Two Yard Crane Scheduling With Dynamic Processing Time and Interference

Feifeng Zheng, Xiaoyi Man, Feng Chu, *Senior Member, IEEE*, Ming Liu, *Senior Member, IEEE*, and Chengbin Chu

***Abstract*— Maritime transportation is an important branch of intelligent transportation system (ITS). It is widely recognized that modern ITS technologies and advanced management methods, such as automated yard crane (YC) planning and scheduling, can significantly improve container terminal performance, and also impact the global performance of maritime transportation. In this paper, we investigate two YC scheduling with storage and retrieval tasks in a container block. The main contributions of this paper are: 1) container reshuffling operations and inter-crane interference constraint are both considered and 2) the dynamic processing times for retrieval containers are taken into consideration. These typical YC operation characteristics complicate the YC scheduling, and cause late delivery and economic loss. In this study, we focus on minimizing the maximum tardiness of container task and establishing an integer linear programming model. Regarding the NP-hardness nature of the problem, we develop a heuristic named dividing, sequencing, and comparing (DSC) and a genetic algorithm (GA) based on the characteristics of the problem. The computational results show the efficient performance of the developed algorithms, compared with the exact solutions via Cplex software for small size instances. The efficiency and effectiveness of DSC outperform those of GA for practical size instances.**

***Index Terms*— Maritime logistics, yard crane scheduling, reshuffling, interference, dynamic processing time, heuristic.**

I. Introduction

IN RECENT decades, with the boom in maritime trade, there is a rapid growth of research on the major resource planning and scheduling in container terminals, such as berth allocation, quay crane (QC) assignment, and integrated scheduling. Especially, on the land terminal, yard cranes (YCs) are becoming a critical resource and the bottleneck during the processing of loading/unloading containers that are temporarily stored in the blocks of the yard. Consequently, the optimization of yard crane scheduling performs an important role in the development of automated container terminal.

In the container yard, there are three typical kinds of container blocks, i.e., *import block*, *export block* and *transshipment block*. Each block is dedicated to the temporary storage and transfer the corresponding containers. In an import block, there are exclusively import containers. An import block receives cargoes from oversea, and these containers are transported via yard trucks from vessels to the yard. Later on they are delivered to domestic customers by customer trucks. In an export block, export containers are sent to the yard by domestic customer trucks, and later on they are loaded onto vessels for oversea destinations. In a transshipment block, the containers from some original foreign countries are unloaded from vessels, and they will be loaded again onto other vessels for their oversea destinations.

In a block, the container loading/unloading operations are mainly handled via YCs. It causes *inter-crane interference* between YCs if they are too close to each other, resulting in a negative impact on their processing efficiency and operation safety. So there is a safety distance requirement between YCs to avoid interference. Moreover, due to the space limitation within a block, there are usually no more than two YCs simultaneously serving container tasks in a block in practice.

There exist two kinds of container tasks in the above three types of blocks. We refer to the tasks for moving containers from yard or customer trucks to the container stacks in a block as *storage tasks*, and refer to those for moving containers from container stacks to yard or customer trucks as *retrieval tasks*. The corresponding containers are respectively called *storage containers* and *retrieval containers*. According to the practical operation of the container terminal, a *storage container* is directly placed on the top of one container stack (Kim and Hong [1], Lee *et al.* [2], Lee *et al.* [3], Guo and Huang [4], and Huang *et al.* [5]). For a retrieval operation, if retrieval container is beneath some other containers, then these containers, called *reshuffle containers*, have to be moved away such that the retrieval container can be moved onto the truck. Reshuffle container move is called reshuffle.

Manuscript received February 13, 2017; revised August 22, 2017 and November 22, 2017; accepted December 2, 2017. This work was supported in part by the National Science Foundation of China under Grant 71771048, Grant 71531011, Grant 71571061, Grant 71571134, and Grant 71428002, in part by the DHU Distinguished Young Professor Program under Grant A201305, in part by the Fundamental Research Funds for Central Universities, and in part by the Shanghai Pujiang Program. The Associate Editor for this paper was A. Che. *(Corresponding author: Ming Liu.)*

F. Zheng and X. Man are with the Glorious Sun School of Business and Management, Donghua University, Shanghai 200051, China (e-mail: ffzheng@dhu.edu.cn; manxiaoyi8996@163.com).

F. Chu is with IBISC, University of Évry Val d'Essonne, University of Paris-Saclay, 91025 Évry, France, and also with the Management Engineering Research Center, Xihua University, Chengdu 610039, China (e-mail: feng.chu@univ-evry.fr).

M. Liu is with School of Economics and Management, Tongji University, Shanghai 200092, China (e-mail: mingliu@tongji.edu.cn).

C. Chu is with ESIEE Paris, Université Paris-Est, 93162 Noisy-le-Grand, France (e-mail: chengbin.chu@cep.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2017.2780256

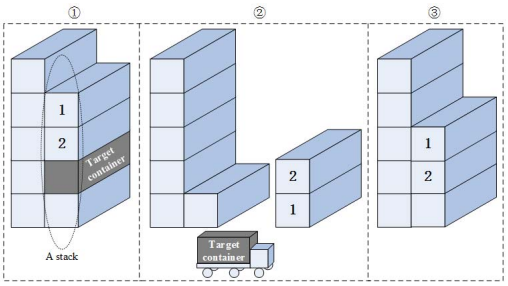


Fig. 1. An internal reshuffling operation.

There are usually two types of reshuffles: *external reshuffles* and *internal reshuffles*. The *external reshuffles* indicate that the reshuffle containers above *retrieval ones* in a stack are removed firstly and then the same number of containers, that may not be the previously removed ones, are reloaded onto the stack (Kim and Hong [1] and Meisel and Wichmann [6]). The *internal reshuffles* mean to reposition the reshuffle containers in the same stack. In practice, the *internal reshuffles* are widely used in the container yard planning and scheduling to facility its management [1]. Another possibility in reshuffling operations is to not put back reshuffle containers to their original stacks. In this case, it is necessary to update global state of the block in computer after each reshuffling operations. This will undoubtedly complicate the container terminal management.

Figure 1 illustrates an internal reshuffling operation to process a retrieval container. Containers 1 and 2 are stacked above the target container. They are moved aside before loading the target container onto the truck, and then moved back to the original stack immediately after handling the retrieval container. Thus, a retrieval task includes the target container and reshuffle container moves. Hence, the processing time of a retrieval task varies with reshuffle container moves. Given that there are more than one task at the same stack, the processing time of a task varies with processing sequence, and it shows dynamic processing time of a task.

In the daily operations of the container terminal, the container yard plays a significant role in connecting the seaside and the landside. That is to say, the accuracy of the YC scheduling has an important impact on both the vessel turnaround time and the customer satisfaction in practice. On one hand, the huge amount of loading/unloading tasks from the vessel puts forwards high requirements to YC. On the other hand, when the external trucks come to the container terminal for delivery or retrieve, it is necessary to reduce the truck waiting time to improve the customer satisfaction. Therefore, the accurate task processing time shall be considered in the scheduling of YC. To achieve this, it is a necessity to detail the processing time of a retrieval task with reshuffling which may be much longer than that of a storage task. The consequent YC schedule will output accurate loading/unloading container time points, and the related vessel turnaround time can be well estimated. Furthermore, in any situation, the minimum safety distance between YCs has to be respected to avoid inter-crane interference at all times when there are more than one YC at the same block.

In this work, we study the two YCs scheduling in a block with the objective of minimizing the maximum tardiness of tasks. The contributions of the paper include: (1) the practical constraints including container reshuffling operations and inter-crane interference are simultaneously taken into consideration; (2) the dynamic processing times for retrieval tasks are calculated; (3) a DSC heuristic and a genetic algorithm based on the characteristics of the problem are developed. The rest of this paper is organized as follows. Section II gives a brief literature review. In Section III, we describe the concerned problem and present a mathematical formulation. In Section IV, we develop a DSC heuristic and a genetic algorithm. Numerical experiments are conducted in Section V. Finally, Section VI concludes the work and indicates future research directions.

II. Literature Review

With the rapid expansion of the world trade, integrated scheduling of quay cranes and berthing operations as an important branch of maritime terminal management has been investigated by many researchers (Liu *et al.* [7], Goodchild and Daganzo [8], Kang *et al.* [9], Nguyen and Kim [10], Carlo *et al.* [11], Zhen *et al.* [12], Liu *et al.* [13], and Cichenski *et al.* [14]). Meanwhile, due to the restrictions of equipment and shoreline resources on the seaside, both the efficiency of the yard crane management and its impact on maritime container terminals have also attracted more attention of practitioners and researchers.

According to the number of yard cranes in one block, we divide the literature into two categories: (1) a single yard crane scheduling, and (2) multiple yard cranes scheduling.

A major body of research focuses on a single yard crane scheduling. Kim and Kim [15] investigated a single yard crane routing problem to support the loading operation of a vessel. A mixed integer programming model was proposed to determine the number of container to be loaded onto the vessel at each bay and the sequence of bays to be visited by yard crane. Then an efficient algorithm was developed. Narasimhan and Palekar [16] studied the same problem. They proved the problem was NP-hard and developed a branch-and-bound algorithm. Ng and Mak [17] investigated a yard crane scheduling to perform handling tasks with release time. An exact branch-and-bound algorithm was proposed. For the same problem, Guo *et al.* [18] proposed two new algorithms with considering First Come First Serve (FCFS) rule, Nearest Job First (NJF) rule, and Smallest Completion Time Job First (SCJF) rule to efficiently compute yard crane dispatching sequences. For all above publications, the processing times of all tasks were considered identical, and reshuffles were not considered. According to Gharehgozli *et al.* [19], dynamic processing time for retrieval container caused by reshuffle was not well studied. Only Huang *et al.* [20] investigated a single yard crane scheduling problem with dynamic processing time. However, the reshuffling operations of reshuffle containers were not considered. Then an embedded simulation was applied to obtain the sequence of yard crane. Numerical experiments with only ten tasks were presented.

For multiple yard cranes scheduling, most of researchers have focused on two yard cranes scheduling. Furthermore, there is no research on reshuffle for multiple yard cranes scheduling. Hence, according to the characteristics of multiple yard crane scheduling, we distinguish existing work into two types: (1) without interference and reshuffle; (2) with interference but without reshuffle.

For the first type, Cao *et al.* [21] first introduced two yard cranes scheduling for the port of Hamburg to minimize the loading time of the containers. They assumed the two yard cranes with different height and width could pass through each other. Moreover, the loading time of each container was assumed to be identical. An heuristic was developed to solve the problem. Instead of only considering loading container by [21] and Vis and Carlo [22] extended two yard cranes scheduling considering process storage and retrieval containers simultaneously. A mixed integer programming model was proposed to minimize the makespan and a simulated-annealing based heuristic method was developed.

Most of the researches focus on two yard crane scheduling with interference. Ng [23] first investigated the problem with interference constraint within a generic yard block. An integer programming model and a dynamic programming-based heuristic were proposed for it. Although inter-crane interference was considered, the safety distance requirement has not been considered and the storage and retrieval tasks have not been distinguished. Li *et al.* [24] studied two yard crane scheduling with safety distances and storage/retrieval task distinction. A discrete time model and a rolling horizon algorithm based on Earliest Due Date (EDD) rule were devised to solve the problem with the objective of minimizing earliness and delays. Li *et al.* [25] further improved their model by developing an efficient continuous-time mixed linear programming model, and the solution quality and speed of the former model were greatly improved. Gharehgozli *et al.* [26] extended the two yard cranes scheduling problem with considering precedence constraints. A new continuous time model and an adaptive large neighborhood search heuristic were developed to solve the problem. Wu *et al.* [27] addressed two yard cranes scheduling with objective of minimizing the deviation between the actual processing time and the ideal processing time. They devised a clustering reassigning approach. Ehleiter and Jaehn [28] addressed the container repositioning problem of scheduling two yard cranes with an initial schedule in a block. A heuristic based dynamic programming approach and a greedy heuristic were devised to solve the problem. Other related studies may include Zhang *et al.* [29], Cheung *et al.* [30], Bazzazi *et al.* [31], Angeloudis and Bell [32], and Saini *et al.* [33]. In brief, most of the current literature only focus on two yard crane scheduling with interference. However, the reshuffle and dynamic processing time of the task caused by reshuffle have not been considered.

Summing up, to the best of our knowledge: (1) interference and reshuffle have not been considered simultaneously in existing work ; (2) dynamic processing time has not been studied in multiple yard crane scheduling (see Table I). To reduce this gap between theoretic method and applications, we investigate two yard crane scheduling considering (1) interference and reshuffle simultaneously; (2) dynamic processing time and interference between yard cranes.

TABLE I
COMPARISON ON STUDIES OF THE SCHEDULING PROBLEM OF YC

Literature	YC	Interference	Reshuffle	Processing time	Approach
Huang et al. [16]	one	no	no	dynamic	greedy
Cao et al. [17]	two	no	no	constant	greedy, simulated annealing
Ng [19]	two	yes	no	constant	dynamic programming
Li et al. [20,21]	two	yes	no	constant	rolling-horizon
Gharehgozli et al. [22]	two	yes	no	constant	neighborhood search
Wu et al. [23]	two	yes	no	constant	clustering reassigning
This paper	**two**	**yes**	**yes**	**dynamic**	**DSC, GA**

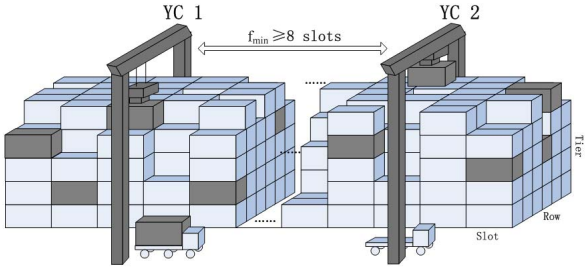


Fig. 2. Illustration of a block with two YCs.

III. PROBLEM STATEMENT AND MATHEMATICAL FORMULATION

A. Problem Statement

There are two identical yard cranes YC1 and YC2 to process a set of storage and retrieval tasks in a container block, as shown in Figure 2. The block consists of at most $x_{\max}$ container slots, $y_{\max}$ rows in each slot, and $z_{\max}$ tiers in each row where the values of $z_{\max}$, $y_{\max}$ and $x_{\max}$ are positive integer constants. Each YC moves along the slot dimension with a constant speed. As the length in slot dimension is much longer than that in row or tier dimension, the moving time of a YC between any two consecutive tasks is determined by the slot distance of the two tasks in this study. This is consistent with most researches (i.e., Kim and Kim [15], Ng and Mak [17], Guo *et al.* [18], Huang *et al.* [20], etc). Each YC occupies a single slot in processing any task, and its slot location changes with the slot location of tasks to be processed. Without loss of generality, the original locations of the two YCs are at the two ends of the block, and YC1 is always to the left of YC2 in the whole time horizon.

Each task i is associated with release time r_i, due date d_i, and a given 3-dimensional container location (x_i, y_i, z_i), representing respectively the indices of slot, row and tier where $1 \leq z_i \leq z_{max}, 1 \leq y_i \leq y_{max}$ and $1 \leq x_i \leq x_{max}$. For example, given the location (7, 5, 1) of task i, it means the task is located on the 1st tier (the lowest level) of the 5th row in the 7th slot. The number of containers in each stack may change over time with storage and retrieval container moves.

The processing time of any storage task is considered as constant and means the duration of moving a target container from a yard or customer truck to the top of the destination stack, i.e., the time of a single container move. If a retrieval task is not laid on the highest tier of the stack, reshuffling operations occur. The processing time of the retrieval task is the total time of moving the target container and related reshuffle containers, i.e., the time of $(1+2u)$ container moves where u is the number of reshuffle containers. In addition, YC first processes the retrieval task, and then storage task when both storage and retrieval tasks are planned in the same stack. This is adaptable in theoretical (Gharehgozli *et al.* [19]) and practical (Yangshan Deep-water Port, Shanghai) settings.

The problem consists of determining the two YCs orders for planned tasks to minimize the maximum tardiness of all tasks. The tardiness of each task is calculated by $\psi_i = \max\{0, \delta_i - d_i\}$ where δ_i is the completion time of task i. Due to the inter-crane interference constraint, the safety distance between YC1 and YC2 for any instance is set to be $f_{\min}$. For example, $f_{\min} = 8$ means that the safety distance between YC1 and YC2 must be at least 8 slots. Especially, if YC1 is processing a task at slot 8 of the block, then YC2 can not handle any containers in slots from 9 to 15 (Li *et al.* [25]).

B. Mathematical Formulation

Below we propose a mixed integer linear programming (MILP) model for the problem. We first give basic notation and decision variables, and then present the objective function and relevant constraints.

1) Basic Notations:

a) Indices:

i, j : indices of storage or retrieval tasks (containers);
x, x' : indices of slots;
t, t' : indices of time steps;
k : index of yard cranes, and $k \in \{1, 2\}$.

b) Input parameters:

N : set of tasks, and $N = \{1, 2, \ldots, |N|\}$;
L : set of slots, and $L = \{1, 2, \ldots, |L|\}$;
T : set of time steps, and $T = \{1, 2, \ldots, |T|\}$;
N^R : set of retrieval tasks, $N^R \subset N$;
$f_{\min}$: the minimum safety distance between the two YCs;
q_i : the number of containers in the stack of task i, where $i \in N$;
x_i : slot position of task i, where $i \in N, 1 \le x_i \le x_{\max}$;
y_i : row position of task i, where $i \in N, 1 \le y_i \le y_{\max}$;
z_i : tier position of task i, where $i \in N, 1 \le z_i \le z_{\max}$;
u_i : the number of reshuffle containers that are piled above the retrieval task i, and $u_i = |q_i - z_i|$, where $i \in N^R$;
G_j : set of tasks which are of slot distance less than $f_{\min}$ to task j, i.e., for any $i \in G_j$, $|x_i - x_j| < f_{\min}$, where $i, j \in N$ and $i \neq j$;
$\omega_{i,j}$: =1 if storage task i and retrieval task j are in the same stack, i.e., $x_i = x_j$ and $y_i = y_j$; = 0 otherwise, where $i \in N \setminus N^R$, $j \in N^R$;
A_j^S : set of storage tasks that are in the same stack as a retrieval task j, i.e., $\omega_{i,j} = 1$ for any $i \in N \setminus N^R$ and $j \in N^R$;
r_i : the release time of task i, where $i \in N$;
d_i : the due date of task i, where $i \in N$;
h : the processing time for a YC to load/unload a single container;
p_i : the processing time of task i, where $i \in N$;
$t_{i,j}$: the time for a YC to move from the slot of task i to the slot of task j, and $t_{i,j} = |x_j - x_i|$, where $i, j \in N$ and $i \neq j$;
g : an upper bound of the difference between the number of tasks processed by YC1 and that of YC2;
M : a sufficiently large integer.

2) Decision Variables:

$\psi_{\max}$: the maximum tardiness of all tasks;
ψ_i : the tardiness of task i, where $i \in N$;
ε_i : the start time of processing task i, where $i \in N$;
δ_i : the completion time of processing task i, and $\delta_i = \varepsilon_i + p_i - 1$, where $i \in N$;
α_i^k : =1 if task i is assigned to YC k; = 0 otherwise, where $i \in N, k \in \{1, 2\}$;
$\beta_{i,j}^k$: =1 if both task i and task j are assigned to YC k and i is processed before task j; = 0 otherwise, where $i, j \in N$ and $i \neq j, k \in \{1, 2\}$;
$\gamma_{i,j}$: =1 if task $i \in G_j$ is processed before task j; = 0 otherwise, where $i, j \in N$ and $i \neq j$;
$\eta_{i,t}$: =1 if task i starts to be processed at t or $\varepsilon_i = t$; = 0 otherwise, where $i \in N, t \in T$;
$\rho_{i,t}$: =1 if task i is being processed at t; = 0 otherwise, where $i \in N, t \in T$;
$\tau_{x,t}^k$: =1 if YC k is located at slot x at t; = 0 otherwise, where $x \in L, t \in T, k \in \{1, 2\}$;
$\xi_{i,t}^k$: =1 if YC k processes task i at t; = 0 otherwise, where $i \in N, t \in T, k \in \{1, 2\}$.

3) Mathematical Model: We define two dummy tasks $\upsilon 1$ and $\upsilon 2$ with locations $x_{\upsilon 1} = 1$ and $x_{\upsilon 2} = x_{\max}$ are processed by YC1 and YC2 respectively. Let $\varepsilon_j = \delta_j = 0$ for $j \in \{\upsilon 1, \upsilon 2\}$. That is, YC1 and YC2 are initially located at slots 1 and $x_{\max}$ respectively.

The objective function of the MILP model is to minimize the largest tardiness, that is,

$$\min \ \psi_{\max}$$

With $\psi_{\max} = \max\{\psi_i\}$ for all i, we have

$$\psi_{\max} \ge \psi_i, \quad i \in N \tag{1}$$

$$\psi_i \ge 0, \quad i \in N \tag{2}$$

$$\psi_i \ge \delta_i - d_i, \quad i \in N \tag{3}$$

The start time of any task i is no earlier than its release time.

$$\varepsilon_i \ge r_i, \quad i \in N \tag{4}$$

If YC k processes task i before j, then the start time of task j is not earlier than the completion time of i plus the move time of the YC from slot x_i to slot x_j.

$$\varepsilon_j \ge \delta_i + t_{i,j} + 1 - (1 - \beta_{i,j}^k)M, \quad i, j \in N, \ i \neq j, \ k \in \{1, 2\} \tag{5}$$

The start times of the first tasks for both YC1 and YC2 cannot be earlier than the move times of the YCs from their initial slot locations to that of the tasks.

$$\varepsilon_j \geq x_j - (1-\alpha_j^1)M, \qquad j \in N \tag{6}$$

$$\varepsilon_j \geq x_{\max} - x_j + 1 - (1-\alpha_j^2)M, \quad j \in N \tag{7}$$

The equations below determine the start time of task i.

$$\sum_{t\in T} \eta_{i,t} = 1, i \in N \tag{8}$$

$$\sum_{t\in T} (t \times \eta_{i,t}) = \varepsilon_i, i \in N \tag{9}$$

For any storage task i and retrieval task j, if they are located in the same stack, then the start time of task i is later than the completion time of task j.

$$\varepsilon_i \geq \delta_j - (1-\omega_{i,j})M, \quad i \in A_j^S, \ j \in N^R \tag{10}$$

The processing time of task i is calculated by the values of u_i and h. The processing time of the storage task is h. For a retrieval task i, when it is on the top of a stack, i.e., $u_i = 0$, it is of no reshuffling and then $p_i = h$.

$$p_i = (1+2u_i) \times h, \quad i \in N \tag{11}$$

At any time t, the distance between YC1 and YC2 is at least $f_{\min}$ slots. Note that YC2 is always on the right side of YC1 with a larger slot index.

$$\sum_{x'\in L} \tau_{x',t}^2 \cdot x' - \sum_{x\in L} \tau_{x,t}^1 \cdot x \geq f_{\min}, \quad t \in T \tag{12}$$

Each task must be processed by either YC.

$$\alpha_i^1 + \alpha_i^2 = 1, \quad i \in N \tag{13}$$

If YC k processes task i preceding task j, then both tasks must be assigned to the YC.

$$\alpha_i^k \geq \beta_{i,j}^k, \quad i,j \in N, \ i \neq j, \ k \in \{1,2\} \tag{14}$$

$$\alpha_j^k \geq \beta_{i,j}^k, \quad i,j \in N, \ i \neq j, \ k \in \{1,2\} \tag{15}$$

For any two tasks i and j, if they are both assigned to YC k, i.e., $\alpha_i^k = \alpha_j^k = 1$, then they are processed sequentially, i.e., $\beta_{i,j}^k + \beta_{j,i}^k = 1$.

$$\beta_{i,j}^k + \beta_{j,i}^k \leq 1, \quad i,j \in N, \ i \neq j, \ k \in \{1,2\} \tag{16}$$

$$\beta_{i,j}^k + \beta_{j,i}^k \geq 1 - (2 - \alpha_i^k - \alpha_j^k)M, \quad i,j \in N, \quad i \neq j, \ k \in \{1,2\} \tag{17}$$

An upper bound g of workload difference between YC1 and YC2 is required to balance their workloads.

$$\sum_{i\in N} \alpha_i^1 - \sum_{j\in N} \alpha_j^2 \leq g, \quad i \neq j \tag{18}$$

$$\sum_{j\in N} \alpha_j^2 - \sum_{i\in N} \alpha_i^1 \leq g, \quad i \neq j \tag{19}$$

At any time t, each YC processes at most one task.

$$\sum_{i\in N} \xi_{i,t}^k \leq 1, \quad t \in T, k \in \{1,2\} \tag{20}$$

If YC k is processing task i at time t, it indicates that the task is assigned to the YC and is being processed at the time.

$$\xi_{i,t}^k \geq \alpha_i^k + \eta_{i,t} - 1, \quad i \in N, t \in T, k \in \{1,2\} \tag{21}$$

Provided that task i starts processing at t, it must be processed by either YC at the time.

$$\xi_{i,t}^1 + \xi_{i,t}^2 = \eta_{i,t}, \quad i \in N, t \in T \tag{22}$$

At any time t, each YC is located at one of the slots.

$$\sum_{x\in L} \tau_{x,t}^k = 1, \quad t \in T, k \in \{1,2\} \tag{23}$$

If YC k is processing task i at time t, the YC must be located at the slot of task i at the time.

$$\xi_{i,t}^k \leq \tau_{x_i,t}^k, \quad i \in N, \ x_i \in L, \ t \in T, \ k \in \{1,2\} \tag{24}$$

The start of processing task i implies that it is being processed at the time, and it cannot be on processing at any time point before its start.

$$\rho_{i,t} \geq \eta_{i,t}, \quad i \in N, \ t \in T \tag{25}$$

$$\sum_{t'=1}^{t-1} \rho_{i,t'} \leq (1-\eta_{i,t})M, \quad i \in N, t \in T \tag{26}$$

The following inequality ensures that the processing of any task i cannot be interrupted.

$$\rho_{i,t+1} + \rho_{i,t-1} - \rho_{i,t} \leq 1, \quad i \in N, t \in [2, T-1] \tag{27}$$

If task i is on processing in two consecutive units of time t and $t+1$, then it must be processed by the same YC.

$$\xi_{i,t}^k - \xi_{i,t+1}^k \geq -(2 - \rho_{i,t} - \rho_{i,t+1})M, \quad i \in N, \quad t \in T, \ k \in \{1,2\} \tag{28}$$

$$\xi_{i,t}^k - \xi_{i,t+1}^k \leq (2 - \rho_{i,t} - \rho_{i,t+1})M, \quad i \in N, t \in T, k \in \{1,2\} \tag{29}$$

For any two tasks i and j with a slot distance less than the minimum safety length, i.e., $|x_i - x_j| < f_{\min}$, they cannot be proceeded at the same time.

$$\gamma_{i,j} + \gamma_{j,i} = 1 \quad , i,j \in G_j \tag{30}$$

If i and j are of a slot distance less than $f_{\min}$ and are both assigned to YC k, then they must be processed in sequence.

$$\gamma_{i,j} \geq \beta_{i,j}^k, \quad i \in G_j, \ j \in N, \ k \in \{1,2\} \tag{31}$$

The range of decision variables are given below.

$$\psi_{\max}, \psi_i, \varepsilon_i, \delta_i \in Z^+, \quad i \in N \tag{32}$$

$$\alpha_i^k, \quad \beta_{i,j}^k, \ \gamma_{i,j}, \ \eta_{i,t}, \ \rho_{i,t}, \ \tau_{x,t}^k, \ \xi_{i,t}^k, \quad \in \{0,1\}, \quad i,j \in N, \quad i \neq j, \ x,x' \in L, \ t,t' \in T, \ k \in \{1,2\} \tag{33}$$

Constraints (1)-(3) calculate the tardiness value of each task. The start time of each task is bounded and figured out via Constraints (4)-(10). Constraint (11) counts the processing time of each task. The safety distance between YC1 and YC2 is guaranteed by Constraint (12). Constraints (13)-(15) present the relationship between tasks and YCs. Constraints (16)-(17) state the processing sequence

of any two tasks that are assigned to the same YC. The balance of workloads between YC1 and YC2 is expressed by Constraints (18)-(19). Constraints (20)-(22) describe the definition of $\xi_{i,t}^{k}$. Constraints (23)-(24) address the location of a YC in processing a task. The time continuities of processing any task and YC operation are presented in Constraints (25)-(29). Constraints (30)-(31) guarantee that any two tasks with processing interference must be processed one by one. Constraints (32)-(33) state the range of decision variables.

IV. Solution Method

The NP-hard nature of the problem has been proven by Gharehgozli *et al.* [26], the optimal solutions via Cplex can be obtained for small task instances. However, for instance with 35 or more tasks in this paper, the solver cannot output feasible solutions within a given time period. Therefore, it is necessary to explore more effective algorithms to solve the problem in practice.

According to the inter-crane interference constraint mentioned in Section III, the two YCs in the block are of a distance at least $f_{\min}$ slots at any time. Therefore, the following property is straightforward for any optimal solution of the considered problem.

Proposition 1. In an optimal solution, the tasks located in the leftmost and the rightmost f_{min} slots are assigned to YC1 and YC2, respectively.

Meanwhile, according to the data obtained in field research (such as Yangshan Deep-water Port, 2016), when there is more than one YC to process containers in a block, both workload balance and requirement of safety distance between YCs have to be satisfied simultaneously. Based on above observation, we propose a new algorithm named *Dividing, Sequencing and Comparing* (DSC). Its main idea is to randomly divide the tasks into two separate sets for the two YCs, and then select a best processing scheme from those produced by random, FCFS or EDD rule. The above division and selection processes are repeated for a given number of times, and finally a best possible scheme in the repetitions is chosen as the solution of the algorithm. In addition, we also propose a *Genetic Algorithm* (GA) based on yard crane scheduling characteristics with new chromosome representation to resolve the problem. Before presenting algorithms DSC and GA, we phrase the calculation of the dynamic processing time of each task, which is repeatedly applied for different processing sequences in both DSC and GA algorithms.

A. Calculation of Task Dynamic Processing Time

Given any processing sequence σ of tasks, let $\sigma_1 = ([1], [2], \ldots, [|N_1|])$ and $\sigma_2 = ([1], [2], \ldots, [|N_2|])$ be the two sub-sequences to be processed by YC1 and YC2, respectively, where $|N_1|, |N_2|$ are the number of tasks in the two sub-sequences. $|N_1| + |N_2| = |N|$. Let $z_{[i]}$ and $q_{[i]}$ be the original values of tier location index of any task $[i]$ and the number of containers in the corresponding stack, respectively. The dynamic processing time of task $[i]$ for $i = 1, 2, \ldots, |N_l|$ in sub-sequence σ_l $(l = 1, 2)$ is sequentially calculated in Algorithm 1.

Algorithm 1 Calculation of $p_{[i]}$ for Sequence σ_l $(l = 1, 2)$

Step 1. Identify each task $[j]$ $(1 \le j \le |N_l|)$ by the relation between $z_{[j]}$ and $q_{[j]}$. If $z_{[j]} > q_{[j]}$, then $[j]$ is a storage task; otherwise $[j]$ is a retrieval task if $z_{[j]} \le q_{[j]}$. Set $i = 1$.
Step 2. Calculate the processing time $p_{[i]}$ of task $[i]$. If $[i]$ is a storage task, then $p_{[i]} = h$; otherwise if $[i]$ is a retrieval task, then $u_{[i]} = q_{[i]} - z_{[i]}$ and $p_{[i]} = (1 + 2u_{[i]}) \cdot h$.
Step 3. Update the tier location of the tasks in the same stack with $[i]$ and the number of containers in this stack. Given that $[i]$ is a retrieval task, update $q_{[i']}$ and $z_{[i']}$ for each retrieval task $[i']$ which is in the same stack of task $[i]$. If $z_{[i']} > z_{[i]}$, i.e., container $[i']$ is on a higher tier than container $[i]$, reset $q_{[i']} := q_{[i']} - 1$ and $z_{[i']} := z_{[i']} - 1$; otherwise if $z_{[i']} < z_{[i]}$, reset $q_{[i']} := q_{[i']} - 1$.
Step 4. $i := i + 1$, and go to Step 2 if $i \le |N_l|$ $(l = 1, 2)$; otherwise if $i = |N_l|$ $(l = 1, 2)$, output the processing times of tasks in sequence σ_l $(l = 1, 2)$.

Note that in the above Step 3, as we have assumed, a retrieval task has to be processed before a storage task once they are in the same stack. Hence, if $[i]$ is a retrieval task, then all the following tasks $[i']$ on the same stack must also be retrieval ones. Similarly, if $[i]$ is storage task, then all its following tasks $[i']$ on the same stack are also storage ones. In the latter case, it is unnecessary to update $q_{[i']}$ and $z_{[i']}$ since $p_{[i']} = h$ always holds.

B. DSC Algorithm

The details of the DSC algorithm are described as Algorithm 2.

Algorithm 2 DSC

Step 0. (Initialization) Let $Maxt$ denote the loop times, and set $loop := 1$. Let σ^* be the best processing schedule temporarily obtained up to the current loop, and originally $\sigma^* = \emptyset$ with the objective value equal to infinity.
Step 1. Divide the $|N|$ tasks randomly into two disjoint sets with respect to Proposition 1 and workload balance constraint. The left-side and right-side sets are assigned to YC1 and YC2, respectively.
Step 2. Produce three processing sequences of tasks, using the random, FCFS and EDD rules, respectively. For each of the three processing sequences, apply Algorithm 1 to calculate the processing times of all the tasks.
Step 3. Produce three processing schedules, in which the constraint of YC interference is respected. That is, for any two consecutive tasks with a distance less than $f_{\min}$ slots, the latter task keeps waiting until the former one has been completed and the two YCs are of a safe distance.
Step 4. Let π be the best schedule among the above three processing schedules. If π is of a smaller objective value than that of σ^*, then reset $\sigma^* := \pi$.
Step 5. $loop := loop + 1$. If $loop \le Maxt$, then go to step 1; otherwise output the algorithm's best processing schedule σ^* and its objective value as well.

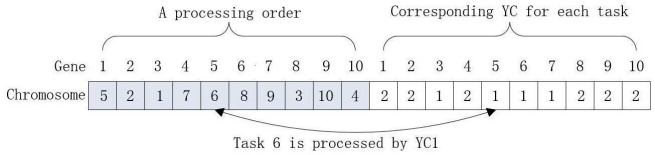


Fig. 3. Chromosome representation.

In *Step 1*, by Proposition 1, the tasks in the leftmost and rightmost $f_{\min}$ slots are assigned to YC1 and YC2, respectively. The remainder tasks are randomly assigned to the two YCs, provided that the numbers of tasks assigned to different YCs is of a difference at most g.

In *Step 2*, the random rule is to randomly generate a processing sequence of tasks based on their assignment in *Step 1*. In the FCFS sequence, a task with a smaller release time is processed earlier, while the EDD rule is to sequence tasks in the non-decreasing order of their due dates. *Step 3* is to produce three processing schedules according to the three sequences obtained in the previous step. In any schedule, each YC starts tasks in the given sequence as soon as possible. That is, it cannot be idle if the task to be processed is ready and the safety distance constraint is satisfied.

Step 4 is to update processing schedule in each running loop, and finally the algorithm selects the best schedule obtained in the $Maxt$ loops.

C. Genetic Algorithm

In this subsection, we adopt GA to solve the problem. The algorithm searches solutions repeatedly within the feasible domain of the problem. In each iteration, a population of feasible solutions (i.e., chromosomes) are renewed after applying a sequence of operations including selection, crossover and mutation. Below we phrase the corresponding solution presentations of these operations, feasibility condition and other involved details.

1) Chromosome Representation: Each feasible solution is represented by a chromosome, which describes the information on both task processing sequence and YC assignment. There are $2|N|$ elements or genes in each chromosome. The former $|N|$ genes denote a processing sequence of tasks, i.e., the digit of gene i is the index of the task to be processed in the ith ($1 \leq i \leq |N|$) position. The latter $|N|$ genes represent the assignment of YCs, that is, the digit of gene ($|N| + i$) is the index of the YC that processes the task in gene i ($1 \leq i \leq |N|$). Figure 3 illustrates a chromosome with $|N| = 10$ tasks where task 6 is processed in the 5th position, and it is processed by YC1.

2) Fitness Function: The considered problem aims at minimizing the maximum tardiness of tasks, and we adopt the objective value of a chromosome to indicate its fitness. That is, we define $fit(\theta) = \frac{1}{obj(\theta)}$ as the fitness function of chromosome θ where $obj(\theta)$ is the corresponding objective value. Note that before calculating the objective value of each chromosome, Algorithm 1 is applied to count the processing times of all the tasks.

3) Genetic Operations: The major genetic operations include selection, crossover and mutation operations.

a) Gene repair strategy: When some genes change their digits during the crossover and mutation operations, it may induce infeasible chromosomes. We present a gene repair strategy to handle such situation. If there is two genes have the same position in the first $|N|$ gene, one task must be assigned to another position. We apply the repair strategy to randomly generate processing positions for those tasks, whose indices correspond to the new values for the genes. For the latter $|N|$ genes, the strategy corrects the assignment of YC such that Proposition 1 and the workload balance are satisfied. Especially, the workload balance is ensured by reassigning tasks in the middlemost slots, if necessary.

b) Selection: The selection of newly generated chromosomes depends on their fitness. We adopt the method of *roulette wheel selection* which chooses each chromosome in proportionated with its fitness. Hence, a chromosome with a higher fitness is more likely to be selected.

c) Crossover: The *two-point crossover* approach is adopted in this work. For any two chromosomes undergoing crossover, two crossover points are randomly generated such that the first and the second crossover points fall in the former and the latter $|N|$ genes, respectively. The gene segments between the two swapping points in the two chromosomes are swapped. Each gene swapping results in two new chromosomes. Using the previously gene repair strategy, we obtain two newly generated feasible chromosomes.

d) Mutation: To guarantee the diversification of the population in the new generation, we apply *uniform mutation* operation to renew some chromosomes. We randomly choose some new chromosomes in which a part of genes are randomly selected for mutation with a predetermined probability. The digit in each selected gene is replaced with another uniformly distributed random number falling in $[1, |N|]$ if $1 \leq i \leq |N|$ and $[1, 2]$ if $|N| + 1 \leq i \leq 2|N|$. The gene repair strategy is applied again following the mutation operation.

4) Termination Conditions: The generation of gene population is repeated again and again through the above genetic operations until one of the following two termination conditions is met: (1) the number of iterations has reached a preset maximum value MAXGEN; and (2) the fitness value of the best chromosome in the population remains unchanged in 100 consecutive generations.

V. Computational Experiments

In this section, we perform numerical experiments to show the efficiency and the effectiveness of the proposed solution methods on a series of instances. Computational experiments are conducted on a PC with Intel Core i5, 3GHz processors and 4 GB RAM, and the computation time is calculated in CPU seconds. The algorithms DSC and GA are implemented in Matlab R2014. And Cplex 12.6 is used to directly solve the integer programming model proposed in Section III and its relaxed model. The time limit of Cplex is set to 3600 seconds. We set $Maxt = 20000$ and $MAXGEN = 500$ so that the running times of DSC and GA algorithms are comparable in

the experiments. To choose the best crossover and mutation probabilities for GA, we compare various combinations of crossover probabilities {0.7, 0.8, 0.9} and mutation probabilities {0.05, 0.10, 0.15} in the preliminary experiments. The combination (0.8, 0.15) outperforms the others and is then adopted in the remaining computational experiments.

A. Instance Settings

The experiments are carried out on 700 randomly generated instances. The instances are with reference to the practical data of Shanghai Yangshan Deep-water Port. The detailed settings of instances are as follows.

- The block is of a size with 40 slots, 6 rows and 5 tiers, i.e., $x_{max} = 40$, $y_{max} = 6$, and $z_{max} = 5$, which are in accordance with the practical situation in Yangshan Port.
- The minimum safety distance is set to 8 slots, i.e., $f_{\min} = 8$.
- YC spends one unit of time to move one slot away, and a single container move costs three time units, i.e., $h = 3$.
- The release time r_i of task i follows uniform distribution in interval $(0, 2|N|]$ in accordance with the number of tasks. The due date d_i follows uniform distribution in interval $[r_i + a_1, 2r_i + a_2]$ related with release times, where $a_1 = a_2 = 5$ and 8 for $|N| = 10$ and 15, respectively, while $a_1 = 10, a_2 = 15$ for $|N| \geq 25$. Notice that the distribution range of release times of tasks is in accordance with the number of tasks. The distribution range of due dates of tasks are related with release times. Especially for instances with only 10 or 15 tasks, we set a narrower and earlier distribution range of due dates to ensure a positive objective value in the optimal solution.
- The ratios between the number $|N_s|$ of storage tasks and the number $|N_r|$ of retrieval tasks are set to 7:3, 6:4, 5:5, 4:6 and 3:7. To reveal the effect of reshuffling operations, for each ratio, 20 instances are randomly generated, and only the average values of 20 instances are presented in the following tables.

We consider various instance sizes in the experiments, that is, $|N| = 10, 15, 25, 30, 35, 40$ and 50. These values approximately double the instance sizes for single YC case analysis in literature (Vis and Carlo [22] and Gharehgozli *et al.* [26]).

B. Experiment Results

For instances in sizes $|N| = 10, 15, 25, 30$, the Cplex solver can output optimal solutions of the MILP model presented in section III. In this case we measure the performances of DSC and GA algorithms in comparison with optimal solutions. For instances with $|N| = 35, 40, 50$, the solver cannot produce any feasible solutions within the pre-set 3600 seconds, so that we relax the safety distance constraint of the MILP to devise MILP model's lower bound by Cplex to evaluate the performances of the two heuristics.

As previously mentioned, Cplex can produce optimal solutions for instances with 10, 15, 25 and 30 tasks. The corresponding results of DSC and GA algorithms are listed in rows 3-27 of Table II. In the first column of the table, "$\frac{|N_s|}{|N_r|}$" represents the ratio of the numbers of storage and retrieval tasks. Column "Opt." means the average objective value of exact solutions produced by Cplex, and column "Time" is its running time in seconds. Columns "UB_{DSC}" and "UB_{GA}" mean the objective value of optimal or near-optimal solutions by DSC and GA, respectively. Column "Gap" denotes the *relative error* between Opt. and UB_{Alg} where $Alg \in \{DSC, GA\}$, that is, Gap = $\frac{UB_{Alg} - Opt.}{Opt.} * 100\%$.

TABLE II
COMPUTATIONAL RESULTS OF INSTANCES

$\|N_s\|$	Cplex		DSC			GA		
$\|N_r\|$	**Opt.**	Time	UB_{DSC}	Time	Gap	UB_{GA}	Time	Gap
				($\|N\| = 10$)				
7:3	20.39	14.33	20.39	4.58	0.00	20.51	3.93	0.59
6:4	25.46	14.33	25.46	4.58	0.00	25.62	3.93	0.63
5:5	28.15	14.33	28.15	4.58	0.00	28.33	3.93	0.64
4:6	31.27	14.33	31.27	4.58	0.00	31.47	3.93	0.64
3:7	34.82	14.34	34.82	4.58	0.00	35.04	3.93	0.63
				($\|N\| = 15$)				
7:3	26.93	49.61	26.93	5.41	0.00	27.28	4.28	1.52
6:4	29.49	48.95	29.52	5.42	0.10	29.96	4.28	1.59
5:5	34.33	49.32	34.33	5.41	0.00	34.93	4.30	1.75
4:6	38.04	49.73	38.04	5.42	0.00	38.68	4.31	1.68
3:7	42.13	48.78	42.13	5.41	0.00	42.82	4.31	1.64
				($\|N\| = 25$)				
7:3	39.19	915.18	39.19	6.17	0.00	40.03	5.19	2.14
6:4	42.84	914.56	42.96	6.16	0.28	43.72	5.20	2.05
5:5	46.13	914.69	46.13	6.17	0.00	47.13	5.20	2.17
4:6	50.28	915.27	50.42	6.17	0.28	51.38	5.20	2.19
3:7	55.47	915.34	55.78	6.17	0.56	56.72	5.20	2.25
				($\|N\| = 30$)				
7:3	40.13	2834.29	40.19	6.62	0.15	41.27	6.73	2.84
6:4	43.69	2833.83	43.69	6.63	0.00	45.01	6.73	3.02
5:5	52.62	2833.92	52.80	6.63	0.34	54.22	6.73	3.04
4:6	63.41	2834.16	63.72	6.63	0.46	65.34	6.74	3.04
3:7	76.13	2833.74	76.50	6.63	0.49	78.43	6.76	3.02
Ave		953.15		5.70	0.13		5.04	1.61
	LB				Gap_{DSC}			Gap_{GA}
				($\|N\| = 35$)				
7:3	57.41	725.36	60.17	7.36	4.81	65.14	7.38	13.46
6:4	60.15	732.45	63.06	7.35	4.86	67.72	7.40	12.59
5:5	64.67	731.68	67.65	7.36	4.61	74.52	7.41	15.23
4:6	70.94	733.02	73.95	7.35	4.28	81.43	7.40	14.83
3:7	75.02	731.49	78.24	7.35	4.29	86.31	7.39	15.05
				($\|N\| = 40$)				
7:3	66.37	1142.64	70.39	7.78	6.06	80.63	8.24	21.49
6:4	72.48	1139.47	76.98	7.76	6.21	87.42	8.26	20.61
5:5	80.23	1138.49	85.16	7.78	6.14	96.91	8.27	20.79
4:6	87.96	1139.86	93.58	7.78	6.39	105.80	8.27	20.28
3:7	96.65	1139.14	102.68	7.78	6.24	117.44	8.27	21.51
				($\|N\| = 50$)				
7:3	70.25	3407.82	75.39	8.57	7.32	91.43	11.70	30.15
6:4	75.39	3410.35	81.02	8.57	7.47	101.24	11.69	34.29
5:5	85.93	3398.67	92.17	8.57	7.26	112.72	11.70	31.18
4:6	94.34	3409.64	101.04	8.58	7.10	125.30	11.70	32.82
3:7	106.65	3403.93	114.25	8.57	7.13	139.52	11.70	30.82
Ave		1758.93		7.90	6.01		9.12	22.34

The first 27 rows of Table II report the results for instances where the number $|N|$ of tasks varies from 10 to 30. According to the table, the running time of Cplex increases rapidly from 14.33 seconds to 2834.29 seconds as the number of tasks is tripled, while both DSC and GA solve all the instances in less than 7 seconds. It means the running time of Cplex other than DSC and GA is quite sensitive to instance size. Meanwhile, the average "Gap" and the maximal "Gap" of DSC are 0.13% and 0.56%, respectively, and the

corresponding values of GA are 1.61% and 3.04%. Therefore, we claim that both algorithms are effective to produce high quality solutions in a short time, and DSC outperforms GA in terms of solution quality.

We have the following further observations by rows 3-27 of Table II. First of all, given any ratio of $\frac{|N_s|}{|N_r|}$, the average *UBs* as well as computational times of both algorithms increase with instance size $|N|$.

Secondly, for any given instance size, the average *UBs* of both algorithms decrease with the ratio of $\frac{|N_s|}{|N_r|}$. That is, the *UBs* increase as the proportion of retrieval tasks rises. This result verifies that more retrieval tasks cause more tardiness. Note that the ratio has no influence on the computational times of both DSC and GA algorithms.

For instances with 35 or more tasks, i.e., $|N| = 35, 40, 50$, Cplex cannot output feasible solutions within 3600s. However, it may provide a lower bound (*LB*) of an optimal solution by relaxing the safety distance constraint of the proposed model. Furthermore, to evaluate the main reason of causing most computational time, we reprocess the experiments for the same instances with 10 to 30 tasks in Table II while omitting the interference constraints between YCs. Cplex outputs LBs with an average relative error to the optimal solutions of 2.44% and an average running time equal to 270.29s. Compared with 953.15s for the optimal solutions, we conclude that its most computational time is to verify the interference constraints in producing an optimal solution.

For row 28 in Table II, the second column "LB" indicates the average lower bounds of the MILP model. Columns "Gap_{DSC}" and "Gap_{GA}" denote the relative error between the lower bound and the objective value by the respective approach, i.e., Gap $= \frac{Opt.(UB_{DSC}, UB_{GA}) - LB}{LB} * 100\%$. The rest columns are of the same meanings as in the first 27 rows of the table.

Rows 29-47 of Table II list the numerical results for instances with 35, 40 and 50 tasks. Firstly, the values of "Gap" for all the three approaches increase with instance size. Especially, the gap for DSC varies from 4.28% to 7.47% with an average of 6.01%, and for GA, the gap changes from 12.59% to 34.29% with an average of 22.34%. One possible reason is that the number of retrieval tasks and their dynamic processing times have a great impact on the objective value.

Secondly, the running times of DSC and GA algorithms rise slightly with instance size. It implies that both heuristics are able to solve large size instances in a short time. Moreover, the results indicate that DSC outperforms GA for all the instances in terms of running time and solution quality.

VI. Conclusion

In this work, we have modeled and solved a practical operational problem arising in container terminals, i.e., scheduling two YCs to execute a set of storage and retrieval tasks in a block with respect to reshuffling operations and interference. Dynamical processing times of retrieval tasks caused by reshuffling operations are firstly introduced in this work. We then build a mixed integer programming model with the objective of minimizing the maximum tardiness of tasks. Two efficient heuristics DSC and GA are designed to solve the problem. Experimental results on 700 randomly generated instances show (i) the proportion of retrieval tasks as well as their dynamic processing times in an instance has an impact on the objective value of a solution but not computational time; (ii) DSC outperforms GA in term of solution quality for all instances; (iii) by the numerical results in Tables II, DSC is able to solve the practical problem of two YC scheduling in a container block.

Future research might include several aspects as follows. (1) During reshuffling operations, reshuffle containers need not be put back to their original stacks. It may improve the results of yard crane scheduling, but it needs much more computational resources to calculate and memorize a large number of possible interim states of all the stacks in processing tasks. It also complicates the yard crane management, and thus more powerful approaches are required to solve the corresponding scenario. (2) The storage and retrieval tasks may be with stochastic release times in real application, but this case has not been investigated in literature (Gharehgozli *et al.* [26], Wu *et al.* [27], etc). For the problem, stochastic programming should be an appropriate method. (3) As various reasons can cause stochastic processing time of a task, considering task processing time being an interval instead of a fixed value could be an interesting challenge.

References

[1] K. H. Kim and G.-P. Hong, "A heuristic rule for relocating blocks," *Comput. Oper. Res.*, vol. 33, no. 4, pp. 940–954, Apr. 2006.

[2] L. H. Lee, E. P. Chew, K. C. Tan, and Y. Han, "An optimization model for storage yard management in transshipment hubs," *OR Spectr.*, vol. 28, no. 4, pp. 539–561, Oct. 2006.

[3] D.-H. Lee, Z. Cao, and Q. Meng, "Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm," *Int. J. Prod. Econ.*, vol. 107, no. 1, pp. 115–124, May 2007.

[4] X. Guo and S. Y. Huang, "Dynamic space and time partitioning for yard crane workload management in container terminals," *Transp. Sci.*, vol. 46, no. 1, pp. 134–148, 2012.

[5] S. Y. Huang, Y. Li, M. Lau, and T. C. Tay, "Yard crane deployment in container terminals," in *Proc. Winter Simulation Conf.*, Dec. 2014, pp. 1735–1746.

[6] F. Meisel and M. Wichmann, "Container sequencing for quay cranes with internal reshuffles," *OR Spectr.*, vol. 32, no. 3, pp. 569–591, Jul. 2010.

[7] C. I. Liu, H. Jula, and P. A. Ioannou, "Design, simulation, and evaluation of automated container terminals," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 1, pp. 12–26, Mar. 2002.

[8] A. V. Goodchild and C. F. Daganzo, "Double-cycling strategies for container ships and their effect on ship loading and unloading operations," *Transp. Sci.*, vol. 40, no. 4, pp. 473–483, 2006.

[9] S. Kang, J. C. Medina, and Y. Ouyang, "Optimal operations of transportation fleet for unloading activities at container ports," *Transp. Res. B, Methodol.*, vol. 42, no. 10, pp. 970–984, Dec. 2008.

[10] V. D. Nguyen and K. H. Kim, "Heuristic algorithms for constructing transporter pools in container terminals," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 517–526, Jun. 2013.

[11] H. J. Carlo, I. F. A. Vis, and K. J. Roodbergen, "Storage yard operations in container terminals: Literature overview, trends, and research directions," *Eur. J. Oper. Res.*, vol. 235, no. 2, pp. 412–430, Jun. 2014.

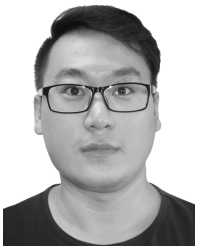
[12] L. Zhen, X. Jiang, L. H. Lee, and E. P. Chew, "A review on yard management in container terminals," *Ind. Eng. Manage. Syst.*, vol. 12, no. 4, pp. 289–305, Dec. 2013.

[13] M. Liu, C.-Y. Lee, Z. Zhang, and C. Chu, "Bi-objective optimization for the container terminal integrated planning," *Transp. Res. B, Methodol.*, vol. 93, pp. 720–749, Nov. 2016.

[14] M. Cichenski, F. Jaehn, G. Pawlak, E. Pesch, G. Singh, and J. Blazewicz, "An integrated model for the transshipment yard scheduling problem," *J. Scheduling*, vol. 20, no. 1, pp. 57–65, Feb. 2017.
[15] K. H. Kim and K. Y. Kim, "An optimal routing algorithm for a transfer crane in port container terminals," *Transp. Sci.*, vol. 33, no. 1, pp. 17–33, 1999.
[16] A. Narasimhan and U. S. Palekar, "Analysis and algorithms for the transtainer routing problem in container port operations," *Transp. Sci.*, vol. 36, no. 1, pp. 63–78, 2002.
[17] W. C. Ng and K. L. Mak, "Yard crane scheduling in port container terminals," *Appl. Math. Model.*, vol. 29, no. 3, pp. 263–276, Mar. 2005.
[18] X. Guo, S. Y. Huang, W. J. Hsu, and M. Y. H. Low, "Dynamic yard crane dispatching in container terminals with predicted vehicle arrival information," *Adv. Eng. Inform.*, vol. 25, no. 3, pp. 472–484, Aug. 2011.
[19] A. H. Gharehgozli, Y. Yu, R. de Koster, and J. T. Udding, "A decision-tree stacking heuristic minimising the expected number of reshuffles at a container terminal," *Int. J. Prod. Res.*, vol. 52, no. 9, pp. 2592–2611, 2014.
[20] S. Y. Huang, X. Guo, W. J. Hsu, and W. L. Lim, "Embedding simulation in yard crane dispatching to minimize job tardiness in container terminals," in *Proc. Winter Simulation Conf.*, vol. 2. Dec. 2012, pp. 1–11.
[21] Z. Cao, D.-H. Lee, and Q. Meng, "Deployment strategies of double-rail-mounted gantry crane systems for loading outbound containers in container terminals," *Int. J. Prod. Econ.*, vol. 115, no. 1, pp. 221–228, Sep. 2008.
[22] I. F. A. Vis and H. J. Carlo, "Sequencing two cooperating automated stacking cranes in a container terminal," *Transp. Sci.*, vol. 44, no. 44, pp. 169–182, 2010.
[23] W. C. Ng, "Crane scheduling in container yards with inter-crane interference," *Eur. J. Oper. Res.*, vol. 164, no. 1, pp. 64–78, Jul. 2005.
[24] W. Li, Y. Wu, M. E. H. Petering, M. Goh, and R. de Souza, "Discrete time model and algorithms for container yard crane scheduling," *Eur. J. Oper. Res.*, vol. 198, no. 1, pp. 165–172, Oct. 2009.
[25] W. Li, M. Goh, Y. Wu, M. E. H. Petering, R. de Souza, and Y. C. Wu, "A continuous time model for multiple yard crane scheduling with last minute job arrivals," *Int. J. Prod. Econ.*, vol. 136, no. 2, pp. 332–343, Apr. 2012.
[26] A. H. Gharehgozli, G. Laporte, Y. Yu, and R. de Koster, "Scheduling twin yard cranes in a container block," *Transp. Sci.*, vol. 49, no. 3, pp. 686–705, 2014.
[27] Y. Wu, W. Li, M. E. H. Petering, M. Goh, and R. de Souza, "Scheduling multiple yard cranes with crane interference and safety distance requirement," *Transp. Sci.*, vol. 49, no. 4, pp. 990–1005, 2015.
[28] A. Ehleiter and F. Jaehn, "Housekeeping: Foresightful container repositioning," *Int. J. Prod. Econ.*, vol. 179, pp. 203–211, Sep. 2016.
[29] C. Zhang, Y.-W. Wan, J. Liu, and R. J. Linn, "Dynamic crane deployment in container storage yards," *Transp. Res. B, Methodol.*, vol. 36, no. 6, pp. 537–555, Jul. 2002.
[30] R. K. Cheung, C.-L. Li, and W. Lin, "Interblock crane deployment in container terminals," *Transp. Sci.*, vol. 36, no. 1, pp. 79–93, 2002.
[31] M. Bazzazi, N. Safaei, and N. Javadian, "A genetic algorithm to solve the storage space allocation problem in a container terminal," *Comput. Ind. Eng.*, vol. 56, no. 1, pp. 44–52, Feb. 2009.
[32] P. Angeloudis and M. G. H. Bell, "An uncertainty-aware AGV assignment algorithm for automated container terminals," *Transp. Res. E, Logistics Transp. Rev.*, vol. 46, no. 3, pp. 354–366, May 2010.
[33] S. Saini, D. Roy, and R. de Koster, "A stochastic model for the throughput analysis of passing dual yard cranes," *Comput. Oper. Res.*, vol. 87, pp. 40–51, Nov. 2017.

Feifeng Zheng received the B.S. degree in information management, the M.S. degree in management science and engineering, and the Ph.D. degree in management science and engineering from Xi'an Jiaotong University, Xi'an, China, in 1998, 2003, and 2006, respectively.

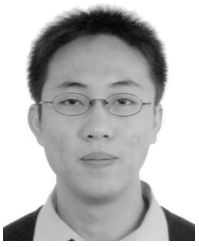
He is currently a Professor with Donghua University, Shanghai. His research interests include production scheduling and container terminal resource scheduling.



Xiaoyi Man received the B.S. degree in human resource management and the M.S. degree in technological economics and management from the Harbin University of Science and Technology, Harbin, China, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree with the Glorious Sun School of Business and Management, Donghua University. His research interests include mathematical modeling and port logistics optimization.

Feng Chu (M'07–SM'12) received the B.S. degree in electrical engineering from the Hefei University of Technology, Hefei, China, in 1986, the M.S. degree in metrology, automatic control, and electrical engineering from the National Polytechnic Institute of Lorraine, Lorraine, France, in 1991, and the Ph.D. degree in automatic control, computer science, and production management from the University of Metz, Metz, France, in 1995.

She is currently a Full Professor of operations research with the University of Évry Val d'Essonne, the University of Paris-Saclay, *É*vry, France, and a Co-Leader of the Algorithmic, Operations Research, Bioinformatics and Statistical Learning Group. She is the author or co-author of over 80 papers in international journals. Her research interests include the modeling, analysis, and optimization of complex systems, such as intelligent transportation systems and logistic and production systems based on combinatorial optimization, operations research, and petri nets. She is an IPC member for over 70 international conferences. She received the award by the French Ministry of Education for her doctoral supervision and research activities in 2005. She was an Associate Editor for the IEEE T-SMC, Part C from 2010 to 2013. She is currently an Associate Editor for the IEEE T-ITS and the IEEE TASE. She was selected for the Foreign Hundred Talents Program of Anhui province of China in 2013 and the Thousand Talents Program of Sichuan province in 2016.



Ming Liu (SM'17) received the B.S. and Ph.D. degrees in management science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2005 and 2010, respectively.

He is currently an Associate Professor with Tongji University, Shanghai. His research interests include port logistics optimization and production scheduling.

Chengbin Chu received the B.S. degree in electrical engineering from the Hefei University of Technology, Hefei, China, in 1985, and the Ph.D. degree in computer science from the University of Metz, Metz, France, in 1990.

He was with the National Research Institute in Computer Science and Automation, Metz, from 1987 to 1996. He was a Professor with the University of Technology of Troyes, Troyes, France, from 1996 to 2008, where he was the Founding Director of the Industrial Systems Optimization Laboratory. He is currently a Professor with ESIEE Paris, Université Paris-Est, France. He is interested in operations research and modeling, analysis, and optimization of supply chain and production systems.

Dr. Chu is currently an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING and the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.