handle these jobs and recover to normal CT operations as soon as possible. Two methods in the rolling-horizon algorithm are discussed and compared in this work to schedule last-minute container arrivals.
This paper is organized as follows. Section 1 provides the introduction and literature review of the crane scheduling problem. The contributions of this paper are summarized. Section 2 presents an efficient continuous-time MILP model. The assumptions applied in the model and computational results for smaller instances are provided and compared with the literature. In section 3, heuristics and a rolling-horizon algorithm (RHA) are proposed to solve large problems. Section 4 discusses approaches for handling last-minute jobs and presents the summary of the results and conclusions of the paper. Finally, Appendix A defines the sets, variables, and parameters and Appendix B presents the constraints of the sub-problems in RHA.

2. Continuous-time model (CMIP1)

We now develop an efficient continuous-time MILP model taking into account all YC operating conditions. As the YC’s handling and gantrying activities are realistically modeled, CT applicable and optimal schedules are guaranteed. Without defining the discrete time intervals, the size of continuous time model is smaller than its discrete-time based counterpart (Li et al. 2009). Much larger size problems (up to 69 jobs in this work versus at most 32 jobs in Li et al. 2009) are handled in this work allowing it to cope with realistic YC operating conditions. Our model can deal with either multiple YC or single YC scenarios with minor modifications to the constraints. In the following sections, we present our formulations based on the multiple YC scenario. If changes are needed to suit the single YC scenario, we will point out the
differences of the formulations for the single YC scenario for completeness sake.

2.1 Objective Function

Appendix A contains the notations used for the indices, sets, parameters and variables in the mathematical formulation whose objective is to minimize a linear combination of the total amount of retrieval earliness, and storage and retrieval delays,

\[
\text{Minimize } TC = W_{re} \sum_{m,b} RE_m + W_{rd} \sum_{m,b} RL_m + W_{sd} \sum_{m,b} STL_m
\]

Minimizing \( TCT \) is used as the objective in most of the literature (Ng 2005). Only for problems with pure storage moves, minimizing \( TCT \) is equivalent to minimizing \( PMWT \). If retrieval moves are involved, minimizing \( TCT \) will cause retrieval moves to be scheduled as early as possible and the \( PMWT \) becomes large. Further, minimizing \( TCT \) glosses over the true gap away from the optimal solution, as the deviation from the optimal \( TCT \) decreases with the increase in job arrival and job finish times. In eq. (1), increasing the weight for \( RL, W_{rd} \), the model may increase the total \( RE \) to reduce the total \( RL \). This may cause the \( TCT \) and \( PMWT \) of the optimal solution to be larger than the sub-optimal solution. In this case, \( TC \) as applied in eq. (1) is the best performance measure.

2.2 Constraints

We now present the following constraints applicable to YC scheduling. Without any loss of generality, we assume that the slots of a block are ordered from left to right
and the moves are ordered according to the slot number of their location, i.e., if \( m < n \), then \( slot_m \leq slot_n \). The YCs are also ordered from left to right as they cannot cross gantry each other.

2.2.1 A container move should be assigned to a YC, i.e.,

\[
\sum_{c \in C} W_{mc} = 1, \quad \forall m \in M, NC > 1
\]

(2)

With only one YC, \( W_{mc} \) is fixed at 1, effectively deleting this variable. To maintain the consistency of constraints across multi/single-YC scenarios, we keep this variable and fix the value of \( W_{mc} \) to 1 i.e. \( W_{mc} = 1, \quad \forall m \in M, NC=1 \).

2.2.2 Identifying overlapping container moves \( n \) against move \( m \)

Different from the discrete-time formulations in which nearby container moves are synchronized to the boundaries of time intervals, container moves can overlap to any extent in a continuous-time formulation. Overlapping container moves (referred to as \( n, n \neq m \)) against move \( m \) are moves whose job finish times, \( T_{fn} (= T_{sn} + HT) \), are greater than the job start time of move \( m \), \( T_{sm} \), and their job start time, \( T_{sn} \), is smaller than the job finish time of move \( m \), \( T_{fm} \). Figure 1 illustrates the definition of overlapping and non-overlapping container moves where each box represents a container move with corresponding job start/finish times shown at the left/right side respectively. Figure 1 shows that only moves 3 and 4 are overlapping moves against move \( m \). To identify the overlapping moves, 0-1 variables \( A1_{mn} \) and \( A2_{mn} \) are defined:
\[ A_{1mn} = \begin{cases} 
1 & \text{if } T_{e_m} > T_{s_n} \\
0 & \text{otherwise} 
\end{cases} \]

That means if move \( n \) happens before move \( m \) and they are non-overlapping, then \( A_{1mn} = 0 \). For example, move 1 in Figure 1. Also,

\[ A_{2mn} = \begin{cases} 
1 & \text{if } T_{s_n} < T_{e_m} \\
0 & \text{otherwise} 
\end{cases} \]

That means if move \( n \) happens after move \( m \) and the two moves are non-overlapping, then \( A_{2mn} = 0 \), for example, move 6 in Figure 1.

For moves whose job finish time equals the job start time of move \( m \), i.e., \( T_{e_n} = T_{s_m} \), or moves whose job start time equals the job finish time of move \( m \), i.e., \( T_{s_n} = T_{e_m} \), we define these moves as non-overlapping moves against \( m \). For example, moves 2 and 5 in Figure 1 are defined as non-overlapping moves against move \( m \). It is clear from Figure 1 that, both \( A_{1mn} \) and \( A_{2mn} \) of overlapping moves take a value of 1. We can then identify the overlapping moves using \( A_{1mn} \) and \( A_{2mn} \), and constraints (3a) to (4b) as follows:

\[ T_{e_n} + HT \geq T_{s_m} - BM \ast (1 - A_{1mn}), \forall m,n \in M, m \neq n, NC > 1 \]

(3a)

\[ T_{s_m} \geq T_{e_n} + HT - BM \ast A_{1mn}, \forall m,n \in M, m \neq n, NC > 1 \]

(3b)

\[ T_{s_m} + HT \geq T_{s_n} - BM \ast (1 - A_{2mn}), \forall m,n \in M, m \neq n, NC > 1 \]

(4a)

\[ T_{s_n} \geq T_{s_m} + HT - BM \ast A_{2mn}, \forall m,n \in M, m \neq n, NC > 1 \]

(4b)
Constraints (3a) and (3b) define the value of $A_{1mn}$. For move 1 in Figure 1, $T_{sn} + HT < T_{sm}$, this forces $A_{1mn}$ to take a value of zero, otherwise (3a) is violated. (3b) is trivially satisfied when $A_{1mn}$ is zero. For move 3 in Figure 1, the value of $A_{1mn}$ is forced to 1 as $T_{sn} + HT > T_{sm}$. Similarly, constraints (4a) and (4b) are used to define the value of $A_{2mn}$. Note that constraints (3a) to (4b) are only used for scenarios with multiple cranes. For scenarios with a single YC, all moves are handled by one YC and thus there are no overlapping moves (a YC can only handle one container move at a time). Constraints (4a)-(4b) can be applied to single YC scenarios by eliminating $HT$ and changing the condition $NC>1$ to $NC=1$. For single-crane scenarios, $A_{1mn}$ is not needed and we can set its value to 0 i.e. $A_{1mn} = 0$, $\forall m,n \in M$, $m \neq n$, $NC=1$.

Figure 1 shows that, if move $n$ is used as the reference move, then when $A_{1mn} = 0$, we have $A_{2nm} = 0$; when $A_{1mn} = 1$, we have $A_{2nm} = 1$, etc. In short, $A_{1mn} = A_{2nm}$ and $A_{2nm}$ can replace $A_{1mn}$. Thus, $A_{1mn}$ and constraints (3a)-(3b) can be eliminated from the model for either multi or single-crane scenarios. Figure 1 also shows that, when both $A_{1mn} = 1$ and $A_{2mn} = 1$, moves $m$ and $n$ overlap. Thus, we can use $(A_{1mn} + A_{2mn})$, or equivalently $(A_{2nm} + A_{2mn})$, to identify overlapping moves against move $m$.

### 2.2.3 Relation of job start times of consecutive container moves at the same YC

If move $n$ happens after $m$, and both jobs $m$ and $n$ are handled by the same YC, then constraint (5), valid for both multi/single-crane scenarios, holds:

$$
T_{sn} \geq T_{sm} + HT + GH_{mn} - [3W_{mc} - W_{nc} - (1 - A_{2mn})]*BM, \\
\forall m,n \in M, m \neq n, \forall c \in C
$$
If both moves $m$ and $n$ are handled by YC $c$ (i.e. $W_{mc}=W_{nc}=1$), and move $n$ happens after $m$ (i.e. $A_{2mn}=0$), then job $n$ should start after the job finish time of move $m$, $T_{sm}+HT$, plus the YC gantry time from slot $m$ to slot $n$, $GH_{mn}$. The parameter $GH_{mn}$ is defined as:

$$GH_{mn} = \begin{cases} |slot_m - slot_n|*0.06 + 0.09 & \text{when } slot_m \neq slot_n \\ 0 & \text{when } slot_m = slot_n \end{cases}$$

$|slot_m - slot_n|$ is the absolute slot number difference. If moves $m$ and $n$ are handled by the same YC consecutively and they are located at the same slot ($slot_m = slot_n$), then the YC does not need to move and the gantry time is zero. If the moves are located at different slots, the gantry time (in minutes) of the YC is $|slot_m - slot_n|*0.06 + 0.09$ by taking into account the YC acceleration/deceleration stages.

### 2.2.4 Relation of job start times of consecutive container moves at different YC’s

If move $n$ happens after move $m$, and moves $m$ and $n$ are handled by different YC’s, then constraint (6) holds:

$$T_{n} \geq T_{m} + HT + OT_{mn} - [3-W_{mc}-W_{nc}-(1-A_{2mn})]*BM, \forall m,n \in M, \forall c,c' \in C, m \neq n, c \neq c', NC > 1$$

Constraint (6), valid for multi-crane scenarios, states that if moves $m$ and $n$ are handled by different YC’s (i.e. $W_{mc}=1$, $W_{nc}=1$, $c \neq c'$), and move $n$ happens after $m$ (i.e. $A_{2mn}=0$), then job $n$ should start after the job finish time of move $m$, $T_{sm}+HT$, plus the YC gantry time, $OT_{mn}$. The definition of $OT_{mn}$ is shown in Figure 2. The height and width of a box in Figure 2 represents the handling time and the influencing slots of a move respectively. The slot number of a move locates at the center of the box. As when a move is being handled by a YC, a neighboring YC should stay $SEP$ slots away.
from it, we use $SEP$ to represent the width of a box. On the left of Figure 2, move $m$ is handled by YC $c$ and move $n$ handled by YC $c'$. Assume $c > c'$. As the slot difference of $m$ and $n$ is less than $SEP$, YC $c'$ cannot handle move $n$ immediately after YC $c$ finishes handling $m$. YC $c'$ has to wait for YC $c$ to gantry aside until the slot difference between YC $c$ and slot$_n$ is big enough. If $c - c' > 1$, e.g. move $m$ handled by $c3$ and $n$ handled by $c1$, there is one YC, $c2$ sits between $c1$ and $c3$. We assume that the safety distance between working/idle YC’s and working/idle YC’s is also $SEP$. The minimum slot number that YC $c$ has to gantry is $(c - c')SEP - |slot_m - slot_n|$ and the gantry time is $OT_{mn} = ((c - c')SEP - |slot_m - slot_n|) \times 0.06 + 0.09$ by taking into account the acceleration / deceleration stages of YC $c$. The feasible schedule of moves $m$ and $n$ is shown at the right of Figure 2. Note that constraint (6) is avoided in the discrete time model (Li et al. 2009) by leaving enough travel time at each time interval for YC to gantry to its next job, which could create a longer waiting time for the PM.

The minimum YC gantry time differs by the YC order and slot difference as follows:

$$OT_{mn} = \begin{cases} 
(\lvert c - c' \rvert SEP - \lvert slot_m - slot_n \rvert) \times 0.06 + 0.09 & \text{when } c > c' \cap slot_m > slot_n \text{ or } c < c' \cap slot_m < slot_n \\
(\lvert c - c' \rvert SEP + \lvert slot_m - slot_n \rvert) \times 0.06 + 0.09 & \text{when } c > c' \cap slot_m < slot_n \text{ or } c < c' \cap slot_m > slot_n 
\end{cases}$$

(7)

One of the four cases in (7) is shown in Figure 3 where move $m$ is handled by $c1$ and move $n$ by $c2$ ($c < c'$). Because move $n$ locates at the left of move $m$ ($slot_m > slot_n$), $c1$ will be pushed to the left of move $n$ ($m$ and $n$ still separated by $SEP$ slots) because $c1$ has to stay left of $c2$ all the time. In this case, the total slots $c1$ has to gantry will be $SEP + |slot_m - slot_n|$.
2.2.5 Moves handled by neighboring YC’s

For overlapping moves, if they are handled by neighboring YC’s, their slot difference should be greater than \(SEP\) multiplied by the YC order difference. Constraint (8) ensures this:

\[
W_{n,c'} \leq 3 - A_{2mn} - A_{2nm} - W_{mc}, \quad \forall c,c' \in C, \forall m,n \in PLJ_{mn}, \quad c' > c, \quad c < NC
\]

(8)

Constraint (8), valid for multi-crane scenarios, states that if moves \(m\) and \(n\) are overlapping in time, i.e., \(A_{2nm} = A_{2mn} = 1\), and move \(m\) is handled by YC \(c\), i.e., \(W_{mc} = 1\), then YC \(c'\) cannot handle moves which belong to \(PLJ_{mn}\). Set \(PLJ_{mn}\) is defined as

\[
PLJ_{mn} = \{m,n \in M : (\text{slot}_m - \text{slot}_n) < (c'-c) \times SEP \wedge m \neq n\}
\]

Hence, the moves that are placed to the left of move \(m\) and the moves that are placed to the right of move \(m\) but their slot difference against move \(m\) is less than \((c'-c) \times SEP\) belongs to \(PLJ_{mn}\) (Figure 4). Note that \(PLJ_{mn}\) varies by the order number of YCs. In Figure 4, YC 2 cannot handle moves that belong to \(PLJ_{mn}\) when YC 1 is handling move \(m\). If YC 3 is assigned to move \(n\), because there is an idle YC (YC 2) sitting in between YC 1 and 3, \(PLJ_{2mn}\) defined for YC 3 extends \(SEP\) slots left compared to \(PLJ_{mn}\). When moves \(m\) and \(n\) are far jobs (\(|\text{slot}_m - \text{slot}_n| \geq SEP\)) and move \(n\) is located to the left of move \(m\) handled by YC \(c\) (i.e. \(W_{mc} = 1, PLJ_{mn} = \text{true}\)), constraint (8) ensures that move \(n\) be handled by a YC whose order number is smaller than \(c\) when moves \(m\) and \(n\) overlap in time (\(A_{2mn} = A_{2nm} = 1\)). If \(PLJ_{mn} = \text{true}\) and \(|\text{slot}_m - \text{slot}_n| < SEP\), constraint (8) ensures that moves \(m\) and \(n\) do not overlap in time. In Figure 5, moves \(m1\) and \(m2\) overlap (at time axis) and they are far jobs because their slot difference is bigger than \(SEP\) (at slot
axis). As \( m_2 \) locates at the left of \( m_1 \), we have \( PLJ_{12} = \text{true} \). If we assume that YC 3 is handling \( m_1 \), constraint (8) then ensures that YCs whose order numbers are bigger than 3 (YC 4, YC 5 ...) cannot handle \( m_2 \). i.e., only YC 1 or YC 2 can handle \( m_2 \). On the contrary, moves \( m_3 \) and \( m_4 \) in Figure 5 are near jobs as \( |\text{slot}_3 - \text{slot}_4| < SEP \). In this case, we have \( PLJ_{34} = \text{true} \) and \( PLJ_{43} = \text{true} \). Applying constraint (8) to moves \( m_3 \) and \( m_4 \) respectively ensures that moves \( m_3 \) and \( m_4 \) cannot overlap in time. Constraint (8) is required by \( SEP \) apart of YC’s and inter-crane interference.

### 2.2.6 A YC can handle only one job at a time

\[
W_{mc} \leq 3 - A_{mn} - A_{nm} - W_{nc}, \quad \forall c \in C, \forall m, n \in M, NC > 1
\]  

Constraint (9) is only valid for multi-crane scenarios. It states that if moves \( m \) and \( n \) are overlapping, i.e., \( A_{mn} = A_{nm} = 1 \), and YC \( c \) is handling move \( m \), i.e., \( W_{mc} = 1 \), then YC \( c \) cannot handle move \( n \), i.e., \( W_{nc} = 0 \).

### 2.2.7 A storage move can take place no earlier than its target time i.e.

\[
T_{sm} \geq T_{gm}, \quad \forall m \in S_m
\]

### 2.2.8 Computing the retrieval earliness, \( RE_m \), of move \( m \)

\[
RE_m \geq T_{gm} - T_{sm}, \quad \forall m \in R_m
\]

### 2.2.9 Computing the retrieval lateness, \( RL_m \), of move \( m \)
2.2.10 Computing the storage lateness, \(STL_m\), of move \(m\)

\[
STL_m \geq Ts_m - Tgt_m, \quad \forall m \in R_m
\]

(11)

2.2.11 Moves located at the firstmost and rightmost slots

Moves located at the first \(SEP\) slots (Figure 4) can only be assigned to the first YC, due to inter-crane interference. Similarly, moves at the last \(SEP\) slots can only be assigned to the last YC, namely, \(W_{mc} = 1, \forall m \in LJ_m, c = 1; W_{mc} = 1, \forall m \in HJ_m, c = NC\).

Set \(LJ_m\) is defined as \(LJ_m = \{m \in M : slot_m \leq SEP\}\). \(LJ_m\) includes moves located at the first \(SEP\) slots (see Figure 4). Similarly, \(HJ_m\) is defined as \(HJ_m = \{m \in M : slot_m > (NSL - SEP)\}\) where \(NSL\) is the total number of slots. \(HJ_m\) includes moves located at the last \(SEP\) slots (see Figure 4).

To avoid YC interference, some YC’s cannot handle jobs located at the leftmost/rightmost slots. For instance, YC 2 cannot handle jobs in the first \(SEP\) slots as YC 1 sits on the left side of YC 2. YC 3 cannot handle jobs in the first 2*\(SEP\) slots as YC 1 and 2 sit on the left side of YC 3. Thus,

\[
W_{mc} = 0, \quad c > 1 \land slot_m \leq (c - 1) * SEP
\]

\[
W_{mc} = 0, \quad c < NC \land slot_m > (NSL - (NC - c)) * SEP
\]
Fixing part of $W_{mc}$ as above can reduce the solution time significantly.

### 2.3 Assumptions

The following assumptions are used for the continuous-time models.

- The target times and locations of container moves, found from the QC work list, are assumed known and fixed. Some studies (Chen and Lu 2012) have proposed approaches to optimize the storage locations of container moves. For retrieval jobs, the target time is the latest time by which a retrieval job can be handled and still meets the deadline set by the QCs. For storage jobs, the target time is the earliest time following the release of the job (i.e. the arrival of the corresponding PM) in the yard.

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

- As the job handling time of a YC is usually 2~4 minutes (Ng and Mak, 2005), we set the job handling time of all YC’s as 3 minutes (20 moves/hour) (i.e. $HT=3$).

- 20 to 60 moves in a two-hour time window are used in the scenarios tested in this paper. Each yard block contains 40-60 container slots (bays) (i.e. $NSL=40-60$).

- The minimum difference in slot numbers allowed for two working/idle YCs at the same time is assumed to be 8 slots (160 feet) (i.e. $SEP=8$).

- The YC acceleration/deceleration stages are taken into account when calculating its gantry time. The average YC gantry speed is assumed to be 100 m/min and the
acceleration/deceleration is 0.3 $ms^{-2}$.

### 2.4 Results of CMIP1

CMIP1 includes objective function (1), constraints (2), (4a)–(4b), (5)–(6), and (8)–(12). To test the performance of CMIP1, we adapted the test scenarios from Li et al. (2009) which include 30 test cases with 20–32 moves in 40-slot blocks and 30 test cases with 20–32 moves in 60-slot blocks. All test scenarios are randomly generated from a simulator developed for an actual large maritime transshipment terminal (Petering et al. 2009). ILOG CPLEX 11.2 is used to solve all the models developed on a Pentium 1.6GHz PC. The MILP relative optimality gap of CPLEX is set to 0 ($optcr = 0.0$). All other settings follow directly from the system default.

Scenario #1, which contains 2 YC's, 40 slots and 20 jobs, is used to illustrate the results. Table 1 contains the dataset for scenario #1. The start time of the scheduling horizon is zero. Using the formulations described in Sections 2.1–2.2 and applying the assumptions in Section 2.3, CMIP1 involves 420 binary variables, 495 single variables and 3,419 single equations. The model size is much smaller than the discrete time model (referred to as DMIP which involves 875 binary variables in 47 time intervals, 2,895 variables and 30,538 single equations) developed in Li et al. (2009). CMIP1 was solved in 0.3 seconds with a total PMWT of 4.28 minutes while DMIP was solved in 1.3 seconds with a total PMWT of 25.35 minutes. The PMWT of CMIP1 is 83% faster than that (25.35 minutes) obtained from DMIP. The results of scenario
#1 from DMIP and CMIP1 are shown in the left and right side of Figure 6, respectively. Each box in Figure 6 represents a move. The height and width of a box represents the handling time (HT) and the influencing slots (SEP) of a move respectively. If the actual scheduled start time of a move is different from the original target time, two boxes, a dotted and a solid one, are used to represent the original and the actual schedule, respectively. The handling YC number, job number, and the job type are shown within each solid box. At the right side of each solid box, a letter and a number show whether the actual schedule is delayed (“D”) or advanced (“E”) and the actual minutes delayed or advanced. As an example, though the target time of move 8 (the first move at the bottom) is 0.99 min, however, because the beginning of a move in a discrete time model should be synchronized to the boundaries of time intervals, move 8 is scheduled to start at 3.5 min which is the beginning of the next time interval. It is clear that the above procedure brings a 2.51 min delay to move 8. These delays or earliness resulting from axis synchronization occurs on many other moves in DMIP.

The delay on move 8 is completely eliminated in CMIP1 because a move can be handled exactly at its target time if there is no other conflict of operational constraints in a continuous time model. Scenario #1 is a relatively small example with moves evenly distributed; only 4.28 min of retrieval earliness occurs on move 15 from the results of CMIP1.

After testing the 60 randomly generated scenarios, the mean/maximum/minimum solution times of CMIP1 are 2,016.5/96,111.1/0.29 seconds respectively. We note that
the solution time of CMIP1 is still unreliable. Table 2 summarizes the comparison between the DMIP and CMIP1 using three performance measures. In Table 2, TCT is defined as the sum of the completion times of all moves ($TCT = \sum_{m\in M} (T_{s_m} + HT)$). The difference between DMIP and CMIP1 is defined as $(DMIP - CMIP1)/CMIP1 \times 100$. Table 2 shows that the average difference of PMWT and the objective value between DMIP and CMIP1 is very large. For example, for scenarios with storage and retrieval moves, the average PWMT of DMIP is 399.0% higher than CMIP1. We note that there is a significant difference between the three performance measures. In this paper, we will use CMIP1 as the benchmark for our models. The optimal solutions of all test scenarios are obtained by solving CMIP1.

Note that the average TCT gap is 1.0% by applying the lower bound evaluation procedure proposed by Ng (2005) for pure storage scenarios with 20–32 jobs. Since we obtain the optimal solutions for all these scenarios by solving CMIP1, the lower bound of TCT gap proposed by Ng (2005) is thus 1.0% below the optimal solution for scenarios with 20–32 jobs.

3. Algorithms integrated with the continuous time model

Though the number of binary variables is reduced significantly by only defining two bi-index binary variables, $W_{mc}$ and $A_{2mn}$, in CMIP1 compared to defining tri-index binary variables (see Ng 2005), the size of the model still increases significantly with the number of moves. To narrow the search space and reduce the solution time,
heuristics and algorithms are developed.

3.1 Heuristics

To narrow the search space for CMIP1, a heuristic (Li et al. 2009) is applied. The heuristics defines a job handling range \((T_{lo}, T_{up})\) for each move, where \(T_{lo}\) and \(T_{up}\) are the lower and upper bounds respectively. Since for both storage and retrieval moves, the greater the difference between the job finish time of a move and its target time, the larger the total \(PMWT\). Thus, the job finish time is scheduled around its corresponding target time as near as possible in an optimal solution. Only moves located in the same job handling range are carefully scheduled together. For moves whose target times are bigger than \(T_{up}\), e.g., move \(n1\) in Figure 7, we assume that it must be handled after move \(m\). For moves whose target times are smaller than \(T_{lo}\), e.g., move \(n3\) in Figure 7, we assume that it must be handled before move \(m\). Applying this heuristic can significantly reduce the model size because we can fix some binary variables. To apply the above heuristic to the continuous time model, we first define sets \(NM_{mn}\), \(NH_{mn}\) and \(NL_{mn}\) as follows:

\[
NM_{mn} = \{m,n \in M : |T_{gt}_n - T_{gt}_m| < JHR * HT \land m \neq n\} \\
NH_{mn} = \{m,n \in M : T_{gt}_n - T_{gt}_m \geq JHR * HT \land m \neq n\} \\
NL_{mn} = \{m,n \in M : T_{gt}_n - T_{gt}_m \leq JHR * HT \land m \neq n\}
\]

where \(JHR\) is a parameter used to define the job handling range. The value of \(JHR\) can be adjusted according to the tradeoff between the solution quality and time. The solution time decreases with the decrease of \(JHR\), but the solution quality may
deteriorate. \( JHR \) is set to 4.5 in this paper. Sets \( NM_{mn} \), \( NH_{mn} \) and \( NL_{mn} \) are defined to include moves \( n \) whose target times are placed inside/above/below the job handling range of move \( m \), respectively (Figure 7). Some constraints are modified to apply sets \( NM_{mn} \), \( NH_{mn} \) and \( NL_{mn} \). First, constraints (4a)–(4b) are defined within \( NM_{mn} \):

\[
\begin{align*}
TS_m + HT & \geq TS_n - BM*(1 - A_{2mn}), \quad \forall m, n \in NM_{mn}, \ m \neq n, \ NC > 1 \quad (4ah) \\
TS_n & \geq TS_m + HT - BM*A_{2mn}, \quad \forall m, n \in NM_{mn}, \ m \neq n, \ NC > 1 \quad (4bh)
\end{align*}
\]

Note that the only difference between (4a)–(4b) and (4ah)–(4bh) is that the condition \( \forall m, n \in M \) is replaced by \( \forall m, n \in NM_{mn} \). Since we assume that all moves \( n \) in \( NH_{mn} \) are handled after move \( m \) and all moves \( n \) in \( NL_{mn} \) are handled before move \( m \), part of \( A_{2mn} \) can be fixed i.e.:

\[
A_{2mn} = 0, \quad \forall m, n \in NH_{mn}; \quad A_{2mn} = 1, \quad \forall m, n \in NL_{mn}.
\]

Constraint (5) should be replaced by (5a)–(5c) as follows:

\[
\begin{align*}
TS_n & \geq TS_m + HT + GH_{mn} - [3 - W_{nc} - W_{nc'} - (1 - A_{2mn})]*BM, \quad \forall (m,n) \in NM_{mn}, \forall c \in C \quad (5a) \\
TS_n & \geq TS_m + HT + GH_{mn}, \quad \forall (m,n) \in NH_{mn}, \forall c \in C \quad (5b) \\
TS_n & \leq TS_m - HT - GH_{mn}, \quad \forall (m,n) \in NL_{mn}, \forall c \in C \quad (5c)
\end{align*}
\]

As constraint (5a) works only on moves within \( NM_{mn} \), two more constraints, (5b) and (5c), are added to ensure that moves within \( NH_{mn} \) are handled after move \( m \) and moves within \( NL_{mn} \) are handled before move \( m \). Similarly, constraint (6) should be replaced by (6a)–(6c):

\[
\begin{align*}
TS_n & \geq TS_m + HT + OT_{mn} - [3 - W_{nc} - W_{nc'} - (1 - A_{2mn})]*BM, \quad \forall (m,n) \in NM_{mn}, \forall c, c' \in C, \ m \neq n, \ c \neq c', \ NC > 1 \quad (6a) \\
TS_n & \geq TS_m + HT + OT_{mn}, \quad \forall (m,n) \in NH_{mn}, \forall c, c' \in C, \ m \neq n, \ c \neq c', \ NC > 1 \quad (6b) \\
TS_n & \leq TS_m - HT - OT_{mn}, \quad \forall (m,n) \in NL_{mn}, \forall c, c' \in C, \ m \neq n, \ c \neq c', \ NC > 1 \quad (6c)
\end{align*}
\]
Again, constraint (6) is replaced by three sets of constraints (6a)–(6c) working on moves within $NM_{mn}$, $NH_{mn}$, and $NL_{mn}$ respectively.

### 3.2 Rolling-horizon algorithm (RHA)

As we have observed in Section 3.1, the job finish time of a move should be scheduled as near as possible around its target time to minimize $PMWT$. As a result, a later move is likely to be handled after an earlier move, to reduce the gap between the job finish time and the target time. This gives us the opportunity to group moves by their target times. In this paper, we have extended Li et al.’s (2009) rolling-horizon algorithm (RHA) and applied RHA to our continuous time model. In RHA, moves are divided into groups in a rolling-horizon fashion (Figure 8). Moves with similar target times are grouped since their handling sequence may be interchanged. Moves in a higher group (moves with bigger job target times) will be handled later than a lower group (moves with smaller job target times). Thus, the problem can be divided into several sub-problems; each sub-problem involving moves of one group. As each sub-problem involves far fewer moves, it can be solved very quickly. As the handling sequence of moves in a neighbouring group which have big target times and moves in a neighbouring group which have small target times may also be interchanged, hence all sub-problems are generated on a rolling horizon basis. After a sub-problem is solved, two moves with the smallest handling times are fixed. In the following sub-problem, two new added moves together with the non-fixed moves from the previous sub-problem are scheduled. If several retrieval moves have very near target
times, it is possible to insert one or several of these moves into the empty space among the jobs fixed. Thus, constraints are added to handle the schedules between fixed and free moves. In RHA, set $PM_m$ is defined to include the free moves involved in a sub-problem. Then, we redefine sets $PLJ_{mn}$, $HJ_{mn}$, $LJ_{mn}$, $NM_{mn}$, $NH_{mn}$ and $NL_{mn}$ under set $PM_m$. For example, set $NM_{mn}$ is redefined as

$NM_{mn} = \{ m, n \in M : \vert Tgt_n - Tgt_s \vert < JHR \cdot HT \wedge m \neq n \wedge PM_m \wedge PM_n \}$. Note that only "$\wedge PM_m \wedge PM_n$" is added from the previous definition (Section 3.1). The MILP model for each sub-problem is defined as CMIP2. Appendix B lists the other constraints involved in CMIP2. In summary, the MILP model for RHA sub-problems (CMIP2) involves two types of constraints:

i). Constraints for the free moves in a sub-problem which include objective function (1), (s2), (4ah)–(4bh), (5a)–(5c), (6a)–(6c), (8), (s9), (s10)–(s12). All these constraints should be defined under set $PM_m$.

ii). Constraints for interactions between all previous fixed moves and current free moves in the sub-problem which include (dc4a1), (dc4b1), (dc4a2), (dc4b2), (dc5a), (dc5b), (dc6a), (dc6b), (dc8a), (dc8b), (dc9).

Figure 9 shows the flow diagram of the RHA for the continuous time model. We set the values of the parameters $JNPI (=7)$ and $JHR (=4.5)$. Set $PM_m$ includes moves involved in an iteration. At the start, $PM_m$ includes the first $JNPI$ moves with the smallest target times. CMIP2 is then solved to obtain the schedule for the first $JNPI$ moves. This takes less than 0.1 second to obtain an optimal solution as the size of
CMIP2 is very small at each iteration. The job finishing times and handling YCs of the two fixed moves in an iteration are recorded into sets $F_{xTsm}$ and $F_{xWmc}$ respectively. The algorithm stops when all moves in $SJ_m$ are scheduled.

### 3.3 Results of the continuous time model integrated with heuristics and RHA

The results of the continuous time model integrated with the heuristics and RHA (referred to as CRHA) are shown in Tables 3 and 4. Table 3 shows that CRHA obtains a near optimal solution in less than 2 seconds on average for scenarios with 20-30 jobs. CRHA is quite robust because the maximum solution time is only 6.0 seconds. For scenario#1 described in Section 2.4, CMIP2 involves 29 binary variables, 82 single variables and 344 single equations at the first iteration. CRHA obtains the optimal solution for scenario#1 in 0.64 second with 8 iterations. The average objective value gap for all scenarios is very small: 0.02% to 4.56% (20-30 jobs). A very low average TCT gap (0.0012%~0.24%) is obtained which is much smaller than the gap obtained in the literature (1.2%~1.9% in Ng 2005). Also, in Ng’s paper, a discrete-time MILP model was used as the benchmark for his heuristics model. As discrete-time MILP models usually yield solutions whose TCT gap can be 2~4% higher than the optimal solution (see Table 2), the actual TCT gap in his paper is higher than 1.2%~1.9%. CRHA obtains very good solution quality for pure storage scenarios, where the average objective value gap is only 0.02% and the average TCT gap is 0.0012%.
To test the performance of CRHA for large instances, we generate 28 test scenarios with moves ranging from 45 to 69. All test scenarios are randomly generated from a simulator developed for an actual large maritime transshipment terminal (Petering et al. 2009). For these large instances with 45 to 69 moves, the average solution time is about 5-12 seconds with a maximum solution time of 16.7 seconds (Table 4). CMIP1 yields no optimal solution for these large test scenarios due to the extremely long solution time required. By applying the lower bound evaluation procedure proposed by Ng (2005), the mean TCT gap for pure storage scenarios is 3.83%. This is smaller than the literature (7.3% in Ng 2005 and 5.2% in Li et al. 2009). Because $TCT_{LB}$ can be 1% lower than the optimal $TCT$ (see Section 2.4 for results from 20-32 jobs), the actual average $TCT$ gap for pure storage scenarios is less. Further, a difference in the $TCT$ gap usually corresponds to a much larger difference (about 10 times) in $PMWT$ and the objective gap, the actual $PMWT$ and objective gap obtained from this paper is much smaller than the literature. A key feature of CRHA is that it finds near optimal solutions for scenarios with pure storage moves in 1.1 seconds (20-30 jobs) and 6.2 seconds (50-60 jobs) on average.

Although the decrease in size is accompanied by an increase in the number of sub-problems to solve, the total solution time still reduces significantly. The results suggest that it is much better to solve a larger number of sub-problems than to solve the MILP model singly. As the number of jobs involved at each iteration is set to a small value, the model size of each iteration is very small. As the total problem size
increases, the total solution time only increases linearly in $n$, i.e. $\alpha \frac{n}{JFPI} e^{JNP1}$. With $JNP1$ much smaller than $n$, the exponential term, $e^{JNP1}$, is much smaller compared to $e^n$.

4. Conclusions

Aimed to generate CT applicable and optimal schedules, this paper develops an efficient continuous-time model and applies a rolling-horizon algorithm for container YC work schedules. The model size is reduced significantly and the solution time is shortened from days to seconds. The algorithm yields a much higher solution quality in a very short time, saving up to 10 minutes (for 20–30 moves scenarios) and 50 minutes (for 50–60 moves scenarios) of the total $PMWT$ compared to other existing heuristic models. A better performance measure is used and discussed in the paper. The proposed formulations and algorithms may be generally extended to problems with time windows and space constraints such as quay crane scheduling (Lim et al. 2004, Zhu et al. 2006) or berth allocation (Wang et al. 2007).

Last-minute container arrivals into the CT disrupt the normal CT operations and may impede CT performance. A CT has to handle these jobs and recover to normal CT operations as quickly as possible. Two initial methods in the rolling-horizon algorithm are discussed to schedule last-minute container arrivals as follows.

1): All moves are fixed except the newly arrived moves. New moves are allocated into the space among all previously fixed moves. As it may yield better solutions by
adjusting the handling time of previous scheduled moves, this method may only yield a near-optimal solution. As only the free jobs are new moves, the model size is very small and the solution time is thus very short.

2): Free all moves whose job handling times are larger than the arrival times of the new moves and solve the new problem. This method can find a better solution than 1) albeit a slightly longer solution time. Since we do not need to run iterations whose job handling times are smaller than the new arrival times, the solution time is shorter than solving CMIP2.

In scenario #1, a last-minute container needs to be stored at time 108, slot 34 (referred to as move \textit{m21}). Using approach 1) above, the \textit{PMWT} yields 14.634 minutes and the solution time is only 0.03 second. The schedule is shown in the left side of Figure 10. As the handling times of moves \textit{m18} to \textit{m20} are fixed, the newly inserted storage move, \textit{m21}, is scheduled to be handled immediately after move \textit{m20}. This introduces a rather large storage delay (10.353 minutes). By applying approach 2), moves \textit{m18} to \textit{m20} are freed and scheduled together with move \textit{m21}. Different from approach 1), \textit{m21} is inserted between \textit{m18} and \textit{m19}. The total \textit{PMWT} is 8.634 minutes, which is also optimal (confirmed by solving \textit{CMIP1}). The schedule is shown on the right of Figure 10. The solution time is 0.06 second.

Future research will further investigate approaches to last-minute jobs handling and look into developing robust YC scheduling models with uncertain PM arrival times.
Appendix A: Definition of Sets and Parameters

(a) Indices

\(c, c' = \) yard cranes, \(c, c' = 1, 2, \ldots, C\)

\(m, n = \) container moves, \(m, n = 1, 2, \ldots, M\)

\(SL = \) slot number, \(SL = 1, 2, NSL\)

(b) Sets

\(C = \) number of YC’s working in the CY.

\(FxJ_m = \) jobs that have been fixed after the current iteration and all previous iterations

\(HJ_m, LJ_m = \) includes moves located at the last, first SEP slots in CY respectively

\(M = \) set of container moves to be scheduled

\(NL_{mn}, NM_{mn}, NH_{mn} = \) moves \(n\) whose target times are placed below, inside, above job handling range of move \(m\) respectively

\(SJ_m = \) all jobs sorted by their target times

\(PLJ_{mn} = \) moves that locate on the left side of move \(m\) and moves that locate at the right side \(PM_m = \) moves involved in a sub-problem of rolling-horizon algorithm

\(R_m, S_m = \) set of yard retrieval, storage moves (the total number of items in \(R_m\) and \(S_m\) is \(M\))

\(TSL = \) number of slots in the CY

(c) Parameters
\(BM\) = a large number, \(BM = Max(Tgt_m) + 50\)

\(FxTsm\) = Job start time \((Tsm)\) of moves that have been fixed

\(FxW_{mc}\) = \(W_{mc}\) of moves that have been fixed

\(GH_{mn}\) = time for a YC to gantry from \(slot_m\) to \(slot_n\) when consecutive moves \(m\) and \(n\) are handled by the same YC

\(HT\) = YC handling time of a container move; 3 minutes is used in this paper

\(JHR\) = parameter used to defined the job handling range

\(JNPI\) = job number per iteration

\(NC\) = total number of YC’s scheduled

\(NSL\) = total number of slots in the CY

\(OT_{mn}\) = time for YC to gantry aside when consecutive nearby moves \(m\) and \(n\) are handled by different YC’s

\(SEP\) = minimum difference in slot numbers allowed for two YC’s at the same time.

\(SEP\) is assumed to be 8 slots in this paper

\(Slot_m\) = Slot number where move \(m\) takes place. Without loss of generality, we assume that if \(m < n\), then \(slot_m \leq slot_n\)

\(Tgt_m\) = target time for move \(m\) in the CY. For retrieval moves, this is the latest job start time that meets the deadline set by the QCs. For storage moves, this is the earliest job start time following the release of the job in the CY

\(W_{res}, W_{rd}\) = weight assigned to total retrieval earliness, total retrieval delay in the objective function respectively

\(W_{sd}\) = weight assigned to total storage delay in the objective function
d) Variables

\( A_{1mn}, A_{2mn} = 0-1 \) variable to denote if move \( n \) happens before, after the start of move \( m \) respectively

\( PMWT = \text{total PM waiting time}, \quad PMWT = \sum_{m \in R} RE_m + \sum_{m \in R} RL_m + \sum_{m \in S} STL_m \)

\( RE_m, RL_m = \text{amount of retrieval earliness, retrieval lateness for move } m \) respectively

\( STL_m = \text{amount of storage lateness for move } m \)

\( Ts_m = \text{job start time of container move } m \)

\( W_{mc} = 0-1, \text{ binary variable to denote if container move } m \text{ is assigned to YC } c \)

\( TC = \text{linear combination of the total amount of retrieval earliness, and storage and retrieval delays.} \)

\( Y_{mn} = 0-1, \text{ continuous variable to denote whether moves } m \text{ and } n \text{ are overlapping} \)

Appendix B: Constraints used in CMIP2

In CMIP2, the following constraints are modified under set \( PM_m \):

Constraint (2) is replaced by (s2):

\[ \sum_{c \in C} W_{mc} = 1, \quad \forall m \in PM_m, NC > 1 \] (s2)

For single YC scenario, we fix \( W_{mc} = 1, \quad \forall m \in PM_m, NC = 1 \)

Constraint (9) is replaced by (s9):

\[ W_{nc} \leq 3 - A_{2mn} - A_{2nm} - W_{mc}, \quad \forall c \in C, \forall m \in PM_m, \forall n \in PM_n, NC > 1 \] (s9)

Constraints (10)-(12) are replaced by (s10)-(s12) respectively:

\[ RE_m \geq T_{gm} - Ts_m, \quad \forall m \in R_m \cap PM_m \] (s10)
To allow unscheduled jobs to be inserted among the fixed jobs, the following constraints are necessary:

\[
RL_m \geq Ts_m - Tgt_m, \quad \forall m \in R_m \cap PM_m
\]  
\[
STL_m \geq Ts_m - Tgt_m, \quad \forall m \in S_m \cap PM_m
\]  

Constraints (dc4a1), (dc4a2), (dc4b1), (dc4b2) are the counterparts of constraints (4a) and (4b) respectively. Note that moves \(m\) and \(n\) in (dc4a1) and (dc4b1) are defined under sets \(PM_m\) and \(FxJ_n\) separately and used to determine the value of \(A2_{mn}\) only. Thus, (dc4a2) and (dc4b2) are added to determine the value of \(A2_{nm}\).
complete schedule. For example, if YC 2 has been scheduled to handle move \(n4\) (this
decision has been fixed because \(n4\) belongs to \(FxJ_n\)) and we want YC 2 to handle \(m6\)
after handling \(n4\). The least YC gantry time between \(n4\) and \(m6\) has to be respected
(see Section 2.2.3 for details). Another important situation is that, if moves in \(PM_m\) are
very crowded (e.g. moves \(m5\) to \(m11\) in Figure B.1), it might be better to insert some
of these moves in the free space of fixed moves, e.g., we could possibly insert move
\(m5\) after \(n1\) and before \(n3\) because the YC’s are free during that period. If this is the
case, then operations constraints between \(m5\) and the fixed moves \(n1\) and \(n3\) should
be respected. The following constraints define the operating constraints between free
moves and fixed moves. They are counterparts of constraints (5) to (9) in Section 2.2
respectively. Constraints (dc5a) to (dc9) differ from (5) to (9) by defining under sets
\(PM_m\) and \(FxJ_n\) instead. They are added for the completeness of CMIP2.

\[
T_{s_n} \geq FxT_{s_n} + HT + GH_{mn} - [3 - W_{mc} - FxW_{mc} - (1 - A_{2nm})]BM,
\forall m \in PM_m, \forall n \in FxJ_n, \forall c \in C
\]  
\text{(dc5a)}

\[
FxT_{s_n} \geq T_{s_n} + HT + GH_{mn} - [3 - W_{mc} - FxW_{mc} - (1 - A_{2nm})]BM,
\forall m \in PM_m, \forall n \in FxJ_n, \forall c \in C
\]  
\text{(dc5b)}

\[
FxT_{s_n} \geq T_{s_n} + HT + OT_{mn}
- [3 - W_{mc} - FxW_{mc} - (1 - A_{2nm})]BM,
\forall c, c' \in C, c \neq c', NC > 1, m \neq n, \forall m \in PM_m, \forall n \in FxJ_n
\]  
\text{(dc6a)}

\[
T_{s_n} \geq FxT_{s_m} + HT + OT_{mn}
- [3 - FxW_{mc} - W_{mc} - (1 - A_{2nm})]BM,
\forall c, c' \in C, c \neq c', NC > 1, m \neq n, \forall m \in PM_m, \forall n \in FxJ_m
\]  
\text{(dc6b)}

\[
W_{nc} \leq 3 - A_{2nm} - A_{2nm} - FxW_{mc}, \forall c, c' \in C, c' > c, c < NC,
(slot_n - slot_m) < (c' - c)*SEP, m \neq n, \forall n \in PM_m, \forall m \in FxJ_m,
\]  
\text{(dc8a)}

\[
FxW_{nc} \leq 3 - A_{2nm} - A_{2nm} - W_{mc}, \forall c, c' \in C, c' > c, c < NC
(slot_n - slot_m) < (c' - c)*SEP, m \neq n, \forall m \in PM_m, \forall n \in FxJ_n,
\]  
\text{(dc8b)}
\[ W_{n_c} \leq 3 - A_{nn} - A_{nm} - F_{x}W_{n_c}, \]
\[ \forall c \in C, \forall n \in PM_{n}, \forall m \in F_{x}J_{m}, \ NC > 1 \]

(de9)
References


transshipment terminals. *OR Spectrum*, 31(4), 801-836


**List of Figure Captions:**

Figure 1 Overlapping and non-overlapping container moves

Figure 2 Least gantry time between consecutive moves at different YCs

Figure 3 YC minimum gantry slots

Figure 4 Definitions of PLJ, HJ and LJ

Figure 5 Illustration of constraint (8)

Figure 6 Quality comparison of Scenario #1 between DMIP and CMIP1

Figure 7 Definition of neighboring moves

Figure 8 Definition of rolling sub-problems

Figure 9 Rolling-horizon algorithm

Figure 10 Results of the two approaches handling last minute job insertions

Figure B.1 Interactions between free and fixed moves in CRHA

**List of Table Captions:**

Table 1 Data for scenario #1

Table 2 Quality comparison between DMIP and CMIP1

Table 3 Performance (20-30 moves) of CRHA

Table 4 Performance (45-69 moves) of CRHA
Table 1
Data for Scenario #1

<table>
<thead>
<tr>
<th>Moves</th>
<th>Target time (min)</th>
<th>slot_m</th>
<th>Type*</th>
<th>moves</th>
<th>Target time (min)</th>
<th>slot_m</th>
<th>Type*</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>49.962</td>
<td>1</td>
<td>R</td>
<td>m11</td>
<td>86.49</td>
<td>24</td>
<td>R</td>
</tr>
<tr>
<td>m2</td>
<td>52.962</td>
<td>1</td>
<td>R</td>
<td>m12</td>
<td>89.49</td>
<td>24</td>
<td>R</td>
</tr>
<tr>
<td>m3</td>
<td>55.962</td>
<td>1</td>
<td>R</td>
<td>m13</td>
<td>72.336</td>
<td>28</td>
<td>S</td>
</tr>
<tr>
<td>m4</td>
<td>34.602</td>
<td>8</td>
<td>S</td>
<td>m14</td>
<td>101.142</td>
<td>30</td>
<td>S</td>
</tr>
<tr>
<td>m5</td>
<td>15.806</td>
<td>14</td>
<td>R</td>
<td>m15</td>
<td>38.433</td>
<td>34</td>
<td>R</td>
</tr>
<tr>
<td>m6</td>
<td>18.806</td>
<td>14</td>
<td>R</td>
<td>m16</td>
<td>41.433</td>
<td>34</td>
<td>R</td>
</tr>
<tr>
<td>m7</td>
<td>21.806</td>
<td>14</td>
<td>R</td>
<td>m17</td>
<td>37.302</td>
<td>35</td>
<td>S</td>
</tr>
<tr>
<td>m8</td>
<td>0.992</td>
<td>18</td>
<td>S</td>
<td>m18</td>
<td>109.023</td>
<td>38</td>
<td>R</td>
</tr>
<tr>
<td>m9</td>
<td>78.323</td>
<td>22</td>
<td>S</td>
<td>m19</td>
<td>112.023</td>
<td>38</td>
<td>R</td>
</tr>
<tr>
<td>m10</td>
<td>83.49</td>
<td>24</td>
<td>R</td>
<td>m20</td>
<td>115.023</td>
<td>38</td>
<td>R</td>
</tr>
</tbody>
</table>

*R: retrieval move; S: storage move
Table 2
Quality comparison between *DMIP* and *CMIP1*

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Average Gap, %</th>
<th>PMWT*</th>
<th>TCT**</th>
<th>Objective Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>40 slots</td>
<td></td>
<td>519.9</td>
<td>2.5</td>
<td>551.3</td>
</tr>
<tr>
<td>60 slots</td>
<td></td>
<td>366.0</td>
<td>3.1</td>
<td>382.3</td>
</tr>
<tr>
<td>Pure storage moves</td>
<td></td>
<td>508.9</td>
<td>4.3</td>
<td>508.9</td>
</tr>
<tr>
<td>Storage + Retrieval</td>
<td></td>
<td>399.0</td>
<td>1.8</td>
<td>438.7</td>
</tr>
</tbody>
</table>

*PMWT*: Total PM Waiting Time; **TCT**: Total Completion Time
Table 3
Performance (20-30 moves) of CRHA

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Solution Time, seconds</th>
<th>Average Gap, %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Maximum</td>
</tr>
<tr>
<td>40 slots</td>
<td>1.2</td>
<td>3.8</td>
</tr>
<tr>
<td>60 slots</td>
<td>1.6</td>
<td>6.0</td>
</tr>
<tr>
<td>Pure storage</td>
<td>1.1</td>
<td>1.8</td>
</tr>
<tr>
<td>Storage+Retrieval</td>
<td>1.6</td>
<td>6.0</td>
</tr>
</tbody>
</table>
Table 4
Performance (45-69 moves) of CRHA

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Solution Time, seconds</th>
<th>((TCT-TCT_{LB})*/TCT_{LB}*100)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Maximum</td>
</tr>
<tr>
<td>40 slots</td>
<td>5.5</td>
<td>15.8</td>
</tr>
<tr>
<td>60 slots</td>
<td>9.1</td>
<td>16.7</td>
</tr>
<tr>
<td>Pure storage moves</td>
<td>6.2</td>
<td>12.6</td>
</tr>
<tr>
<td>Storage+Retrieval</td>
<td>11.4</td>
<td>16.7</td>
</tr>
</tbody>
</table>

* \(TCT_{LB}\): lower bound of \(TCT\) obtained by applying the approach of (Ng 2005)
Fig. 1 Overlapping and non-overlapping container moves
Fig. 2 Least gantry time between consecutive moves at different YC's
Fig. 3 YC minimum gantry slots

Before c1 gantry

After c1 gantry

c1 new position

c1 original position

c1 gantry direction: left

Time

OT_{mn}

SEP

Slot_{m}

Slot_{n}

m, c1

n, c2
Fig. 4. Definitions of PLJ, HJ and LJ.
Fig. 5 Illustration of constraint (8)
Fig. 6. Quality comparison of Scenario #1 between DMIP and CMIP1
Fig. 7 Definition of neighboring moves

Range of job handling time of move $m$
Fig. 8 Definition of rolling sub-problems (from Li et al. 2009)
Read *include* files
Set JNP1, JHR
Set $F_{xJm} = 0; F_{xTsm} = 0; F_{xWmc} = 0$
Define $SJ_m, PM_m$

Update sets: $PL_{mn}, HJ_{mn}, L_{mn}, NL_{mn}, NM_{mn}, NH_{mn}$
Fix variables: $W_{mc}, A_{2mn}$

Solve CMIP2

- Identify two jobs with smallest and the next smallest finish time. Fix them.
- Add the two fixed jobs into $FxJ_m$.
- Set $FxTsm$ and $FxWmc$.
- Record $RE, RL, STL$ and handling $YC$ of the two fixed jobs.

Are there any jobs left in $SJ_m$ that need to be scheduled?

NO

END

YES

- Update $PM_m$
Delete the two fixed jobs from $PM_m$ and add two new jobs with the smallest and least smallest target time from $SJ_m$.

Fig. 9 Rolling-horizon algorithm
Fig. 10 Results of the two approaches handling last minute job insertions
Fig. B.1 Interactions between free and fixed moves in CRHA