IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 12, 2020, accepted August 20, 2020, date of publication August 24, 2020, date of current version September 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3019004

A Job Sequencing Problem of an Overhead Shuttle Crane in a Rail-Based Automated Container Terminal

HENOKH YERNIAS FIBRIANTO[1,*], BONGGWON KANG[2,*], AND SOONDO HONG[2]
[1]GrabTaxi Holdings Pte., Ltd., Singapore 573972
[2]Department of Industrial Engineering, Pusan National University, Busan 46241, South Korea

Corresponding author: Soondo Hong (soondo.hong@pusan.ac.kr)

This work was supported in part by the framework of the International Cooperation Program managed by the National Research Foundation of Korea (NRF) under Grant NRF-2016K1A3A1A48954044, and in part by the NRF funded by the Korean Government (the Ministry of Science and ICT) under Grant NRF-2020R1A2C2004320.

*Henokh Yernias Fibrianto and Bonggwon Kang are co-first authors

ABSTRACT This study proposes a job scheduling model and its heuristics for an automated container terminal with an overhead shuttle crane (OS) to reduce the total tardiness time of flatcars and external trucks by considering the separation of each job into a main job, and a premarshaling or remarshaling job. The OS is busy or idle according to the fluctuations in the processing times of different pieces of equipment. We identify the OS job sequencing problem considering job separation (OSJSPS) as a mixed-integer programming (MIP) model, which simultaneously sequences a set of jobs and searches for their possible separation into premarshaling and remarshaling jobs. We present a two-stage genetic algorithm (TGA) based on two local improvement procedures: an iterative local search procedure and an opportunistic job separation procedure. We conclude that the two-stage genetic algorithm reduces the total tardiness time of the container terminal's flatcars and external trucks as the number of OS jobs increases.

INDEX TERMS Automated container terminal, job sequencing, job separation, overhead shuttle crane.

I. INTRODUCTION

Increased trade volume carried by container ships has motivated container terminal operators and engineers to develop more sophisticated strategies and container terminal designs [2]–[5] for improving throughput and storage capacity [6]–[8]. One design concept is the rail-based automated container terminal (RACT) [8], which comprises a quay area, a transport area, and a storage yard area.

Figure 1 (a) and (b) shows the infrastructure of the RACT. In the quay area, containers are transferred between a vessel and the transporters on land by a double trolley quay crane [9]. In the transport lines, containers are transferred between the quay area and the storage area by flatcars, which operate on top of a set of fixed rails. The interactions between flatcars and the overhead shuttle crane (OS) or a quay crane (QC) in the transport lines occur at a handover location. In the storage yard area, containers are temporarily stored and retrieved by the OS which interacts with a flatcar or an external truck (ET).

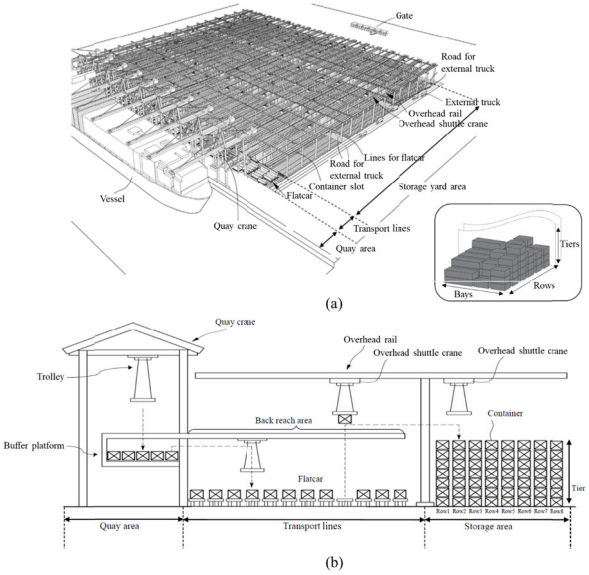

FIGURE 1. Schematic layout of RACT: (a) isometric view; (b) side view [1].

When designing a container terminal layout, the terminal planner needs to determine the size of its storage areas and the

The associate editor coordinating the review of this manuscript and approving it for publication was Bohui Wang.

number of yard cranes per area, considering investment cost and crane capacity [10], [11], and when designing a RACT layout, the planner wants to maximize the performance of the single overhead shuttle, considering investment cost of cranes and overhead rails. When a terminal includes automatic stacking cranes (ASC), the planner wants to maximize ASC performance [12], ASC dispatching [13], ASC scheduling [14], etc.

In any container layout, the planner needs to ensure that the synchronization procedure supports operating at maximum capacity and preventing delays [15], similar to secure communication [16] or multi-agent control [17] procedures. In a RACT layout, the process planner sets a due time for each piece of terminal equipment required for the synchronization [8]. If any due time is missed, the risk of other jobs being completed late increases. For instance, if an OS fails to serve a flatcar by the due time, the flatcar obstructs other flatcars sharing the same rail and may even delay subsequent QC operations.

Fluctuations between QC processing times and flatcar moving times are mostly influenced by containers' locations on the vessel [18], while fluctuations between OS processing times and flatcar moving times are mostly influenced by the container terminal's flatcar routing policy and the distance between the flatcar's origin and destination [8]. Consequently, some jobs will be completed after their due times during busy periods and OS utilization will decline during idle periods. Terminal management has to account for the inevitable fluctuations affecting the synchronization.

This article describes an OS job sequencing model that determines which jobs should be premarshaled and remarshaled by using the OS idle time to sequence them. Premarshaling transports a retrieval container closer to its expected destination location in the terminal in preparation for the retrieval request, while remarshaling transports a storage container from its temporary location to its final location in preparation for the storage request (see Figure 2). Both procedures help to reduce vehicle waiting times. The OS job sequencing model determines which jobs should be premarshaled or remarshaled and sequences the set of jobs in the OS. Finally, the OS job sequencing with premarshaled and remarshaled jobs minimizes the total delay time of vehicles.

This article contributes to the published literature as follows. (1) We formulate an OS job sequencing mixed-integer programming (MIP) model that selectively combines premarshaled and remarshaled jobs for a RACT. The proposed model smooths the synchronization procedure by determining whether a requested job should be separated, or not, from the main job and auxiliary job under the premarshaling and remarshaling allowance. (2) We develop a two-stage genetic algorithm based on an iterative local search procedure and an opportunistic job separation procedure. The two-stage genetic algorithm reduces the total tardiness time of vehicles.

The remainder of this article is structured as follows. Section II reviews the relevant literature. Section III describes the OS job sequencing problem in greater detail. Section IV describes the mathematical model of the OS job sequencing problem, considering premarshaling and remarshaling. Section V describes the two-stage genetic algorithm. Section VI discusses the results of the heuristics for reducing average job tardiness time, and the simulations and case study used to validate it. Chapter VII concludes and suggests future research.

II. LITERATURE REVIEW

A. MANAGEMENT OF YARD CRANE AND TRANSPORTER

To operate a container terminal at maximum capacity, several studies propose an integrated approach [19]–[21]. Lau and Zhao [22] develop an integrated model to manage the transporter, QC, and yard crane and to shorten container processing times. Cao *et al.* [23] propose an integrated yard crane and transporter problem to improve loading. Homayouni *et al.* [24] address the integrated scheduling of multiple pieces of equipment in an automated container terminal that uses a split-platform storage/retrieval system. He *et al.* [25] focus on the trade-off between equipment efficiency and energy consumption by integrating the management of the QC, transporter, and yard crane. Due to the direct relationship between the storage location and the routing path of a straddle carrier, Dkhil *et al.* [26] study an integrated space allocation and vehicle dispatching problem. Kress *et al.* [27] present a partitioning min-max weighted matching problem to model the temporary storage allocation problem and the job-to-equipment assignment problem. Chen *et al.* [28], who propose a scheduling model, considering the cooperation between vehicles and cranes, use a network flow with space and time windows, and emphasize that cranes and vehicles are synchronized.

Other studies focus on managing transporters independently [29], [30]. Briskorn *et al.* [31] approach the automated guided vehicle (AGV) dispatching problem from the viewpoint of inventory management. Grunow *et al.* [32] study the assignment of jobs considering the multi-load capability of AGVs. Kim *et al.* [33] develop an algorithm to detect and prevent deadlock between AGVs, where the AGV travel area is partitioned into grid blocks. Nguyen and Kim [34] propose a heuristic algorithm to dispatch automated lifting vehicles (ALVs) considering the buffer space for the containers in the apron and yard areas.

B. YARD CRANE JOB SEQUENCING

The management of yard cranes and jobs has been rigorously examined [35]–[38]. Chen and Langevin [39] use a metaheuristic to minimize the makespan of rubber-tired gantry crane (RTGC) jobs. Gharehgozli *et al.* [40] focus on the makespan minimization of single rail-mounted gantry crane (RMGC) jobs. Later, Gharehgozli *et al.* [41] propose an adaptive large neighborhood search algorithm to minimize the makespan of twin yard cranes' jobs considering job precedence. Vis and Carlo [42] and Nossack *et al.* [43] aim to minimize the makespan of crossover cranes' jobs.

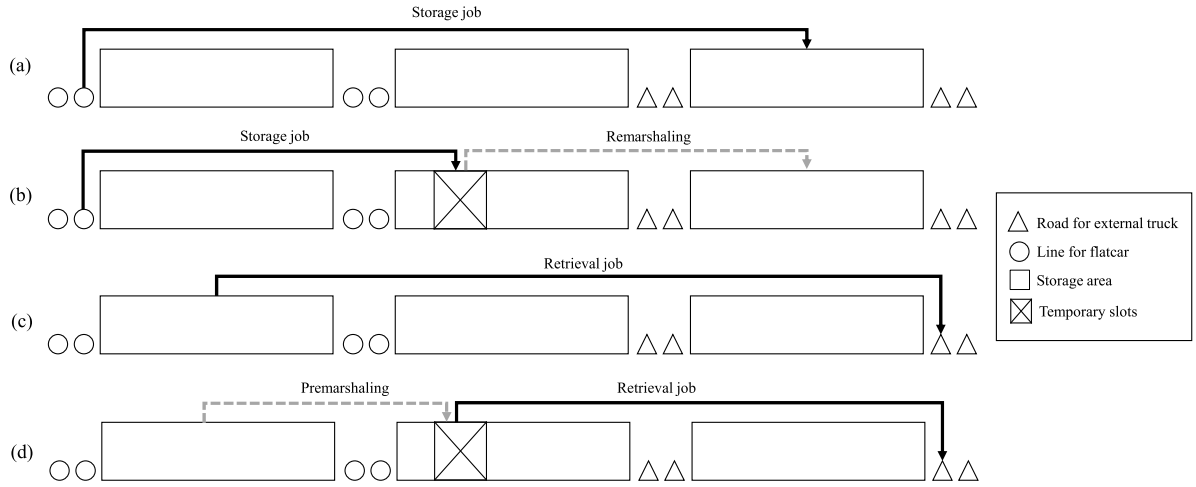

FIGURE 2. OS operation type: (a) storage job without remarshaling (b) storage job with remarshaling (c) retrieval job without premarshaling (d) retrieval job with premarshaling.

Park *et al.* [44] develop a heuristic and a simulated annealing algorithm to minimize the waiting times of the internal and external vehicles by scheduling twin RMGCs. Choe *et al.* [45] incorporate the possibilities of remarshaling jobs into scheduling twin RMGCs considering the minimization of penalties for the remarshaling jobs, makespan, and the transporters' waiting times. Galle *et al.* [46] incorporate the selection of container slots into the job sequencing problem to minimize makespan and the chance of future relocation.

Guo *et al.* [47] address the job sequencing problem involving a single yard crane to minimize external and internal vehicle delays. Ng and Mak [48] develop a heuristic to sequence yard crane jobs to minimize vehicle waiting times. Similarly, Li *et al.* [49] propose a heuristic to sequence yard crane jobs by minimizing the earliness and tardiness of each job. Huang and Li [50] present three proxies for the job sequencing problem: the first considers the minimization of weighted vessel loading time, the second considers the minimization of weighted vehicle tardiness, and the third considers the minimization of makespan.

C. YARD CRANE JOB SCHEDULING

When the major decisions also include the time in which each job must be performed, the job sequencing problem becomes a job scheduling problem that must consider the interference between multiple cranes sharing the same track. Cao *et al.* [51] discuss the job scheduling problem for cross-over cranes to minimize makespan. Dorndorf and Schneider [52] study a container terminal that utilizes triple crossover stacking cranes. Han *et al.* [53] attempt to mitigate interferences between cranes to minimize the makespan when input/output points are available for immediate ASC operations.

The job scheduling problem can also include measuring crane productivity according to the number of productive moves per hour. Lee *et al.* [54] propose a heuristic to minimize the loading time of yard cranes by determining which crane should perform a job at a specific time. Li *et al.* [55] develop a discrete-time model and propose a heuristic coupled with a rolling horizon approach to minimize the earliness and tardiness of each job. Zheng *et al.* [56] incorporate reshuffling jobs and yard cranes' interference into a twin yard cranes scheduling problem. He *et al.* [57] address a yard crane scheduling problem with uncertainty and reduce the completion time.

Some studies propose scheduling yard crane jobs by limiting the job sequence alteration. Briskorn and Angeloudis [58] present a graphical optimization problem to minimize makespan by scheduling the movement of crossover cranes or twin cranes with a fixed job sequence. Briskorn *et al.* [59] include intermediate slot locations for containers being transferred from the quay area to ETs at both ends of the storage area. Their model allows job sequence alteration. Jaehn and Kress [60] and Kress *et al.* [61] schedule twin yard cranes by altering only the sequence of landside jobs. Carlo and Martínez-Acevedo [62] strictly fix the job sequence to find the best priority rule that minimizes the interference between twin yard cranes.

D. RELATIONSHIP BETWEEN OUR PAPER AND OTHER STUDIES

Two published studies relate closely to our research. Park *et al.* [44] develop a twin RMGC scheduling model and consider the option for a RMGC to conduct premarshaling for existing jobs, and Choe *et al.* [45] consider premarshaling for future jobs. Unlike these two studies, we consider premarshaling and remarshaling as well as the realistic restrictions on the time when the OS can start or complete a job based on the job type. We develop a two-stage genetic algorithm for OS job sequencing, considering the benefit by the job separation. To our best knowledge, our study is the first to consider these aspects of OS job sequencing.

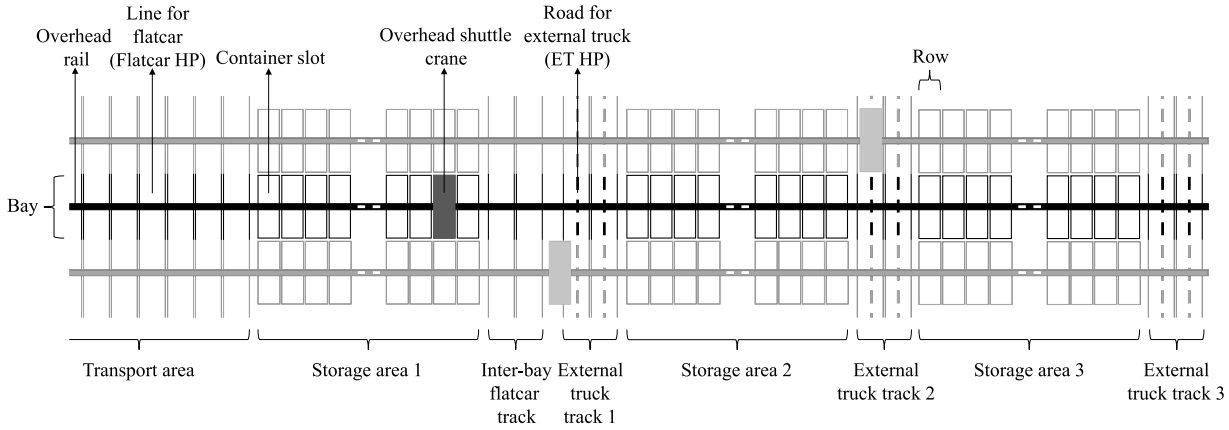

FIGURE 3. The areas covered by an overhead rail in a RACT.

III. PROBLEM DEFINITION

We define a job sequencing problem for an OS to reduce the total tardiness time of flatcars and ETs in a RACT. We define the OS operations, classify them, and introduce a job separation procedure.

A. OS OPERATIONS IN A RACT

We consider a single OS working on an overhead rail covering a part of the transport area and the entire storage yard area as shown in Figure 3. The storage yard area is partitioned into three smaller storage areas each containing a set of container slots. The handover point (HP) is the location where the OS picks up and drops off containers from/to waiting vehicles (flatcars and ETs). Since an OS can interact with a vehicle positioned exactly under the OS rail, the line for the flatcar and the road for the ET also serve as the HP.

The container terminal's OS manager sequences OS jobs to ensure that each OS can serve any vehicle within a certain time window. The process manager regulates the time windows and provides the OS manager with a due time for each OS job to promote synchronization. The flatcar manager provides the OS manager with the expected locations of the interactions between flatcars and the OS, and the expected earliest times of the interactions (earliest arrival time) [8]. In particular, the storage job's earliest time denotes when the OS arrives to pick a container from a vehicle and the retrieval job's earliest time denotes when the OS arrives to drop a container onto a vehicle. Synchronizing the OS and other equipment, considering both of these earliest times, remains challenging due to the inevitable fluctuations in QC and flatcar operations that cause irregularity in the number of OS jobs over time.

B. OS JOB CLASSIFICATION

The OS handles three types of jobs distinguished by their origins and destination locations: retrieval, storage, and auxiliary. For the retrieval job, the origin location is the storage area and the destination location is the HP. The OS

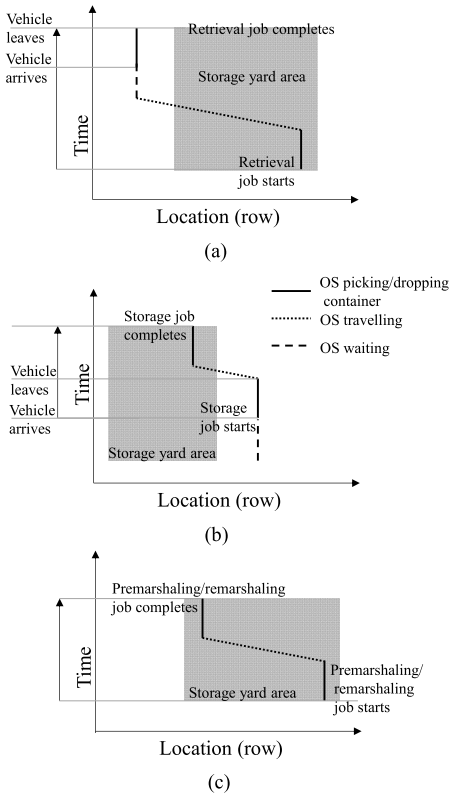

FIGURE 4. Time-space diagrams for three OS jobs: (a) a retrieval job; (b) a storage job; (c) an auxiliary job.

only completes a retrieval job after the corresponding vehicle arrives at the HP location (see Figure 4 (a)). For the storage job, the origin location is the HP and the destination location is the RACT's storage area. The OS only starts a storage job after the corresponding vehicle arrives at the HP location (see Figure 4 (b)).

For the auxiliary jobs, the origin and destination locations are any storage area. Each auxiliary job must have a corresponding main job. The main job must be completed

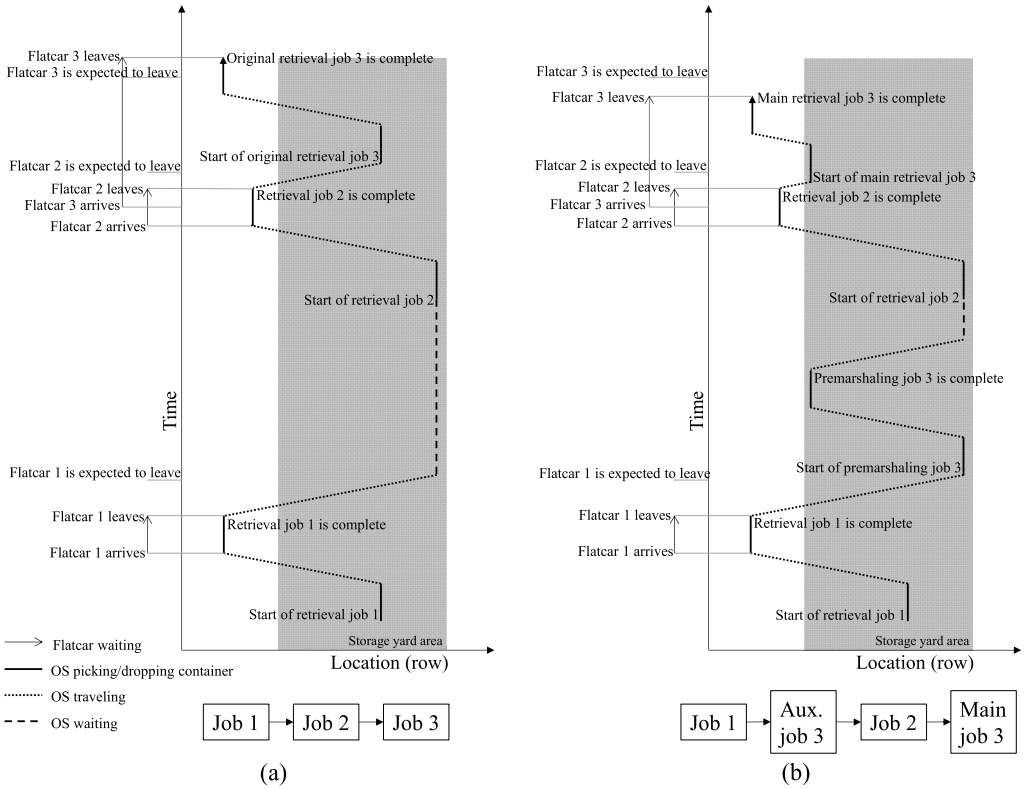

FIGURE 5. Time-space diagram of OS completing three retrieval jobs and block diagram of OS job sequence, before separation of job 3 (a); and after separation of job 3 (b).

before, or can only be started, after the related auxiliary job is completed, i.e., the job having a precedence relationship with the related auxiliary job. Note that the auxiliary job is classified as premarshaling and remarshaling based on the purpose of the job. The OS may start or complete any auxiliary job when the container is in the storage yard area (see Figure 4 (c)).

C. OS JOB SEPARATION

The job separation procedure separates an original job into a main job and an auxiliary job, where the total travel time of an OS when transporting a container (loaded travel time) from the main job and the auxiliary job equals the loaded travel time of the original job. The OS can reduce its workload during busy periods at the expense of adding more workload during idle periods. For instance, Figure 5 shows how premarshaling job 3 in the idle period prevents flatcar 3 from waiting. When the job separation is applied to a retrieval job, the resulting jobs are a retrieval job with shorter loaded travel time as the main job, and a premarshaling job. When the job separation is applied to a storage job, the resulting jobs are a storage job with shorter loaded travel time as the main job, and a remarshaling job. When separating a job does not aid the OS in reducing tardiness, the resulting main and auxiliary jobs are merged back into the original job.

We begin by separating all jobs that can be separable. A job that is eligible for separation has a temporary slot located between the job's origin location and destination location. We assume that the temporary slot is fixed to simplify storage space management. For instance, the origin and destination location of job 3 in Figure 6 encloses the temporary location L_B and therefore, job 3 is separable job. The resulting storage job or retrieval job inherits the arrival time and due time of the original job. We set the arrival time and due time of the resulting premarshaling job or remarshaling job as 0 and M, respectively, with M as a positive large number. We also enforce a precedence relationship for the resulting retrieval job or remarshaling job so that the OS completes the premarshaling job before starting the separated retrieval job, or the storage job before starting the separated remarshaling job. After sequencing all jobs, we merge any separated jobs, whose main and auxiliary jobs are sequenced next to one another in the sequence, back into the original job.

IV. OS JOB SEQUENCING PROBLEM CONSIDERING JOB SEPARATION (OSJSPS)

We formulate the OS job sequencing problem considering job separation (OSJSPS) as an MIP model. Given a set of OS jobs J with $N+1$ jobs, where $J = \{0, 1, 2, \ldots, N\}$, the problem is how to sequence all the jobs in J so that the total tardiness time of all jobs is minimized, with a dummy job $j = 0$ denoting the beginning of the sequence. We also incorporate decision variable w_j to eliminate any subtour in the job sequence [63].

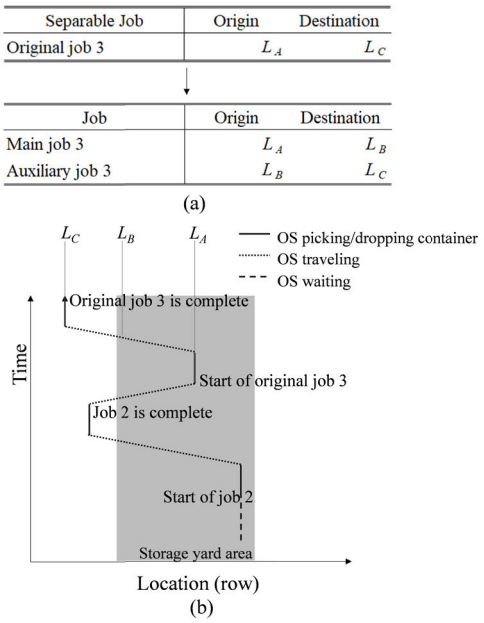

FIGURE 6. **(a) Identification of separable job; (b) Time-space diagram of OS jobs.**

Additionally, we assume that the OS moves at a constant speed and handles containers with a constant picking time regardless of container position in a container slot or vehicle. The sets, parameters, decision variables, and formulation are as follows.

Sets and indexes

J, i,j	The set of jobs J and its indexes $i, j \in J$
S	The set of storage jobs S
R	The set of retrieval jobs R
$G,\{i,j\}$	The set of main jobs and auxiliary jobs G and its indexes $\{i,j\} \in G$

Parameters

N	The number of jobs
MT_{ij}	Empty travel time to the origin location of job j from the last location after processing job i plus the container handling time for picking the container up at the origin location of job j
LT_j	Loaded travel time while processing job j plus the container handling time for dropping the container at the destination location of job j
DT_j	The due time of the vehicle corresponding to job j
AT_j	The arrival time of the vehicle corresponding to job j
PT	Pick-up or drop-off time
M	Positive large number

Decision variables

CR_j	The completion time of the pick-up operation of job j at the origin location
CD_j	The completion time of the drop-off operation of job j at the destination location
D_j	The tardiness time of job j
X_{ij}	$\begin{cases}1, & \text{if OS processes job } j \text{ after job } i\\ 0, & \text{otherwise}\end{cases}$
w_j	Arbitrary real number, j $\in \{0,\ldots,N\}$

$$\text{(OSJSPS) } min\left(\sum\nolimits_{i\in J} D_j\right) \tag{1}$$

subject to

$$\sum\nolimits_{i\in J} X_{ij} = 1, \forall j \in J\backslash\{0\}, \tag{2}$$

$$\sum\nolimits_{j\in J\backslash\{0\}} X_{ij} = 1, \quad \forall i \in J, \tag{3}$$

$$X_{ij} = 0, \quad \forall i \in J, j \in J, i = j, \tag{4}$$

$$w_i - w_j + N \cdot X_{ij} \le N - 1, \quad \forall i \in J\backslash\{0\},\ j \in J\backslash\{0\},\ i \ne j, \tag{5}$$

$$CR_j \ge \begin{cases} CD_i + MT_{ij} - M \cdot (1 - X_{ij}) - PT \cdot X_{ij} \\ \qquad if\ \{i,j\} \in G \\ CD_i + MT_{ij} - M \cdot (1 - X_{ij}) \\ \qquad otherwise \end{cases}$$
$$\forall i \in J\backslash\{0\},\ j \in J\backslash\{0\},\ i \ne j, \tag{6}$$

$$CD_i \ge \begin{cases} CR_i + LT_i - PT \cdot X_{ij} & if\ \{i,j\} \in G \\ CR_i + LT_i & otherwise, \end{cases}$$
$$\forall i \in J\backslash\{0\},\ j \in J\backslash\{0\},\ i \ne j, \tag{7}$$

$$CR_i \ge CD_j, \quad \forall \{i,j\} \in G,\ i \in R, \tag{8}$$

$$CR_j \ge CD_i, \quad \forall \{i,j\} \in G,\ i \in S, \tag{9}$$

$$CR_j \ge AT_j + PT, \quad \forall j \in S, \tag{10}$$

$$CD_j \ge AT_j + PT, \quad \forall j \in R, \tag{11}$$

$$D_j \ge CR_j - DT_j, \quad \forall j \in S, \tag{12}$$

$$D_j \ge CD_j - DT_j, \quad \forall j \in R, \tag{13}$$

$$CR_j, CD_j = 0, \quad j \in \{0\}, \tag{14}$$

$$D_j, CR_j, CD_j \ge 0, \quad \forall j \in J\backslash\{0\}. \tag{15}$$

The objective is to minimize the sum of all jobs' tardiness times (1). Constraints (2) and (3) ensure that each job is sequenced once. Constraints (4) and (5) restrict the sequence to a single closed loop tour for completing the jobs. Constraint (6) updates the completion time for picking a container at the origin location for the job j, and constraint (7) updates the completion time for dropping it at the destination location for the job i. Note that the additional pick-up or drop-off time required by the OS to perform an auxiliary job is negated when the auxiliary job is sequenced before/after its corresponding main job because the resulting main and auxiliary jobs are merged into the original job. Constraint (8) ensures that premarshaling must be completed prior to starting its corresponding retrieval job, and constraint (9) ensures that the remarshaling operation can only be performed after completing its corresponding storage job. Constraint (10) ensures that an OS can only pick a container

from a vehicle after the vehicle arrives, and Constraint (11) ensures that an OS can only drop a container onto a vehicle after the vehicle arrives. Constraints (12) and (13) calculate the tardiness times of all storage and retrieval jobs. Constraint (14) sets job j as the dummy job where the sequence begins. Constraint (15) ensures that all decision variables relating to time must be greater than zero. We validated the OSJSPS model in comparison with its solution and a full-enumeration model (see Appendix A for the details).

V. A TWO-STAGE GENETIC ALGORITHM FOR OSJSPS

A genetic algorithm is a metaheuristic that evolves chromosomes and solves optimization problems based on the principle of natural selection [64]. In scheduling problems, the genetic algorithm changes job sequences as chromosomes of a population. In general, the genetic algorithm generates initial population, evaluates the fitness of each chromosome by the objective function, provides inheritance from parent chromosomes, and decides to terminate if there is no improvement of the best chromosome or the number of repeated steps reaches a maximum iteration [65].

We develop a two-stage genetic algorithm (TGA) based on an iterative local search procedure (iLS) and an opportunistic job separation procedure (OSJSPS). The OSJSPS involves two sequencing problems for main jobs and auxiliary jobs, respectively. We connect the two sequencing problems and use a two-stage algorithm from..Defersha *et al.* [66] and.Tsai and Li [67]. The first stage searches the candidate sequences among the main jobs, and second stage searches the best sequence including auxiliary jobs, given the main job sequence and considering the precedence relationships.

We incorporate local search procedures into the two-stage genetic algorithm because a traditional GA-only algorithm may not guarantee convergence of a global optimal. The GA with the local improvement procedures accelerates searching local optima. We incorporate iLS into the first stage of TGA and an opportunity job separation procedure into the second stage of TGA. The local improvement procedures enable the stages to take advantage of iLS and the opportunistic job separation procedure, thereby improving the initial population the TGA generates. Figure 7 shows the flowchart of TGA.

Section V, part A, describes an initialization of the population based on the iLS and opportunistic job separation procedures. Section V, part B, gives the details of the genetic operators, selection, and crossover. Since the OSJSPS resembles the traveling salesman problem (TSP), it follows the complexity of the TSP which is NP-hard [68].

A. INITIALIZATION OF THE POPULATION

The initial population of the first stage consists of an elite sequence from an iLS and random sequences satisfying a condition that the random sequences should not exceed the maximum makespan from EAT. The initial population in the second stage consists of an elite sequence from the job separation procedure and random sequences from random numbers.

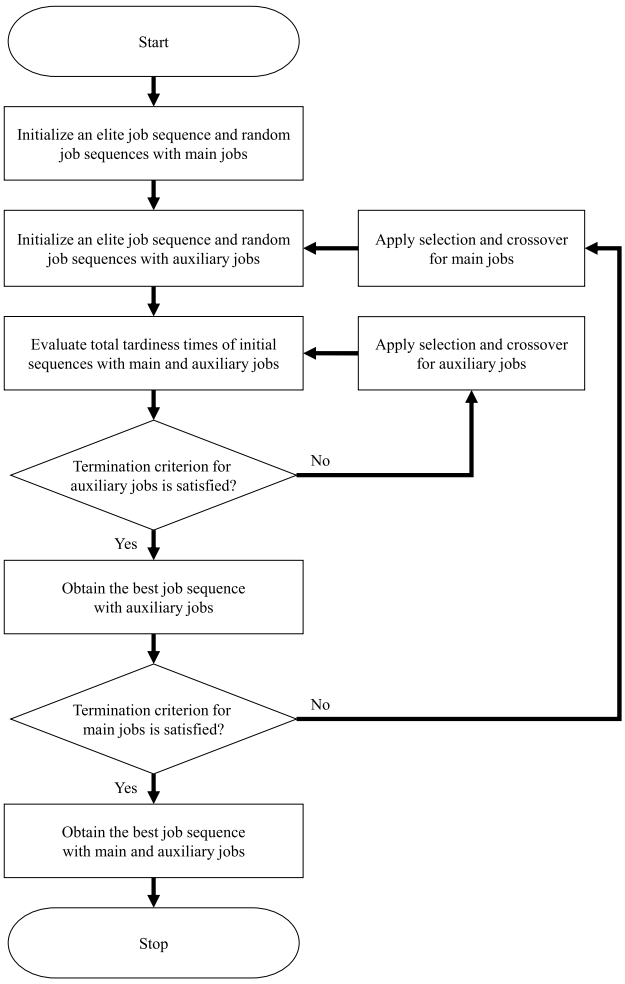

FIGURE 7. Flowchart of TGA.

1) ITERATIVE LOCAL SEARCH PROCEDURE

The iLS starts with a sequence obtained from the earliest-arrival-time procedure (EAT), a common default job sequencing procedure (Ng, 2005) that sequences a set of jobs by sorting them according to job arrival time in non-decreasing order. The iLS iteratively updates the job sequence by using local search sub-procedures. The local search sub-procedures scramble m consecutive jobs in a given sequence starting from the s-th job. For each scramble, the iLS produces a list of $m!$ unique combinations of sequence, selects the best combination that minimizes the total tardiness time of all jobs, and updates the job sequence according to the best combination. An iLS with a larger m value yields better, or at least the same solution as an iLS with a smaller m value, but at exponentially longer computation time. The iterative local search sub-procedure starts with $s = 1$ until $s = N - m$ with an interval of 1 job per iteration. Therefore, the complexity of iLS is approximately $O((N - m + 1) \cdot m!)$.

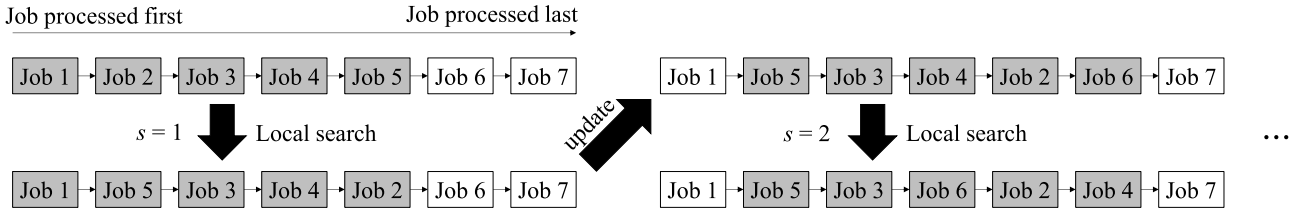

FIGURE 8. Example of the iterative local search procedure with m = 5.

Algorithm 1 summarizes the iLS procedure and Figure 8 illustrates the steps.

Algorithm 1 Iterative Local Search Procedure

Step 1.	Apply EAT to a given set of jobs J.
Step 2.	Start the iteration of the local search.
Step 2.1	Set s as 1.
Step 2.2	Create a job set V representing the permutations of the s-th job to $(s+m-1)$-th job in J.
Step 2.3	Calculate the total tardiness of each job set in V.
Step 2.4	Replace job set J with the job set with the lowest total tardiness in V, and delete V.
Step 2.5	If $s+m$ is smaller than N, set s as $s+1$ and go to *Step* 2.2; otherwise, continue to *Step* 3.
Step 3.	Return J.

2) OPPORTUNISTIC JOB SEPARATION PROCEDURE

The opportunistic job separation procedure aims to reduce the total tardiness of all jobs by simultaneously selecting which job should be separated and determining when the OS processes the resulting auxiliary job. The procedure begins by identifying all separable jobs and time slots. Then the procedure assigns each of the resulting auxiliary jobs from the selected separable job to a time slot that minimizes the total tardiness of all jobs. We define a time slot as the period when the OS is idle. All separable jobs (original job) j are identified based on the criterion described in Section III, part B. We use index $j+N$ for the resulting main job and $j+2N$ for the resulting auxiliary job. At this point, we have three sets of jobs: a set of original jobs $J \subseteq \{1,\ldots,N\}$, a set of main jobs $K \subseteq \{N+1,\ldots,2N\}$, and a set of auxiliary jobs $L \subseteq \{2N+1,\ldots,3N\}$, where the separation of job j results in the main job $j+N$ and an auxiliary job $j+2N$. We consider the elements with indexes between $N+1$ to $2N$ as separable jobs.

Identifying the time slot requires the job sequence information and an executable OSJSPS model. Figure 9 shows the OSJSPS model execution, in which we can track the OS status over time, and identify any time slot (constraint (6) and constraint (7)). The procedure only considers time slots with durations longer than the total duration for the OS to pick and drop a container. The procedure marks each time slot with the job index that succeeds in the corresponding time slot (constraint (8) and constraint (9)). For instance, if there is a time slot before the OS completes job j, the time slot is identified as time slot j. This example suggests that the OS can process an auxiliary job before it starts job j. Preliminary simulations showed that most of the time slots had short durations, so we set 1 as the number of auxiliary jobs that can be conducted during a time slot.

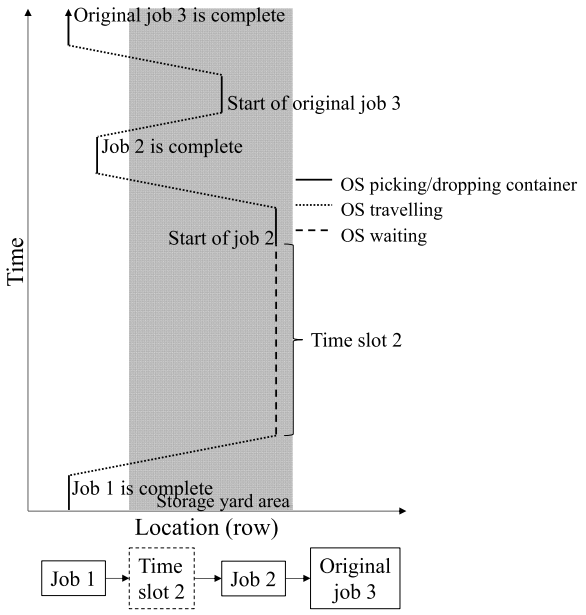

FIGURE 9. Identification of a time slot.

We define a unique time slot for each auxiliary job that can only be taken by and serves as the default time slot for, the corresponding auxiliary job. We set the default time slot immediately before or after the respective storage or retrieval job, depending on the precedence relationship. Therefore, when job separation brings no benefit, the auxiliary job will take the default time slot and it is merged back into the original job.

After identifying all time slots and all separable jobs, the procedure replaces all separable jobs j (original job) with the resulting storage or retrieval job (main job) $j+N$ and an auxiliary job $j+2N$. To guide the assignment of each auxiliary job to a time slot, the procedure iteratively selects the best time slot for each auxiliary job, which reduces the total tardiness time. In the case of equivalent total tardiness times, the procedure selects the auxiliary job that requires the longest loaded travel time. In the case of yet another equivalent loaded travel time, the procedure randomly selects from the pairs.

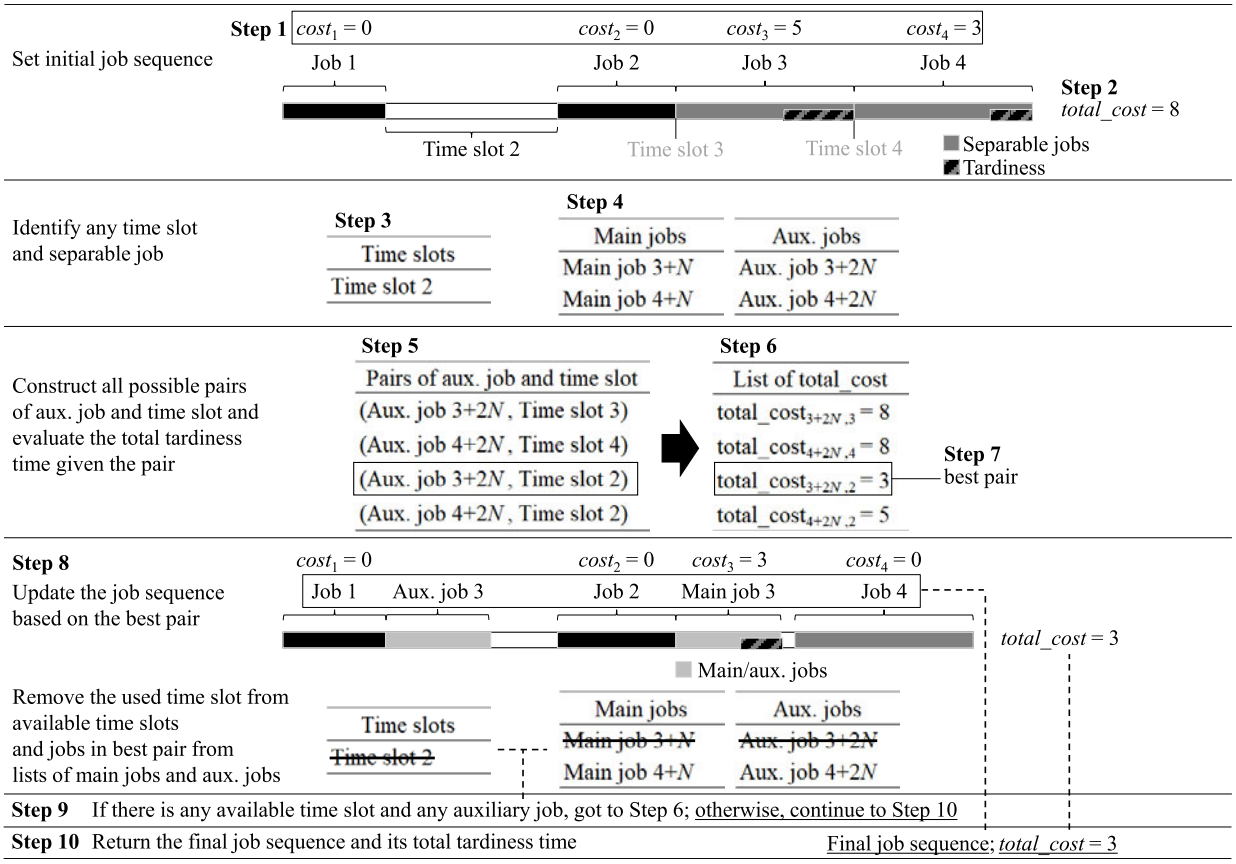

FIGURE 10. Numerical example of the opportunistic job separation procedure.

The procedure maintains the sequence of original jobs and main jobs, and only changes when it inserts an auxiliary job between the original job and the main jobs. Obviously, the sequence of the given set of jobs J influences the procedure's effectiveness. Algorithm 2 summarizes the procedure and Figure 10 illustrates a numerical example.

Algorithm 2 also shows that the procedure has a complexity of $O(N^2+(N-1)^2+\ldots+12) = O(N\cdot(N+1)(2N+1)/6))$ for the most complex case, i.e., all jobs are separable and there is a one slot between each job.

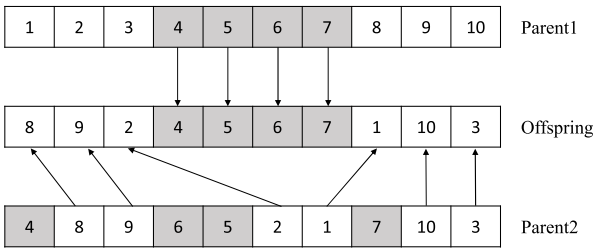

FIGURE 11. Crossover procedure for first stage (modified from Pinedo, 2011).

B. SELECTION AND CROSSOVER OPERATORS

We introduce selection and crossover operators for the job scheduling problem. The genetic operators select parents and generate new offspring. We use two selection and crossover operators for the main jobs and auxiliary jobs. The genetic operators repeat until each termination criterion is satisfied, i.e., the fitness becomes zero or the number of iterations reaches the maximum iteration.

1) LINEAR ORDER CROSSOVER

We adopt the popular crossover operator, linear order crossover (LOX) mentioned in.Pinedo [69] for searching main job sequences. It generates offspring from two random crossover points which can be applied to a single machine scheduling problem. The numbers in Figure 11 denote the IDs of main jobs with their sequences. A gene represents a job ID with its order and a chromosome represents the set of genes. The genes between two random crossover points are inherited from the first parent and other genes are copied from the second parent.

2) TWO-POINT CROSSOVER

We also adopt the two-point crossover operator in Murata *et al.* [70] for auxiliary job sequences. We define the feasible solution space because the auxiliary job corresponding to a storage job should be placed before (forward) the main job, i.e., the auxiliary job to a retrieval job should be placed after (backward) the main job. The numbers in Figure 12 denote the offsets from a random number, where the number of steps the auxiliary job are located forward

Algorithm 2 Opportunistic Job Separation Procedure

Step 1.	Given a set of sequenced jobs J, calculate $cost_j$ as the tardiness of each job $j \in J$.
Step 2.	Calculate *total_cost* as the total tardiness time, given a set of sequenced jobs J.
Step 3.	Identify and list any time slot z into a set of time slots Z.
Step 4.	Identify any separable job $j \in J$, and list its corresponding main job $j+N$ to K and its corresponding auxiliary job $j+2N$ to L.
Step 5.	Construct all possible pairs of auxiliary jobs $j+2N \in L$ and time slots $z \in Z$.
Step 6.	Calculate the $total\_cost_{j+2N,z}$ for each pair of auxiliary job $j+2N$ and time slot z, as the total tardiness time of all jobs in J, except that job j is replaced by main job $j+N$ and auxiliary job $j+2N$ is included in time slot z.
Step 7.	If there is a $total\ cost_{j+2N,z}$ with smaller value than *total_cost*, select the pair of auxiliary job $j+2N$ and time slot z with the minimum $total\_cost_{j+2N,z}$; if more than one pair has an equal minimum value, select the one with the longest loaded travel time, if there are still pairs with equal solutions, then select randomly, then continue to *Step* 8; otherwise, go to *Step* 10.
Step 8.	Update J by replacing the selected job j with $j+N$ and putting $j+2N$ into time slot z, and then remove $j+N$ from K, $j+2N$ from L, and z from Z.
Step 9.	If there is any auxiliary job in L and any time slot z in Z, go to *Step* 6; otherwise, continue to *Step* 10.
Step 10.	Calculate the *total_cost* as the total tardiness of J, and return J and the *total_cost*.

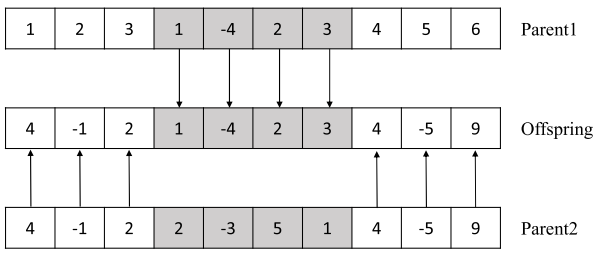

FIGURE 12. Crossover procedure for second stage (modified from Murata, 1996).

or backward within a range of the feasible solution space. If more than one auxiliary job is assigned to the same position, the auxiliary jobs are re-sequenced randomly.

VI. COMPUTATIONAL EXPERIMENT

In this section we evaluate the proposed heuristics under three scenarios where job separation is possible. We modify data from a real container terminal to conduct a case study, considering the inter-arrival times of flatcars and external trucks. We compare the results obtained by the proposed heuristics and by the optimal approach.

A. EXPERIMENT DESIGN

We design an experiment to evaluate the effectiveness of the proposed heuristics for reducing total tardiness time and the effects of different scenarios on each heuristic, and to measure the computation required by each heuristic. Table 1 lists the parameters in the experiment.

The real-size terminal has a single OS serving a partial area of 260.5 meters in length in a bay (shaded area in Figure 13). To enable job separation, we consider five temporary slots adjacent to each other, and randomly choose each slot during job separation. We set the length of the bay and OS moving times based on the simulation in [1]. We assume a constant OS speed of 4 m/s and a constant pick-up/drop-off time of 1 minute regardless of container location in a slot or vehicle. This assumption is reasonable for the fast-vertical speed of the OS, so we represent the speed by an average. Given the OS speed and moving times and the job's origin and destination locations, we calculate the empty travel timetable and loaded travel time before starting the job sequencing procedure.

The parameters, which represent common practical situations, are the numbers of jobs typically considered during a job sequencing period, the job compositions, and the distribution of flatcar inter-arrival times. The job compositions parameter represents different stages of QC activities in handling a vessel. For example, when QCs unload containers from a vessel, storage jobs from flatcars predominate, and when QCs load containers to a vessel, retrieval jobs predominate. The distribution flatcar inter-arrival time represents the workload distribution for a given vessel across multiple bays. For example, when a vessel's workload is concentrated in a bay, the frequency of arriving flatcars increases, and when the workload is sparsely distributed, the frequency of arriving flatcars decreases.

In Table 1 the numbers in bold represent the default setting. Typically, a bay handles 5 to 8 jobs per hour. For larger-scale problems, we consider up to 25 jobs during a single sequencing procedure. The flatcar inter-arrival time is represented by the minimum and maximum values of the inter-arrival times between the flatcars and they follow a uniform distribution. The ET inter-arrival time is represented by the uniform distribution with a minimum value of 0 and a maximum value of the last ET's arrival time. The job composition assumes a fixed percentage (10%) of the storage job and retrieval job to/from the ET. Our assumption of no rehandling is reasonable since 80% of the real jobs relate to vessels where rehandling can be prevented by careful planning. We assume the same priority between a flatcar and an external truck to prevent obtaining biased tardiness times by vehicle types. See Appendix B for the details.

TABLE 1. Summary of parameters in the experiment.

Parameters	Parameters
Algorithms	EAT, EAT-S, iLS, iLS-S, TGA, OPT
Number of jobs	**10,** 15, 20, 25
Flatcar inter-arrival time [min, max]	[30, 360], **[30, 420]**, [30, 480]
Job composition in percentage [Storage from flatcar (S), Retrieval from flatcar (R)]	**[70, 10]**, [40, 40], [10, 70]

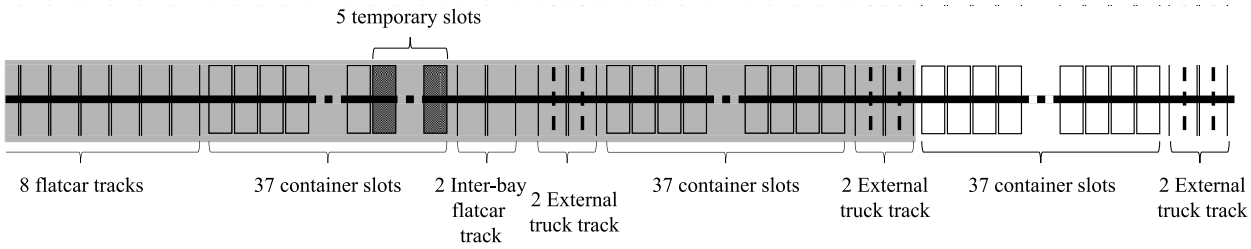

FIGURE 13. Schematic layout of a bay [1].

We test five heuristics (earliest-arrival-time heuristic (EAT), earliest-arrival time with opportunistic job separation heuristic (EAT-S), iterative local search heuristic (iLS), iterative local search with opportunistic job separation heuristic (iLS-S), two-stage genetic algorithm (TGA), and an optimal algorithm (OPT). Only the EAT and the iLS apply the job sequencing procedure without any job separation procedure. The EAT-S uses an earliest-arrival time procedure to sequence the jobs and then uses the opportunistic job separation procedure for the resulting sequence. The iLS-S uses the iLS procedure combined with the job separation procedure to search for the sequence that minimizes total tardiness time. The TGA may search for further improvement from the local optima and obtain the global optimal. We also use an MIP solver to obtain the OPT solutions from the formulations.

Since we apply the opportunistic job separation procedure after finding the best job sequence, we compute the total tardiness time of each job sequence during the iterations without considering the improvement by job separation. For the iLS-based heuristic, we use the sub-procedures scramble $m = 5$. To test TGA, we use the genetic parameters and set the first stage population and sub-population sizes as 5 times the number of jobs. We set the maximum iterations for the first stage and second stage as a fixed number 25 after preliminary experiments.

We measure the total tardiness time obtained by each heuristic and the required CPU time. We compare the solutions obtained by the heuristics to the optimal solution using small-scale problems.

We use Siemens Tecnomatix Plant Simulation version 12 as our simulation platform and SimTalk, the built-in programming language of the Plant Simulation software, to implement the heuristics. We use Visual Studio 2017 to construct the model and IBM ILOG CPlex version 12.6 C++ Callable library to solve it optimally. We perform 20 trials for each parameter setting. We run the experiment on a PC equipped with Windows 10 Pro and Intel Core i5-7600L @3.80 GHz and 16GB RAM.

B. EXPERIMENTAL RESULTS

Table 2 compares OSJSP, which allows no job separation, to OSJSPS using OPT. The job separation allowance in OSJSPS reduces total tardiness time by 39.09% on average. Job separation increases both computation time and complexity.

TABLE 2. Comparison between OSJSP and OSJSPS.

Number of jobs	Total tardiness time (s)			CPU Time (s)	
	OSJSP	OSJSPS	Gap(%)	OSJSP	OSJSPS
6	14.93	10.28	45.16	0.03	0.05
7	19.06	12.99	46.71	0.03	0.07
8	21.76	15.69	38.67	0.04	0.09
9	35.70	25.22	41.55	0.04	0.18
10	55.68	43.21	28.84	0.07	1.06
11	59.88	45.55	31.46	0.09	1.51
12	63.56	47.37	34.18	0.11	1.76
13	73.54	54.41	35.14	0.14	4.70
14	79.47	54.76	45.11	0.15	7.32
15	102.21	70.93	44.09	0.17	917.08

Table 3 summarizes the experimental results for the small problem sizes. The TGA outperforms the other heuristics which are still trapped at local optimal solutions. When the number of jobs is small enough, e.g., 6 to 8, the TGA obtains optimal solutions. When the number of jobs is large enough, e.g., 9 to 15, the TGA's optimality gap is between 5.90% and 9.74%. The CPlex solver is unable to solve problems with 15 or more jobs within the time allowance of 28800 seconds.

Figure 14 illustrates the effectiveness of the heuristics algorithms relative to the EAT rule. On average, EAT-S and iLS-S improve 11.06% and 56.56%, respectively, and TGA and OPT improve 66.24% and 68.37% in total tardiness time, respectively.

TABLE 3. Experimental results for small problem sizes.

Number of Jobs	Total tardiness time (s)						CPU Time (s)					
	EAT	EAT-S	iLS	iLS-S	TGA	OPT	EAT	EAT-S	iLS	iLS-S	TGA	OPT
6	24.02	21.85	9.61	9.61	3.26	3.26	0.00	0.00	0.07	0.07	0.13	0.04
7	27.97	24.97	11.21	11.21	4.39	4.39	0.00	0.00	0.15	0.15	0.31	0.07
8	42.56	39.56	21.32	20.09	12.78	12.78	0.00	0.00	0.25	0.25	0.61	0.09
9	97.84	94.84	46.50	45.28	40.57	36.97	0.00	0.00	0.36	0.37	0.86	0.22
10	130.86	121.86	66.48	64.23	60.25	54.96	0.00	0.00	0.51	0.51	1.23	1.12
11	144.72	127.10	73.92	67.69	62.44	57.30	0.00	0.01	0.66	0.66	1.59	1.60
12	151.92	128.80	74.89	66.44	64.24	59.12	0.00	0.01	0.84	0.84	2.39	2.29
13	193.85	162.71	91.98	78.50	71.23	66.18	0.00	0.01	1.04	1.05	3.97	5.78
14	214.01	180.99	98.37	81.33	70.46	66.53	0.00	0.01	1.25	1.27	9.21	8.80
15	247.34	210.35	124.45	105.52	88.32	82.70	0.00	0.01	1.48	1.51	12.18	919.20

TABLE 4. Experimental results for various job compositions.

Number of Jobs	Job composition (%) [S, R]	Total tardiness time (s)					CPU Time (s)				
		EAT	EAT-S	iLS	iLS-S	TGA	EAT	EAT-S	iLS	iLS-S	TGA
10	[70, 10]	98.82	89.81	55.68	53.43	48.50	0.00	0.00	0.45	0.46	1.06
	[40, 40]	123.89	108.84	67.58	60.90	52.86	0.00	0.00	0.46	0.46	1.68
	[10, 70]	225.74	183.70	147.98	114.55	111.81	0.00	0.00	0.47	0.47	3.83
15	[70, 10]	185.00	150.37	102.21	85.64	76.56	0.00	0.01	1.36	1.36	11.26
	[40, 40]	284.47	206.78	147.82	99.14	91.41	0.00	0.01	1.37	1.38	13.52
	[10, 70]	521.00	409.17	327.49	251.52	243.26	0.00	0.01	1.39	1.40	19.77
20	[70, 10]	318.30	268.47	215.57	188.23	170.73	0.00	0.03	2.78	2.80	52.02
	[40, 40]	470.10	368.46	258.17	191.11	179.29	0.00	0.03	2.81	2.83	78.12
	[10, 70]	769.58	598.88	505.42	407.82	399.61	0.00	0.02	2.84	2.86	85.41
25	[70, 10]	373.93	303.66	263.44	222.23	213.50	0.00	0.07	4.79	4.85	116.57
	[40, 40]	579.44	435.67	320.47	238.25	226.03	0.00	0.07	4.84	4.90	159.21
	[10, 70]	936.28	742.46	578.57	472.49	469.76	0.00	0.05	4.90	4.96	148.81

TABLE 5. Experimental results for various flatcar inter-arrival times.

Number of Jobs	Flatcar Inter-arrival Time (s) [min, max]	Total tardiness time (s)					CPU Time (s)				
		EAT	EAT-S	iLS	iLS-S	TGA	EAT	EAT-S	iLS	iLS-S	TGA
10	[30, 360]	130.56	121.88	76.83	72.36	59.90	0.00	0.00	0.46	0.46	2.26
	[30, 420]	98.82	89.81	55.68	53.43	48.50	0.00	0.00	0.46	0.46	1.05
	[30, 480]	77.90	72.93	40.23	38.24	36.64	0.00	0.00	0.46	0.46	0.60
15	[30, 360]	316.20	272.88	192.44	176.97	143.38	0.00	0.01	1.35	1.35	15.42
	[30, 420]	185.00	150.37	102.21	85.64	76.56	0.00	0.01	1.35	1.36	11.08
	[30, 480]	134.41	109.27	70.58	63.33	58.14	0.00	0.01	1.36	1.37	5.13
20	[30, 360]	544.05	487.15	363.68	343.66	304.68	0.00	0.02	2.75	2.76	69.01
	[30, 420]	318.30	268.47	215.57	188.23	170.73	0.00	0.03	2.76	2.78	51.09
	[30, 480]	221.64	179.22	146.55	122.57	116.17	0.00	0.04	2.77	2.80	44.32
25	[30, 360]	699.02	603.77	469.30	429.33	396.98	0.00	0.05	4.71	4.77	169.32
	[30, 420]	373.93	303.66	263.44	222.23	213.50	0.00	0.08	4.75	4.82	114.51
	[30, 480]	266.98	213.46	183.75	153.12	148.84	0.00	0.09	4.76	4.86	93.49

Table 4 lists the results for the various job compositions. The job separation procedure contributes to the reduction of total tardiness time as indicated by the relative improvement by EAT-S, ranging from 9.11% to 27.31%, the relative improvement by iLS-S, ranging from 29.55% to 65.15%, and the relative improvement by TGA, ranging from 42.90% to 67.86%. The reductions in total tardiness time by job separation persist throughout all job composition settings. Job separation particularly achieves high improvement when there is a balanced ratio between storage and retrieval jobs to/from flatcars; the average improvement for the iLS-S and TGA is 51.66% and 54.26%, respectively.

Table 5 lists the results for the various distributions of the flatcar arrival time periods. We note that the job separation procedure is only effective if the average flatcar inter-arrival time is at least longer than or equal to 195 seconds. After a certain point where the average flatcar inter-arrival time period increases, the improvement by job separation on iLS

TABLE 6. Experimental results of a case study*.

Number of Jobs	Total tardiness time (s)						CPU Time (s)					
	EAT	EAT-S	iLS	iLS-S	TGA	OPT	EAT	EAT-S	iLS	iLS-S	TGA	OPT
6	249.58	242.92	195.31	192.34	166.90	139.20	0.00	0.00	0.06	0.06	0.88	0.68
7	411.07	396.06	321.23	313.27	262.66	231.05	0.00	0.00	0.13	0.13	2.66	35.69
8	582.01	565.98	428.69	422.62	353.11	326.75	0.00	0.00	0.23	0.23	4.21	1522.43
9	775.27	758.16	588.71	580.19	496.08	-	0.00	0.00	0.33	0.34	6.33	-
10	1058.97	1036.79	799.02	782.55	678.78	-	0.00	0.00	0.46	0.46	8.79	-
11	1329.94	1300.59	998.90	976.17	847.95	-	0.00	0.01	0.60	0.60	10.44	-
12	1610.29	1550.52	1253.74	1211.88	1067.88	-	0.00	0.01	0.76	0.76	14.17	-
13	2011.46	1941.51	1567.66	1526.69	1367.88	-	0.00	0.01	0.95	0.95	17.87	-
14	2268.54	2189.31	1693.16	1651.30	1511.26	-	0.00	0.01	1.15	1.15	20.51	-
15	2592.79	2520.77	1949.33	1896.97	1736.32	-	0.00	0.01	1.36	1.37	24.36	-

'-' OSJSPS fails to generate a solution within 28800s.
* The datasets are available from the authors upon request.

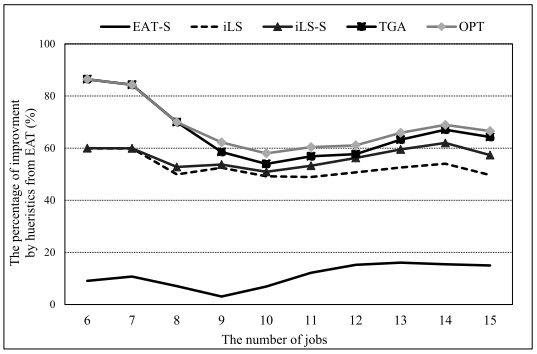

FIGURE 14. The percentage of improvement of heuristics in total tardiness time to EAT for small problem sizes.

and TGA heuristics starts to decrease. Ultimately, when the flatcar inter-arrival time is long enough so that each flatcar experiences no tardiness, job separation becomes obsolete. We note that the computation time required by TGA is heavily influenced by the average flatcar inter-arrival time. As the average of flatcar inter-arrival time becomes longer, the TGA quickly converges to the local or global optimal. When the number of jobs increases in iLS-S and EAT-S, computation times also increase, e.g., 5 to 8 jobs arrive every hour in a real bay of a RACT.

The experimental results show that the proposed heuristics reduce the total tardiness time of vehicles under circumstances representing the QC's different stages: handling vessels or vehicles, and bay workloads. The iterative local search within the iLS-S limits the search space of the possible job sequence enough to obtain a local optimum within a short time. The TGA reduces the total tardiness time of the local search procedures in most scenarios in the experiments. Overall, iLS-based heuristics outperform EAT-based heuristics under any circumstances, partly because when iLS-based heuristics re-sequence a subset of neighboring jobs, the total tardiness time of a series of jobs decreases, but at the expense of a slight increase in the tardiness time of one job.

C. CASE STUDY

We consider a case study with the inter-arrival time of flatcars and external trucks modified for the RACT configuration from real data. We assume that the due time of vehicles follows a lognormal distribution from the vehicle inter-arrival time data. Table 6 reports that the TGA outperforms the algorithms that are trapped in local optimal solutions. Figure 15 shows the relative improvement by TGA, ranging from 32.01% to 39.33%.

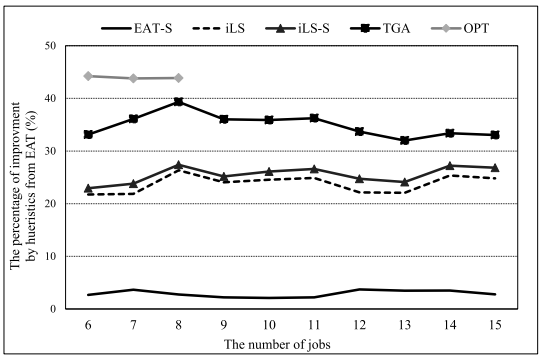

FIGURE 15. The percentage of improvement of heuristics in total tardiness time to EAT in the case study.

VII. CONCLUSION

This article presented a job sequencing problem, considering job separation, and developed a two-stage genetic algorithm based on an iterative local search procedure and an opportunistic job separation procedure for sequencing a set of OS jobs in a RACT with a single OS traveling on a rail in a bay. The proposed algorithms enhanced the flexibility of OS operations and smoothed the synchronization procedure between yard storage and vehicles in the RACT.

The iterative local search procedure was developed based on the notion that an efficient sequence is obtainable by scrambling only a small subset of jobs with similar arrival times. The opportunistic job separation procedure improved the sequence by exploiting OS idle time. The two-stage

genetic algorithm incorporated local search procedures to further improve total tardiness time within reasonable calculation time.

The four heuristics, EAT-S, iLS, iLS-S, and TGA, include the iterative local search procedure, or opportunistic job separation procedure, or both. The EAT-S heuristic applies the opportunistic job separation procedure after the jobs are sequenced. The iLS uses the iterative search procedure without job separation. The iLS-S, an extension of the iLS, adds the opportunistic job separation procedure after the jobs are sequenced. The TGA applies an opportunistic job separation procedure during iterative local search procedures.

Future research should address the rehandling job problem when the requested containers are not at the top of the container stack. The impacts of decisions about temporary slot locations on the four heuristics deserve more examination. Future research should also consider integrating the flatcar routing problem into the OS job sequencing problem to obtain smoother synchronization, by modifying the proposed heuristics to address the different equipment configurations in RACTs. The proposed algorithm and procedures could also be modified to obtain smoother synchronization between yard cranes and waiting vehicles in vertical terminals with transport locations and yard cranes under Chebyshev movement.

APPENDIX

A. THE OSJSPS VALIDATION

We validate the OSJSPS model in comparison a with full-enumeration model (Solution) The full-enumeration model uses 30 instances. We enumerate full cases and obtain an optimal solution, which means the OSJSPS model works correctly.

TABLE 7. Experiment results for a comparison with OSJSPS and full-enumeration.

Instance	The total tardiness time		Instance	The total tardiness time	
	OSJSPS	Solution		OSJSPS	Solution
1	48.90	48.90	16	9.40	9.40
2	13.70	13.70	17	14.79	14.79
3	4.23	4.23	18	4.14	4.14
4	45.57	45.57	19	52.89	52.89
5	211.00	211.00	20	10.16	10.16
6	23.13	23.13	21	12.52	12.52
7	18.46	18.46	22	289.78	289.78
8	69.97	69.97	23	152.06	152.06
9	25.60	25.60	24	13.28	13.28
10	34.44	34.44	25	1.76	1.76
11	164.94	164.94	26	17.48	17.48
12	140.80	140.80	27	63.99	63.99
13	17.29	17.29	28	36.49	36.49
14	63.22	63.22	29	172.35	172.35
15	10.82	10.82	30	80.01	80.01

B. EXPERIMENTAL RESULTS FROM WEIGHTS BETWEEN A FLATCAR AND EXTERNAL TRUCK

We conduct sensitivity analysis over weights by vehicle type. Table 8 reports that the weights modulate the tardiness of flatcars, external trucks, and the total sum of tardiness. In general, the weight [50, 50] are balanced.

TABLE 8. Experimental results for various priorities by vehicle type.

Truck Composition [Flatcar, External truck]	Truck weight	Total tardiness time (s)		
		Flatcar	External truck	Total
[80, 20]	[10, 90]	73.19	8.06	81.25
	[30, 70]	50.53	10.94	61.47
	[50, 50]	49.75	10.94	60.69
	[70, 30]	19.08	40.48	59.56
	[90, 10]	6.23	93.28	99.51
[50, 50]	[10, 90]	672.65	113.82	786.47
	[30, 70]	521.19	137.77	658.96
	[50, 50]	317.99	289.31	607.30
	[70, 30]	107.02	660.26	767.27
	[90, 10]	37.27	918.81	956.09

REFERENCES

[1] J. H. Seo, S. Yi, and K. H. Kim, "Dispatching vehicles in a rail-based container terminal," *J. Korean Inst. Ind. Eng.*, vol. 43, no. 6, pp. 464–473, Dec. 2017.

[2] H.-O. Günther and K.-H. Kim, "Container terminals and terminal operations," *OR Spectr.*, vol. 28, no. 4, pp. 437–445, Oct. 2006.

[3] K. H. Kim, M.-H.-T. Phan, and Y. J. Woo, "New conceptual handling systems in container terminals," *Ind. Eng. Manage. Syst.*, vol. 11, no. 4, pp. 299–309, Dec. 2012.

[4] B. Dragović, E. Tzannatos, and N. K. Park, "Simulation modelling in ports and container terminals: Literature overview and analysis by research field, application area and tool," *Flexible Services Manuf. J.*, vol. 29, no. 1, pp. 4–34, Mar. 2017.

[5] M. E. H. Petering, "Decision support for yard capacity, fleet composition, truck substitutability, and scalability issues at seaport container terminals," *Transp. Res. E, Logistics Transp. Rev.*, vol. 47, no. 1, pp. 85–103, Jan. 2011.

[6] C. Zhou, E. P. Chew, L. H. Lee, and D. Liu, "An introduction and performance evaluation of the GRID system for transshipment terminals," *Simulation*, vol. 92, no. 3, pp. 277–293, Mar. 2016.

[7] L. Zhen, L. H. Lee, E. P. Chew, D.-F. Chang, and Z.-X. Xu, "A comparative study on two types of automated container terminal systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 1, pp. 56–69, Jan. 2012.

[8] I. K. Singgih, S. Hong, and K. H. Kim, "Flow path design for automated transport systems in container terminals considering traffic congestion," *Ind. Eng. Manage. Syst.*, vol. 15, no. 1, pp. 19–31, Mar. 2016.

[9] M.-H. Phan-Thi, K. Ryu, and K. H. Kim, "Comparing cycle times of advanced quay cranes in container terminals," *Ind. Eng. Manage. Syst.*, vol. 12, no. 4, pp. 359–367, Dec. 2013.

[10] K. H. Kim and H. B. Kim, "The optimal sizing of the storage space and handling facilities for import containers," *Transp. Res. B, Methodol.*, vol. 36, no. 9, pp. 821–835, Nov. 2002.

[11] J. Wiese, L. Suhl, and N. Kliewer, "Planning container terminal layouts considering equipment types and storage block design," in *Handbook Terminal Planning*. New York, NY, USA: Springer, 2011, pp. 219–245.

[12] I. F. A. Vis, "A comparative analysis of storage and retrieval equipment at a container terminal," *Int. J. Prod. Econ.*, vol. 103, no. 2, pp. 680–693, Oct. 2006.

[13] X. Guo, S. Y. Huang, W. J. Hsu, and M. Y. H. Low, "Yard crane dispatching based on real time data driven simulation for container terminals," in *Proc. Winter Simulation Conf.*, Dec. 2008, pp. 2648–2655.

[14] H. Lu and S. Wang, "A study on multi-ASC scheduling method of automated container terminals based on graph theory," *Comput. Ind. Eng.*, vol. 129, pp. 404–416, Mar. 2019.

[15] R. Stahlbock and S. Voß, "Efficiency considerations for sequencing and scheduling of double-rail-mounted gantry cranes at maritime container terminals," *Int. J. Shipping Transp. Logistics*, vol. 2, no. 1, pp. 95–123, 2010.

[16] B. Wang, W. Chen, B. Zhang, and Y. Zhao, "Regulation cooperative control for heterogeneous uncertain chaotic systems with time delay: A synchronization errors estimation framework," *Automatica*, vol. 108, Oct. 2019, Art. no. 108486.

[17] C. Deng, M. J. Er, G.-H. Yang, and N. Wang, "Event-triggered consensus of linear multiagent systems with time-varying communication delays," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 2916–2925, Jul. 2020.
[18] H. J. Carlo, I. F. A. Vis, and K. J. Roodbergen, "Seaside operations in container terminals: Literature overview, trends, and research directions," *Flexible Services Manuf. J.*, vol. 27, nos. 2–3, pp. 224–262, Sep. 2015.
[19] C. Bierwirth and F. Meisel, "A follow-up survey of berth allocation and quay crane scheduling problems in container terminals," *Eur. J. Oper. Res.*, vol. 244, no. 3, pp. 675–689, Aug. 2015.
[20] R. Stahlbock and S. Voß, "Operations research at container terminals: A literature update," *OR Spectr.*, vol. 30, no. 1, pp. 1–52, Nov. 2007.
[21] C. Bierwirth and F. Meisel, "A survey of berth allocation and quay crane scheduling problems in container terminals," *Eur. J. Oper. Res.*, vol. 202, no. 3, pp. 615–627, May 2010.
[22] H. Y. K. Lau and Y. Zhao, "Integrated scheduling of handling equipment at automated container terminals," *Int. J. Prod. Econ.*, vol. 112, no. 2, pp. 665–682, Apr. 2008.
[23] J. X. Cao, D.-H. Lee, J. H. Chen, and Q. Shi, "The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods," *Transp. Res. E, Logistics Transp. Rev.*, vol. 46, no. 3, pp. 344–353, May 2010.
[24] S. M. Homayouni, S. H. Tang, and O. Motlagh, "A genetic algorithm for optimization of integrated scheduling of cranes, vehicles, and storage platforms at automated container terminals," *J. Comput. Appl. Math.*, vol. 270, pp. 545–556, Nov. 2014.
[25] J. He, Y. Huang, W. Yan, and S. Wang, "Integrated internal truck, yard crane and quay crane scheduling in a container terminal considering energy consumption," *Expert Syst. Appl.*, vol. 42, no. 5, pp. 2464–2487, Apr. 2015.
[26] H. Dkhil, A. Yassine, and H. Chabchoub, "Multi-objective optimization of the integrated problem of location assignment and straddle carrier scheduling in maritime container terminal at import," *J. Oper. Res. Soc.*, vol. 69, no. 2, pp. 247–269, Feb. 2018.
[27] D. Kress, S. Meiswinkel, and E. Pesch, "The partitioning min-max weighted matching problem," *Eur. J. Oper. Res.*, vol. 247, no. 3, pp. 745–754, 2015.
[28] X. Chen, S. He, Y. Zhang, L. Tong, P. Shang, and X. Zhou, "Yard crane and AGV scheduling in automated container terminal: A multi-robot task allocation framework," *Transp. Res. C, Emerg. Technol.*, vol. 114, pp. 241–271, May 2020.
[29] H. J. Carlo, I. F. A. Vis, and K. J. Roodbergen, "Transport operations in container terminals: Literature overview, trends, research directions and classification scheme," *Eur. J. Oper. Res.*, vol. 236, no. 1, pp. 1–13, Jul. 2014.
[30] R. Stahlbock and S. Voß, "Vehicle routing problems and container terminal operations—An update of research," in *The Vehicle Routing Problem: Latest Advances and New Challenges*, B. Golden, S. Raghavan, and E. Wasil, Eds. Boston, MA, USA: Springer, 2008, pp. 551–589.
[31] D. Briskorn, A. Drexl, and S. Hartmann, "Inventory-based dispatching of automated guided vehicles on container terminals," *OR Spectr.*, vol. 28, no. 4, pp. 611–630, Oct. 2006.
[32] M. Grunow, H.-O. Günther, and M. Lehmann, "Strategies for dispatching AGVs at automated seaport container terminals," *OR Spectr.*, vol. 28, no. 4, pp. 587–610, Oct. 2006.
[33] K. H. Kim, S. M. Jeon, and K. R. Ryu, "Deadlock prevention for automated guided vehicles in automated container terminals," in *Container Terminals and Cargo Systems: Design, Operations Management, and Logistics Control Issues*, K. H. Kim and H.-O. Günther, Eds. Berlin, Germany: Springer, 2007, pp. 243–263.
[34] V. D. Nguyen and K. H. Kim, "A dispatching method for automated lifting vehicles in automated port container terminals," *Comput. Ind. Eng.*, vol. 56, no. 3, pp. 1002–1020, Apr. 2009.
[35] 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, Feb. 1999.
[36] K. H. Kim and K. Y. Kim, "Routing straddle carriers for the loading operation of containers using a beam search algorithm," *Comput. Ind. Eng.*, vol. 36, no. 1, pp. 109–136, Jan. 1999.
[37] K. H. Kim, K. M. Lee, and H. Hwang, "Sequencing delivery and receiving operations for yard cranes in port container terminals," *Int. J. Prod. Econ.*, vol. 84, no. 3, pp. 283–292, Jun. 2003.
[38] J. Lehnfeld and S. Knust, "Loading, unloading and premarshalling of stacks in storage areas: Survey and classification," *Eur. J. Oper. Res.*, vol. 239, no. 2, pp. 297–312, Dec. 2014.
[39] L. Chen and A. Langevin, "Multiple yard cranes scheduling for loading operations in a container terminal," *Eng. Optim.*, vol. 43, no. 11, pp. 1205–1221, Nov. 2011.
[40] A. H. Gharehgozli, Y. Yu, R. de Koster, and J. T. Udding, "An exact method for scheduling a yard crane," *Eur. J. Oper. Res.*, vol. 235, no. 2, pp. 431–447, Jun. 2014.
[41] 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, Aug. 2015.
[42] I. F. A. Vis and H. J. Carlo, "Sequencing two cooperating automated stacking cranes in a container terminal," *Transp. Sci.*, vol. 44, no. 2, pp. 169–182, May 2010.
[43] J. Nossack, D. Briskorn, and E. Pesch, "Container dispatching and conflict-free yard crane routing in an automated container terminal," *Transp. Sci.*, vol. 52, no. 5, pp. 1059–1076, Oct. 2018.
[44] T. Park, R. Choe, S. M. Ok, and K. R. Ryu, "Real-time scheduling for twin RMGs in an automated container yard," *OR Spectr.*, vol. 32, no. 3, pp. 593–615, Jul. 2010.
[45] R. Choe, T. S. Kim, T. Kim, and K. R. Ryu, "Crane scheduling for opportunistic remarshaling of containers in an automated stacking yard," *Flexible Services Manuf. J.*, vol. 27, nos. 2–3, pp. 331–349, Sep. 2015.
[46] V. Galle, C. Barnhart, and P. Jaillet, "Yard crane scheduling for container storage, retrieval, and relocation," *Eur. J. Oper. Res.*, vol. 271, no. 1, pp. 288–316, Nov. 2018.
[47] 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. Informat.*, vol. 25, no. 3, pp. 472–484, Aug. 2011.
[48] W. C. Ng and K. L. Mak, "An effective heuristic for scheduling a yard crane to handle jobs with different ready times," *Eng. Optim.*, vol. 37, no. 8, pp. 867–877, Dec. 2005.
[49] 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.
[50] S. Y. Huang and Y. Li, "Yard crane scheduling to minimize total weighted vessel loading time in container terminals," *Flexible Services Manuf. J.*, vol. 29, nos. 3–4, pp. 689–720, Dec. 2017.
[51] 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.
[52] U. Dorndorf and F. Schneider, "Scheduling automated triple cross-over stacking cranes in a container yard," *OR Spectr.*, vol. 32, no. 3, pp. 617–632, Jul. 2010.
[53] X. Han, Q. Wang, and J. Huang, "Scheduling cooperative twin automated stacking cranes in automated container terminals," *Comput. Ind. Eng.*, vol. 128, pp. 553–558, Feb. 2019.
[54] D.-H. Lee, Z. Cao, J. H. Chen, and J. X. Cao, "Load scheduling of multiple yard crane systems in container terminal with buffer areas," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2097, no. 1, pp. 70–77, Jan. 2009.
[55] W. Li, Y. Wu, M. E. H. Petering, M. Goh, and R. D. Souza, "Discrete time model and algorithms for container yard crane scheduling," *Eur. J. Oper. Res.*, vol. 198, no. 1, pp. 165–172, Oct. 2009.
[56] F. Zheng, X. Man, F. Chu, M. Liu, and C. Chu, "Two yard crane scheduling with dynamic processing time and interference," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 12, pp. 3775–3784, Dec. 2018.
[57] J. He, C. Tan, and Y. Zhang, "Yard crane scheduling problem in a container terminal considering risk caused by uncertainty," *Adv. Eng. Informat.*, vol. 39, pp. 14–24, Jan. 2019.
[58] D. Briskorn and P. Angeloudis, "Scheduling co-operating stacking cranes with predetermined container sequences," *Discrete Appl. Math.*, vol. 201, pp. 70–85, Mar. 2016.
[59] D. Briskorn, S. Emde, and N. Boysen, "Cooperative twin-crane scheduling," *Discrete Appl. Math.*, vol. 211, pp. 40–57, Oct. 2016.
[60] F. Jaehn and D. Kress, "Scheduling cooperative gantry cranes with seaside and landside jobs," *Discrete Appl. Math.*, vol. 242, pp. 53–68, Jun. 2018.
[61] D. Kress, J. Dornseifer, and F. Jaehn, "An exact solution approach for scheduling cooperative gantry cranes," *Eur. J. Oper. Res.*, vol. 273, no. 1, pp. 82–101, Feb. 2019.
[62] H. J. Carlo and F. L. Martínez-Acevedo, "Priority rules for twin automated stacking cranes that collaborate," *Comput. Ind. Eng.*, vol. 89, pp. 23–33, Nov. 2015.
[63] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *J. ACM*, vol. 7, no. 4, pp. 326–329, Oct. 1960.

[64] D. Whitley, "A genetic algorithm tutorial," *Statist. Comput.*, vol. 4, no. 2, pp. 65–85, Jun. 1994.
[65] N. M. Razali and J. Geraghty, "Genetic algorithm performance with different selection strategies in solving TSP," in *Proc. World Congr. Eng.* Hong Kong: International Association of Engineers, 2011, vol. 2, no. 1, pp. 1–6.
[66] F. M. Defersha and D. Rooyani, "An efficient two-stage genetic algorithm for a flexible job-shop scheduling problem with sequence dependent attached/detached setup, machine release date and lag-time," *Comput. Ind. Eng.*, vol. 147, Sep. 2020, Art. no. 106605.
[67] C.-C. Tsai and S. H. A. Li, "A two-stage modeling with genetic algorithms for the nurse scheduling problem," *Expert Syst. Appl.*, vol. 36, no. 5, pp. 9506–9512, Jul. 2009.
[68] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1979, p. 338.
[69] M. Pinedo, *Scheduling*. New York, NY, USA: Springer, 2012.
[70] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 1061–1071, Sep. 1996.

HENOKH YERNIAS FIBRIANTO received the B.S. degree in engineering physics from the Institut Teknologi Sepuluh Nopember, Indonesia, and the M.S. degree in industrial engineering from Pusan National University, South Korea. He is currently a Data Scientist with GrabTaxi Holdings Pte., Ltd., Singapore. His research interests include logistics, simulations, and optimizations.

BONGGWON KANG received the B.S. degree in industrial engineering from Pusan National University, South Korea, where he is currently pursuing the M.S. degree with the Department of Industrial Engineering. His research interests include analysis and optimization of material handling operations in container terminals and large-scale simulations in semiconductor fabs.

SOONDO HONG received the B.S. and M.S. degrees in industrial engineering from POSTECH, South Korea, and the Ph.D. degree from the Department of Industrial and Systems Engineering, Texas A&M University. He is currently an Associate Professor with the Department of Industrial Engineering, Pusan National University, South Korea. He has worked on scheduling and material handling at semiconductor and display companies. His research interests include logistics operations and material handling in manufacturing, warehousing, display, semiconductor, and container terminal industries. He is a member of INFORMS and the Korean Institute of Industrial Engineers.