Transportation Research Part E 138 (2020) 101966

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

Integrated optimization on yard crane scheduling and vehicle positioning at container yards

Chenhao Zhou, Byung Kwon Lee*, Haobin Li

Centre of Excellence in Modelling and Simulation for Next Generation Ports, Department of Industrial Systems Engineering and Management, National University of Singapore, 3 Research Link, Singapore 117602, Singapore

ARTICLE INFO

Keywords:
Container yard
Chebyshev movement
Yard crane scheduling
Vehicle position
Integrated optimization

ABSTRACT

A container yard is a storage facility that allows handling resources to improve operational efficiency by facilitating container flows at a container terminal. The container yard system consists of a set of storage blocks with yard cranes performing stacking and unstacking operations for containers to be transported by vehicles. High operational efficiency can be achieved by managing and coordinating the handling operations of yard cranes and vehicles (e.g., the yard crane scheduling, vehicle job dispatching, and coordinating handshakes between yard cranes and vehicles). This study proposes an integrated optimization approach for simultaneously determining the yard crane schedules and the vehicle parking positions under the Chebyshev movement allowing for the simultaneous movement of gantry and trolley of the yard crane. A mixed-integer programming model is formulated to optimize the problem, and the two-stage heuristic algorithm is developed to solve the problem efficiently. Several propositions are also provided to search the optimal boundary of vehicle parking slots for pairs of jobs. Numerical experiments are conducted to show the outperformance of the proposed heuristic algorithm compared to the well-known rule-based heuristics.

1. Introduction

A container yard system is a key facility temporarily accommodating outbound, inbound, and transshipment containers with a set of container handling equipment. The storage space not only supports the throughput of quayside operations (loading and unloading) but also the handling service for landside operations (receiving and delivery operations). It means that the operational efficiency of a container terminal is conditioned by the yard operation collaboratively achieved by the set of handling equipment. Studies on yard operations are typically categorized into storage space allocation, remarshaling and reshuffling, and equipment scheduling under the handling equipment configurations of Quay Cranes (QCs), vehicles, and Yard Cranes (YCs). Note that this study specifies vehicles as a mode of transportation for containers within a container terminal. The storage space allocation and the remarshaling/reshuffling are also conditioned by the equipment set (Carlo et al., 2014; Gharehgozli et al., 2015; Lehnfeld and Knust, 2014). A container yard consists of many storage blocks for stacking containers, and each block is surrounded by aisles for vehicles to travel and is equipped with YCs to perform stacking and unstacking operations. Hence, the yard operational efficiency is likely dependent on the scheduling operations for YCs and vehicles, including the handshakes between them, which is inevitable when there is a handshake between the two types of equipment.

There are typically two types of block layouts, namely the end-loading and side-loading blocks, as illustrated in Fig. 1. The end-

* Corresponding author.
E-mail addresses: zhou_chenhao@nus.edu.sg (C. Zhou), leebk@nus.edu.sg (B.K. Lee), li_haobin@nus.edu.sg (H. Li).

https://doi.org/10.1016/j.tre.2020.101966
Received 17 June 2019; Received in revised form 11 March 2020; Accepted 2 May 2020

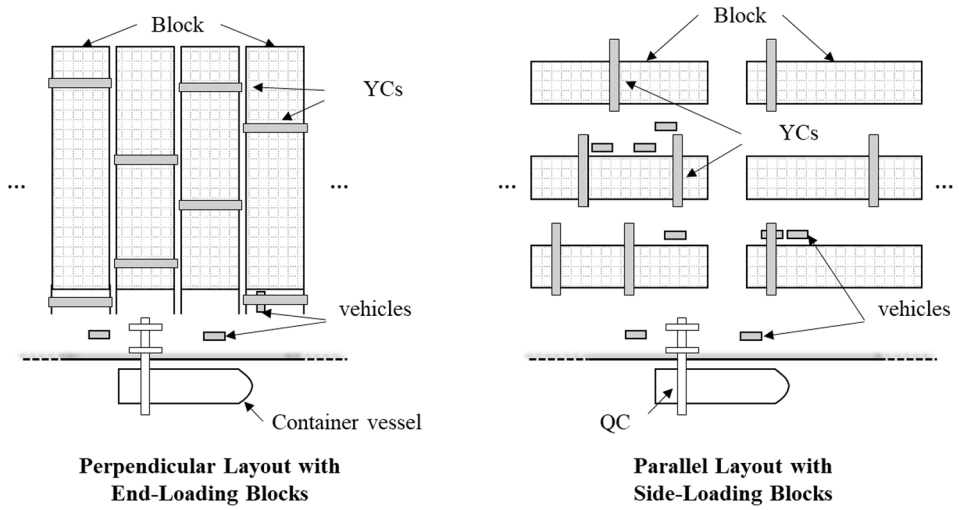

Fig. 1. Perpendicular layout and parallel layout.

loading block layout requires vehicles to interact with YCs at the ends of the block. This block layout has an advantage for container yards as the traffic controls at seaside and landside are separated so that the seaside blocks are accessed only by vehicles for loading (unstacking) and unloading (stacking) operations whereas the landside is accessed only by road trucks for receiving (stacking) and delivery (unstacking) operations. The representative real-world examples are the Euromax terminal in Rotterdam and Altenwerder terminal in Hamburg. The end-loading block layout requires YCs to perform a round-trip per handling job, possibly resulting in a long waiting time of vehicles. The side-loading block layout is often found in many conventional container terminals such as Pasir Panjang Terminal in Singapore and Modern Terminal in Hong Kong, leading to challenging traffic control requirements for different types of handling equipment. This block layout requires vehicles to temporarily park beside the block for the YCs to perform handing activities typically within a small range of the block in the vicinity of the vehicle to improve the operational efficiency by reducing the YC gantry travels.

A number of leading container terminals (e.g., Tuas Maritime Hubs in Singapore and Yangshan Deepwater Harbor Phase IV in China) have been applying automation technologies to side-loading blocks. A YC usually consists of three-movement components: (a) the gantry, which moves along the long-side of the block, (b) the trolley, which is attached on the gantry arm and moves perpendicularly to the gantry direction, and (c) the spreader (not shown in the figure), which is attached beneath the trolley and lifts containers from/to the stacking slot. The YC movement traditionally performs in a rectilinear way, as illustrated by solid arrow lines in Fig. 2. The rectilinear movement allows the trolley to move only after completing the gantry movement. Fig. 2 indexes the parking slots where vehicles temporarily stay for job handling as row numbers 0 and 6, and the stacking slots where containers are stacked as row numbers 1–5, together with bay numbers 1–20. The latest technology allows YCs to apply the Chebyshev metric to the two movement components (i.e., gantry and trolley) to reduce travel time. The Chebyshev movements are depicted as dotted arrow lines in Fig. 2. Hoisting and lowering the spreader must be performed only after the movement of gantry or trolley for the safety and stability of container handling.

The Chebyshev movement contributes to the reduction of YC travel time compared to the rectilinear movement (Lee and Kim, 2010). The side-loading block layout provides room for further improvement of the operational efficiency of YCs when looking into the YC movement together with vehicle parking positions. For example, referring to Fig. 2, it is assumed that the unit travel time per slot to be 1 for both gantry and trolley movements and the current trolley position at the slot of (bay 1 & row 5). So the total travel

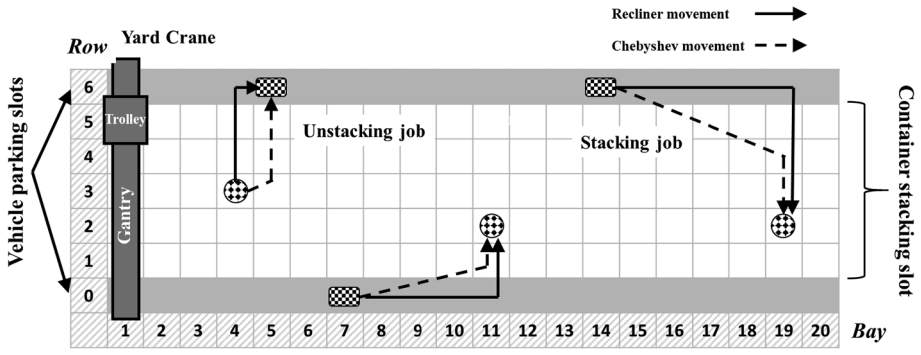

Fig. 2. Illustration of side-loading layout block and YC movement.

time is shorter when the YC performs a stacking activity to the slot of (bay 11 & row 2) directly from the parking position beside bay 7, (bay 7 & row 0), than having the vehicle and YC first move to beside bay 11, (bay 11 & row 0) before YC operation. Without considering the pick-up and release time, the parking slot of (bay 11 & row 0) requires 12 travel time-units, whereas the YC needs to spend ten travel time-units for the parking slot of (bay 7 & row 0). This shows that the Chebyshev movement can potentially further reduce the YC travel time when the YC operation sequence is coordinated with vehicle parking positions in the side-loading block layout. In the alternative end-loading block layout, there will be little merit to apply the Chebyshev movement due to the long round-trip distance and limited parking positions.

The YC scheduling problem is a representative research theme in the area of equipment scheduling for container terminals as it is the most popular equipment type for handling containers at the yard. A key element for YC scheduling is the reduction of the completion time (i.e., makespan) for incoming jobs, including the waiting time for handshakes. This study proposes a significant element (i.e., vehicle parking positions) contributing to the operational efficiency of a YC at a side-loading block by utilizing the advanced technology (i.e., Chebyshev movement). This study develops an integration scheduling approach for YC scheduling and vehicle parking positions to minimize the makespan. This study contributes to the followings:

- Simultaneous optimization of the job sequencing of YCs and the parking positions of vehicles for a batch of jobs;
- Mixed-integer programming models for optimally integrating the two scheduling elements (i.e., YC sequencing and vehicle positioning) and a two-stage heuristic algorithm for efficiently finding solutions; and
- Myopic decision propositions determining the optimal vehicle parking positions for a pair of job types under the Chebyshev movement.

According to our best knowledge, this is the first study to optimize the two scheduling elements (i.e., YC sequencing and vehicle positioning) simultaneously. The remaining of this study is organized as follows: Section 2 reviews the literature; Section 3 provides mixed-integer programming models to optimize the problem; Section 4 develops a tabu search-based heuristic algorithm to solve the problem efficiently; Section 5 conducts various experiment and examines the efficacy of the proposed algorithm compared to benchmarking algorithms, and conclusions are drawn in Section 6.

2. Related studies

The investigated YC scheduling studies can be applied to both conventional and automated YCs as the constructed schedules would be managed by a computerized decision-making system covering all handling sub-systems at a container terminal.

Scheduling problems of YCs have been popularly studied over several decades in application in container terminals concentrating on reducing empty travels of YCs and the waiting queue of vehicles. The single YC scheduling problem is a well-known variant of traveling salesman problems, as studied by Kim and Kim (1999) and Kim and Kim (2003). Kim and Kim (1999) optimized the bay visiting sequence of a YC and the number of container handlings at each bay at a side-loading block, and Kim and Kim (2003) suggested metaheuristic approaches for the same problem. Kim et al. (2003) provided a rolling horizon dynamic programming approach to optimize the operation sequence of a single YC and compared various sequencing rules and learning-based heuristics. Gharehgozli et al. (2014b) formulated the branch-and-bound based optimal algorithm to optimize the operation sequence of a single YC considering round-trip travels under the Chebyshev movement.

As for multiple YCs at a block, both the interference and the cooperation between adjacent YCs become key determinants for scheduling problems. The two-YC problems are categorized into the studies on twin YCs and passing YCs. In the twin YCs problems, Gharehgozli et al. (2014a) proposed a scheduling problem of twin YCs determining the sequence to stack containers at a single end-loading block and developed an adaptive large neighborhood search heuristic for fast computation. Gharehgozli et al. (2017) investigated the effects of the handshake area on the performance of the twin YCs by taking into account containers that need to pass through the block in a relay way. Hu et al. (2016) formulated the twin YCs scheduling problem in a time–space diagram and developed strategy-based algorithms. Zheng et al. (2018) developed a dividing-sequencing-comparing algorithm, which divides tasks randomly into two separate sets for the two YCs and selecting the best processing scheme for a given number of times for twin YCs by considering the variant processing time of a task. For the passing two YCs, Vis and Carlo (2010) proposed a scheduling algorithm on top of the algorithm to derive a lower bound for the makespan for two passing YCs that cooperate in a block. Saini et al. (2017) developed a continuous time Markov chain model to estimate the throughput capacity of the dual passing YC system at an end-loading block under YC operational protocols.

The problem complexity increases exponentially when more than two YCs operate concurrently at a block. Wu et al. (2015) studied the non-crossing multiple YC scheduling at a side-loading block by dynamically clustering the job area of each YC. Dorndorf and Schneider (2010) developed a beam search-based scheduling algorithm for the triple YC system consisting of two inner and one outer YCs. The objective is to minimize the makespan for the job set, considering the crane traveling, interference, and tardiness. Speer and Fischer (2016) conducted a comparative analysis with various performance measures on four YC systems working on an end-loading block.

The previous studies on the YC scheduling have been mostly applied to the end-loading block layout (Dorndorf and Schneider, 2010; Vis and Carlo, 2010; Gharehgozli et al., 2014a, 2014b, 2017; Hu et al., 2016; Speer and Fischer, 2016; Saini et al., 2017), whereas a few studies have been worked on the side-loading block layouts (Kim and Kim, 1999, Kim et al., 2003; Kim et al., 2003; Wu et al., 2015; Zheng et al., 2018).

The operations schedule of YCs is highly dependent on the handling requests by vehicles. Many studies on YC scheduling consider

vehicle arrivals as uncertain or as given information. A number of studies have discussed the necessity of integrated schedules. Chen et al. (2007) studied the coordination of different types of handling equipment, such as QCs, vehicles, and YCs. The authors developed a mixed-integer programming model and the tabu search-based heuristic to minimize the makespan under the QC work schedule. Chen et al. (2013) proposed a three-stage iterative algorithm modeled by the constraint programming to coordinate the schedule of the three types of equipment optimally. The first stage generates the operations schedules of both QCs and YCs, and the second stage generates the route schedule of vehicles, and the last stage sequences the individual tasks using a disjunctive graph. Cao et al. (2010) proposed an integrated scheduling approach for vehicles and YCs with the formulation of minimizing the makespan for the loading operations by optimizing the routing sequence of YCs considering the arrivals of vehicles. Luo and Wu (2015) developed a customized genetic algorithm to integrate the scheduling work of vehicles and YCs, as well as the storage location of containers considering the dual-cycle operation of a vehicle under the predetermined QC schedule for a container yard having the end-loading blocks. Lu and Le (2014) discussed the integration scheduling of YCs and vehicles considering uncertain factors such as the travel speeds of vehicles, YCs, and unit time of hoisting/lowering operations by developing particle swarm optimization-based heuristic algorithm.

According to previously published studies, in addition to efforts improving the YC efficiency, increasing attention has been given to the coordination between the vehicles and YCs when the system efficiency of the yard block is of interest. Researchers have discussed the coordination of the handshakes between a vehicle and a YC by controlling vehicle visits and YC operational sequence. However, researchers have been paid little attention to the parking slot determination of vehicles when scheduling YC operations. Unlike many YC scheduling studies, this study works on the side-loading blocks, which will be highly beneficial from the YC Chebyshev movement. In addition, this study introduces another key determinant of vehicle positioning to YC scheduling, which allows a YC to fully utilize the movement components under the Chebyshev metric to reduce the travel time and increase the block efficiency. While the scheduling algorithm developed by this study can be applied to both conventional and automated container yards, the employment of Chebyshev technology and the determinants of the scheduling objective may require computerized decision-making and communications between the decision-making system and the handling equipment.

3. Model development

YCs and vehicles need to be well-coordinated for the efficient completion of stacking and unstacking jobs in a storage block. Due to operation uncertainty (e.g., delay, congestion, and waiting) generated from across all container handling sub-systems (i.e., wharf, gate, yard, etc.), the arrivals of vehicles have dynamic nature from the perspective of a YC, though, advanced tracking technologies and well-managed storage space allocation have helped to improve the predictability of vehicle arrivals. When a long parking queue is created at any parking position at the block, it causes negative effects that cascade to the handling operations and lane traffic flows at adjacent bays. Therefore, the reservation and release of a parking slot should be effectively managed together with the job scheduling of a YC.

This section develops an optimization model to simultaneously determine the YC handling sequence and the vehicle parking slots for a batch of jobs at the side-loading block layout via utilizing the technical advantage of the Chebyshev movement to minimize the makespan. Without losing generality, the following assumptions can be made:

(1) There is only a single YC on a block as the number of bays in a side-loading block layout deployed by a YC is typically less than 20 bays. A limited number of bays is typically allocated to each YC to minimize the chance of interference between adjacent YCs;
(2) The vehicle arrival time, which is referred to as the job ready time, is known in advance as advanced tracking technologies help to detect the real-time locations of vehicles;
(3) The handling time for a stacking/unstacking job is a constant, as the YC travel time is of interest and represents the makespan;
(4) Both sides of the block allow vehicles parking for jobs;
(5) Both acceleration and deceleration of the YC are excluded from the model simplicity, and;
(6) Each parking slot is reserved for only one vehicle for each batch of jobs (i.e., during the planning horizon), and the parking slots of finished jobs are released and reserved by other vehicles in the subsequent planning horizon.

3.1. Prerequisites

There are two types of jobs: a stacking job requiring a YC to pick-up a container from a vehicle at a parking slot and stack into a pre-assigned stacking slot, and an unstacking job requiring a YC to retrieve a container from the stacking slot and release it onto a vehicle at a parking slot. Since the stacking slots for jobs for each batch are given information, the decision variables left are parking slots for vehicles and the operation sequence of a YC.

The locations (i.e., parking and stacking slots) can be defined by its bay and row numbers by referring to Fig. 2. Let a job j be defined as its type $\tau_j \in \{S, U\}$ (known information), where S and U mean stacking and unstacking job types respectively, stacking slot $\alpha_j = (X_j^{\alpha}, Y_j^{\alpha})$ (known information) and parking slot $\beta_j = (X_j^{\beta}, Y_j^{\beta})$ (unknown information). Since the Chebyshev movement is employed, when a YC travel between two positions $\alpha_j = (X_j^{\alpha}, Y_j^{\alpha})$ and $\beta_j = (X_j^{\beta}, Y_j^{\beta})$, the travel time is calculated as $T(\alpha_j, \beta_j) = max(t_h\,|X_j^{\alpha} - X_j^{\beta}|,\, t_v\,|Y_j^{\alpha} - Y_j^{\beta}|)$, where t_h and t_v are the unit travel time of gantry and trolley movements, respectively. T_j^{H1} and T_j^{H2} are the first and second operational components for job j. For example, for a stacking job, T_j^{H1} represents the pick-up time for a vehicle and T_j^{H2} represents the stacking time of the YC. On the other hand, for an unstacking job, T_j^{H1} is the retrieval time from the stack, and T_j^{H2} represents the release time for a vehicle. When a vehicle arrives at a designated parking slot, the job is ready at the

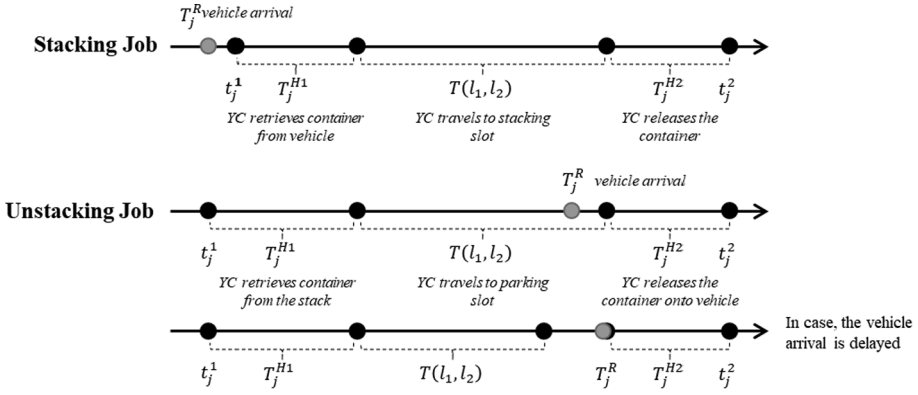

Fig. 3. Time components of stacking and unstacking jobs.

time T_j^R.

Fig. 3 explains the time sequence of the YC operation for stacking and unstacking jobs. t_j^1 and t_j^2 represent the start time and end time of a job. A stacking job can only start after the vehicle arrives, while the YC can retrieve a container from the stack before vehicle arrival for an unstacking job. Note that the unstacking job is completed when the YC releases the container onto the arrived vehicle.

Fig. 4 demonstrates an example of a scheduling result (i.e., determination of parking slots and job handling sequence) for a batch of three jobs. The YC starts from (bay 1, row 0) and takes an empty travel ① to (bay 4, row 3) to retrieve the container for job 1 and takes a laden travel ② to its parking slot (bay 5, row 6) under the Chebyshev metric. As the stacking job 2 is ready at (bay 7, row 0), the YC takes an empty travel ③ to pick the container up and takes another laden travel ④ to release the container into the stacking slot (bay 11, row 2). Similar movements go for the travel routes ⑤ and ⑥.

3.2. Integrated problem formulation

This subsection provides a mixed-integer programming model to optimize the YC handling sequence and the parking slots for incoming vehicles.

Parameters

X = The total number of bays

Y = The total number of rows

$\boldsymbol{J}$ = The set of jobs $\boldsymbol{J} = \{0, 1, 2. ..J + 1\}$. 0 and $J + 1$ are dummy jobs representing the start and end positions of the trolley.

$\boldsymbol{P}$ = The set of parking slots, $\boldsymbol{P} = \{1, 2. ..P\}$. The parking slot p can be presented as (X_p, Y_p).

τ_j = Types of job j, $\tau_j \in \{S, U\}$, where $j \in \boldsymbol{J}\backslash\{0, J + 1\}$. S and U stand for stacking and unstacking jobs, respectively.

α_j = The stacking slot for job j, $\alpha_j = (X_j^{\alpha}, Y_j^{\alpha})$ where $j \in \boldsymbol{J}\backslash\{0, J + 1\}$.

ps = The start position of the trolley as a dummy job, $ps = (X^s, Y^s)$, associated with 0 of $\boldsymbol{J}$.

pe = The end position of the trolley as a dummy job, $pe = (X^e, Y^e)$, associated with $J + 1$ of $\boldsymbol{J}$.

T^X and T^Y = The unit travel time in gantry and trolley directions, respectively.

T_j^{H1} and T_j^{H2} = The handling time of job j for the first and second operational components.

T_j^R = The ready time of job j, where $j \in \boldsymbol{J}\backslash\{0, J + 1\}$.

$T(a, b)$ = The required travel time from positions a to b under Chebyshev metric.

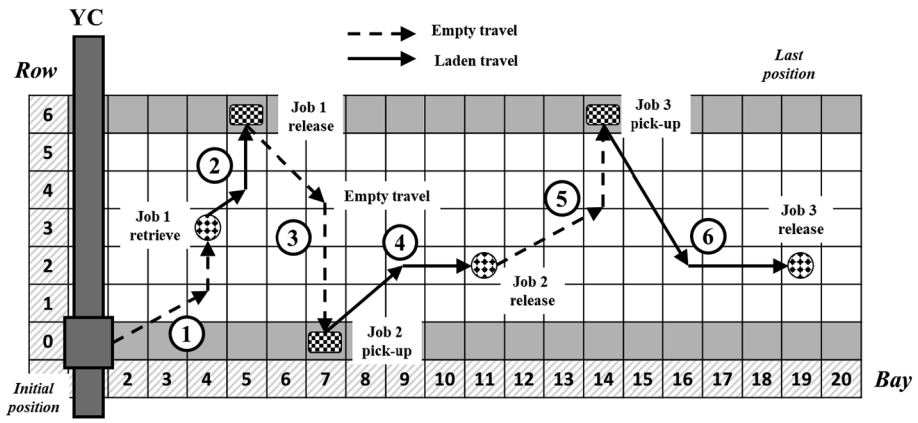

Fig. 4. An example of YC operation sequence.

M = A very large positive number.

Decision variables

x_{jp} = Binary decision variable. $x_{jp} = 1$ if and only if the parking slot for job j is assigned to p, that is, $\beta_j = (X_j^\beta, Y_j^\beta) = (X_p, Y_p)$.

$s_{j_1 j_2}$ = Binary decision variable. $s_{j_1 j_2} = 1$ if and only if the job j_1 is handled before the job j_2, where $j_1 \neq j_2$.

t_j^1 = Continuous decision variable. The start time of the YC for job j.

t_j^2 = Continuous decision variable. The end time of the YC for job j.

te = Continuous decision variable. The completion time of the YC for a batch of jobs.

Modeling of integrated problem

Objective: $min\ (te)$ (1)

Subject to:

Constraints (2) – (4) formulate the handling sequence.

$$s_{0,j}=1 \qquad \forall\, j \in \boldsymbol{J}\backslash\{0\} \tag{2}$$

$$s_{j,J+1}=1 \qquad \forall\, j \in \boldsymbol{J}\backslash\{J+1\} \tag{3}$$

$$s_{j_1 j_2} + s_{j_2 j_1} = 1 \qquad \forall\, j_1, j_2 \in \boldsymbol{J}, j_1 \neq j_2 \tag{4}$$

Constraints (5) and (6) determine that each job to have only one parking slot, and each parking slot can only be reserve for one job.

$$\sum\nolimits_p x_{jp}=1 \qquad \forall\, j \in \boldsymbol{J}\backslash\{0, J+1\} \tag{5}$$

$$\sum\nolimits_{j\in J/\{0,J+1\}} x_{jp} \leq 1 \qquad \forall\, p \in \boldsymbol{P} \tag{6}$$

Constraints (7) – (17) describe the time relation of each job or between jobs. To be specific, the handling time for each job can be presented as:

$$t_j^1 + T_j^{H1} + T(\alpha_j, p) + T_j^{H2} \leq t_j^2 + M\cdot(1 - x_{jp}) \qquad \forall\, j \in \boldsymbol{J}\backslash\{0, J+1\}, \forall\, p \in \boldsymbol{P} \tag{7}$$

If a job is a stacking job ($\tau_j = S$), the YC can pick-up the container only after the vehicle arrival.

$$T_j^R \leq t_j^1 \qquad \forall\, j \in \boldsymbol{J}\backslash\{0, J+1\}, \tau_j = S \tag{8}$$

On the other hand, if a job is an unstacking job ($\tau_j = U$), the YC can release the container only after the empty vehicle arrival.

$$T_j^R + T_j^{H2} \leq t_j^2 + M\cdot(1 - x_{jp}) \qquad \forall\, j \in \boldsymbol{J}\backslash\{0, J+1\}, \tau_j = U, \forall\, p \in \boldsymbol{P} \tag{9}$$

The first slot for a stacking job is a parking slot and the second is a stacking slot, whereas the first slot is a stacking slot and the second is a parking slot for an unstacking job. The four possible combinations of the proceeding job j_1 and succeeding job j_2 are as follows:

- For a pair of two stacking j_1 and j_2, the transit movement is determined by α_{j_1} and $x_{j_2 p}$;
- For a pair of stacking and unstacking jobs j_1 and j_2, the transit movement is determined by α_{j_1} and α_{j_2} ;
- For a pair of two unstacking jobs j_1 and j_2, the transit movement is determined by $x_{j_1 p}$ and α_{j_2} , and;
- For a pair of unstacking and stacking jobs j_1 and j_2, the transit movement is determined by $x_{j_1 p}$ and $x_{j_2 p}$;

For any pair of two jobs j_1 and j_2, for both stacking jobs ($\tau_{j_1} = S$ and $\tau_{j_2} = S$),

$$t_{j_1}^2 + T(\alpha_{j_1}, p) \leq t_{j_2}^1 + M\cdot(2 - s_{j_1 j_2} - x_{j_2 p})$$

$$\forall\, j_1, j_2 \in \boldsymbol{J}\backslash\{0, J+1\}, j_1 \neq j_2, \tau_{j_1} = S, \tau_{j_2} = S, \forall\, p \in \boldsymbol{P} \tag{10}$$

For a stacking job ($\tau_{j_1} = S$) followed by an unstacking job ($\tau_{j_2} = U$),

$$t_{j_1}^2 + T(\alpha_{j_1}, \alpha_{j_2}) \leq t_{j_2}^1 + M\cdot(1 - s_{j_1 j_2})$$

$$\forall\, j_1, j_2 \in \boldsymbol{J}\backslash\{0, J+1\}, j_1 \neq j_2, \tau_{j_1} = S, \tau_{j_2} = U \tag{11}$$

For a pair of unstacking jobs ($\tau_{j_1} = U$ and $\tau_{j_2} = U$),

$$t_{j_1}^2 + T(p, \alpha_{j_2}) \leq t_{j_2}^1 + M\cdot(2 - s_{j_1 j_2} - x_{j_1 p})$$

$$\forall\, j_1, j_2 \in \boldsymbol{J}\backslash\{0, J+1\}, j_1 \neq j_2, \tau_{j_1} = U, \tau_{j_2} = U, \forall\, p \in \boldsymbol{P} \tag{12}$$

When an unstacking job ($\tau_{j_1} = U$) is followed by a stacking job ($\tau_{j_2} = S$),

$$t_{j_1}^2 + T(p_1, p_2) \leq t_{j_2}^1 + M\cdot(3 - s_{j_1 j_2} - x_{j_1 p_1} - x_{j_2 p_2})$$

$$\forall j_1, j_2 \in \boldsymbol{J}\backslash\{0, J+1\}, j_1 \neq j_2, \tau_{j_1} = U, \tau_{j_2} = S, \forall p_1, p_2 \in \boldsymbol{P} \tag{13}$$

For the start and end positions, there is a movement from starting position to the first job, and another movement from the last job to the ending position. For the first job $j \in \boldsymbol{J}\backslash\{0, J+1\}$, if the jobs type is a stacking job ($\tau_j = S$),

$$T(ps, p) \leq t_j^1 + M\cdot(2 - s_{0,j} - x_{jp}) \qquad \forall j \in \boldsymbol{J}\backslash\{0, J+1\}, \tau_j = S, \forall p \in \boldsymbol{P} \tag{14}$$

If it is an unstacking job ($\tau_j = U$),

$$T(ps, \alpha_j) \leq t_j^1 + M\cdot(1 - s_{0,j}) \qquad \forall j \in \boldsymbol{J}\backslash\{0, J+1\}, \tau_j = U \tag{15}$$

For the last job $j \in \boldsymbol{J}\backslash\{0, J+1\}$, for a stacking job ($\tau_j = S$),

$$t_j^2 + T(\alpha_j, pe) \leq te + M\cdot(1 - s_{j,J+1}) \qquad \forall j \in \boldsymbol{J}\backslash\{0, J+1\}, \tau_j = S \tag{16}$$

And for an unstacking job ($\tau_j = U$),

$$t_j^2 + T(p, pe) \leq te + M\cdot(2 - s_{j,J+1} - x_{jp}) \qquad \forall j \in \boldsymbol{J}\backslash\{0, J+1\}, \tau_j = U, \forall p \in \boldsymbol{P} \tag{17}$$

3.3. Constraint relaxation for shared parking slots

The previous subsection assumes that each vehicle reserves only a parking slot for each batch of jobs (i.e., the planning horizon). This subsection relaxes the assumption so that a number of vehicles can share a parking slot if and only if the completion time of a job does not conflict with the ready time of the next job. When two jobs j_1 and j_2 have the same parking slot, the job j_1 is handled before the job j_2 (i.e., $s_{j_1 j_2} + x_{j_1 p} + x_{j_2 p} = 3$) if and only if the ready time of the job j_2 is not earlier than the completion time of the job j_1 (i.e., $T_{j_2}^R \geq t_{j_1}^2$). Fig. 5 explains the time conflict for two jobs and the condition of sharing a parking slot for the two vehicles.

Modeling of integrated problem with shared parking slots

Objective: (1)

Subject to: (2)–(5), (7)–(17)

$$T_{j_2}^R \geq t_{j_1}^2 - M(3 - s_{j_1 j_2} - x_{j_1 p} - x_{j_2 p}) \qquad \forall j_1, j_2 \in \boldsymbol{J}\backslash\{0, J+1\}, j_1 \neq j_2, \forall p \in \boldsymbol{P} \tag{18}$$

Note that the relaxed model introduces the constraint (18), allowing for the sharing of a parking slot if there is no time conflict between two jobs instead of the constraint (6) limiting each parking slot to a vehicle.

3.4. Theoretical upper bound

To further restrict the value of M, a theoretical upper bound is derived as below. In the worst scenario, one job may operate from one corner to the other corner and then return the original corner. Thus, the traveling time of each job including empty travel is $2\bullet max(t_h\bullet X, t_v\bullet Y)$ and total traveling time is $2\bullet max(t_h\bullet X, t_v\bullet Y)\bullet(|\boldsymbol{J}| + 1)$, where $|\boldsymbol{J}|$ represents total job number, and $(|\boldsymbol{J}| + 1)$ represents that the travels include both actual jobs, and the initial and final empty moves. From ready time perspective, the worst scenario is to have all jobs starting from the same time, which can be presented as $max_{j\in\boldsymbol{J}\backslash\{0,J+1\}} T_j^R$. Therefore the theoretical upper bound TUB can be written as

$$TUB = (max_{j\in\boldsymbol{J}\backslash\{0,J+1\}} T_j^R) + 2\bullet max(t_h\bullet X, t_v\bullet Y)\bullet(|\boldsymbol{J}| + 1) + \sum_{j\in\boldsymbol{J}\backslash\{0,J+1\}} (T_j^{H1} + T_j^{H2}) \tag{19}$$

4. Heuristic approach

The proposed problem in this study has a nature of short-term scheduling requiring fast response time as the YC needs to support

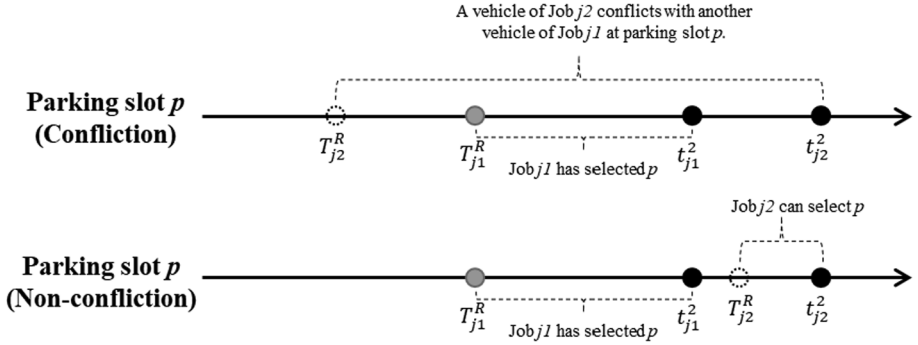

Fig. 5. Illustrated example of time conflict at a parking slot for two vehicles.

the seaside and landside operations continuously via serving incoming vehicles. A computational challenge comes from the synchronized decision on optimizing the parking slots for vehicles and the operation sequence of a YC with discretized decision variables. This section develops a tabu search-based two-stage heuristic algorithm to overcome the computational challenge of the proposed optimization models (Sections 3.2 and 3.3).

4.1. Tabu search-based two-stage algorithm

The proposed tabu search-based two-stage approach (TSA) decomposes the problem into the two sub-problems (stages), namely, the YC sequencing sub-problem and the vehicle parking sub-problem. Tabu search is a local search algorithm that restricts the feasible neighborhood by neighbors that are excluded (Glover, 1990, Edelkamp and Schrödl, 2012). Therein, neighborhood states are maintained in a data structure – tabu list, which helps avoid being trapped in a local optimum. If all neighbors are tabu, a move that worsens the value of the objective function is accepted, whereas an ordinary deepest decent method would be trapped. In addiiton, aspiration criteria and diversification can be introduced as a refinement (Liu et al., 2003): aspiration criterion can admit a solution, even if it is forbidden by the tabu list when it outperforms the best result found so far; diversification strategy is used to encourage searching more solution space and jump out the local optimal regions.

In this study, the TSA generates a neighborhood solution representing the handling sequence of jobs during its process. As the tabu search only samples the job handling sequence and selects the best sampling direction, the move evaluation step needs to solve a feasible parking decision efficiently to calculate the objective value during the search process over a great number of samples. Regarding feasible parking decisions, a sequential parking heuristic is developed to solve the sub-problem. The parking heuristic aims to determine the parking slots for every pair of consecutive jobs sequentially by utilizing the propositions proposed in Appendix A. Note that this study designs the tabu search structure by referring to Malek et al. (1989) and Basu et al. (2017) as the job handling sequence problem can be seen as a variant of the well-known traveling salesman problem.

The TSA structure is presented in Fig. 6 in the form of pseudo-codes. The algorithmic structure likely converges the solutions iteratively by the two stages with high efficiency and quality. The solution convergence is discussed with the experiment results in Section 5.4.

- **Solution representation:** In the original problem, a solution of the sequencing sub-problem is represented as a series of binary variables $s_{j_1 j_2}$ to represent precedence relation. Hence a solution of TSA can be represented as a job sequence (i.e., a series of job indices, e.g., [5, 3, 1, 2…]).
- **Initial solution:** The initial solution is chosen from the best solution (minimal objective value) among the randomly generated $N_{neighbor}$ outcomes and the solutions of the three heuristic algorithms of the sub-section 4.3.
- **Neighborhood:** $N_{neighbour}$ neighborhoods of the current solution are generated by swapping two different jobs. For the ease of neighborhood selection, the neighborhood list will be sorted according to the ascending order of the objective values first, so the neighborhood with minimal objective value is in the front of the list. Each neighborhood will be checked: (1) if it is not on the tabu list, and (2) if it is on the tabu list, but the aspiration strategy is met. If either (1) or (2) is satisfied, then select the neighborhood as current solution, and add the two jobs to the tabu list. Once the current solution s_{now} is found and if its objective value $f(s_{now})$ is better than the best objective value f_{best}, the best solution is replaced with the current solution.
- **Tabu list:** Tabu list is defined as a list of job pairs that were swapped (smaller job index goes first), e.g., {(2,5), (3,5), …}. The tabu list has a length limit, and the first tabu on the list will be removed if the list is full.
- **Move evaluation $f(\bullet)$ and sequential parking heuristic:** As the TSA only samples job handling sequences, the algorithm needs a feasible parking decision to calculate the objective value, which can be used as the move evaluation$f(\bullet)$.

The sequential parking heuristic algorithm is to determine the parking slots for vehicles sequentially for every consecutive job-pair via utilizing the propositions proposed in Appendix A. The propositions serve to significantly reduce the search effort to obtain a good feasible solution. The algorithm procedure repeats until all parking slots are determined, and ultimately provides a feasible parking decision with the corresponding objective value. Note that if a parking slot is reserved by a job of the batch, the other jobs cannot occupy this slot. A flow chart of this heuristic approach is given in Fig. 7.

The outcome of the step "find optimal parking for j_1, j_2" depends on the types of job j_1 and j_2. For example, for two stacking jobs, the YC needs to take a laden travel from the parking slot of j_1 to its stacking slot, an empty travel to the parking slot of j_2, and then a laden travel to the stacking slot of j_2. In this pair of jobs, according to the propositions in Appendix A, the determinant is the parking slot for j_2. Note that the parking slot of j_1 has been determined in the previous steps. Since a vehicle might have multiple possible parking slots having the same travel time for the pair of jobs, the tie is broken by choosing a parking slot nearest to the corresponding job (i.e., job j_2). Once a vehicle chooses a parking slot, the slot is removed from the available parking list of the search algorithm.

- **Aspiration strategy:** The strategy is applied when selecting the current solution from the neighborhoods. If the neighborhood is on tabu list, then there are two criteria to check (1) if the objective value of the neighborhood is smaller than the global best solution (Malek et al., 1989), and (2) if the criterion (1) is not met, but $f(s_j) < f(s_{now})$ and certain criteria are met (Salhi, 2002). The condition (2) generates a uniform random number between 0 and 1 and compares it with a fixed threshold A. If either condition is met, the neighborhood can be selected as the current solution, even if it is on the tabu list.
- **Diversification strategy:** If the global best solution remains unchanged for N_{no_change} iterations, the tabu list will be emptied and the current solution is regenerated the initial solution from the $N_{neighbor}$ randomly generated solutions.

```
s_0 = initial_and_best(); // generate initial solutions and get the best one
s_now = s_0; s_best = s_0; f_best = f(s_0);
tabu_list ← ∅; neighour_list ← ∅; obj_list ← ∅;
i = 0; i_change = 0; i_neighbor = 0;
While (i < N_max)
{
    neighour_list ← neighour(s_now) // generate N_neighbor neighborhoods of s_now
    for (s_j ∈ neighour_list)
        obj_list ← f(s_j); // f(s_j) obtains parking decision and calculate the objective value
    // sort neighborhood list according to the ascending order of the objective values
    neighour_list ← sort(neighour_list, obj_list);
    // select candidate solution from neighborhoods
    for (s_j ∈ neighour_list)
    {
        // check_tabu(s_j) checks if solution s_j is on the tabu list
        If (check_tabu(s_j)= false)
        // if current candidate is not in tabu list
        {
            // update_tabu_list(s_j) adds the current solution to tabu_list
            s_now = s_j; update_tabu_list(s_j); break;
        }
        Else if (f(s_j) < f_best)
        // aspiration strategy: if the candidate is in the tabu list, but the objective value is
    smaller than f_best
        {
            s_now = s_j; update_tabu_list(s_j); break;
        }
        Else
        // aspiration strategy: if the candidate is in tabu list and objective value is worst than
        f_best
        {
            double a ← random(0,1); // uniformly generate a float value between 0 & 1
            If (a ≤ A && f(s_j) < f(s_now))
            {
                s_now = s_j; update_tabu_list(s_j); break;
            }
        }
    }
    If (f(s_now) ≤ f_best)
    {
        s_best = s_now; f_best = f(s_now); i_no_change = 0;
    }
    Else
        i_change ← i_change + 1; // count the case that the best solution does not improve
    If (i_change = N_no_change) // regenerate the initial solution for diversification
    {
        s_now = initial_and_best(); tabu_list ← ∅; i_change = 0;
    }
    i ← i + 1;
}
```

Fig. 6. Pseudo-code of TSA.

- **Termination condition:** The TSA terminates after N_{max} of iterations.

4.2. Revised TSA for shared parking slots

Since multiple vehicles can share a parking slot as long as there is no time conflict, the proposed TSA needs to be revised to allow the sharing of parking slots when solving the sequential parking heuristic, as described in Fig. 8. The sequential parking heuristic determines parking slots for every pair of consecutive jobs sequentially. The revised sequential parking heuristic algorithm needs to record information on the latest occupation time of each parking slot. Whenever a parking slot is reserved for a job, the latest occupation time of this slot is equal to the end time of this job. Therefore, the revised heuristic searches parking slots only for a job with the condition that their latest occupation time values should be earlier than or equal to the job ready time.

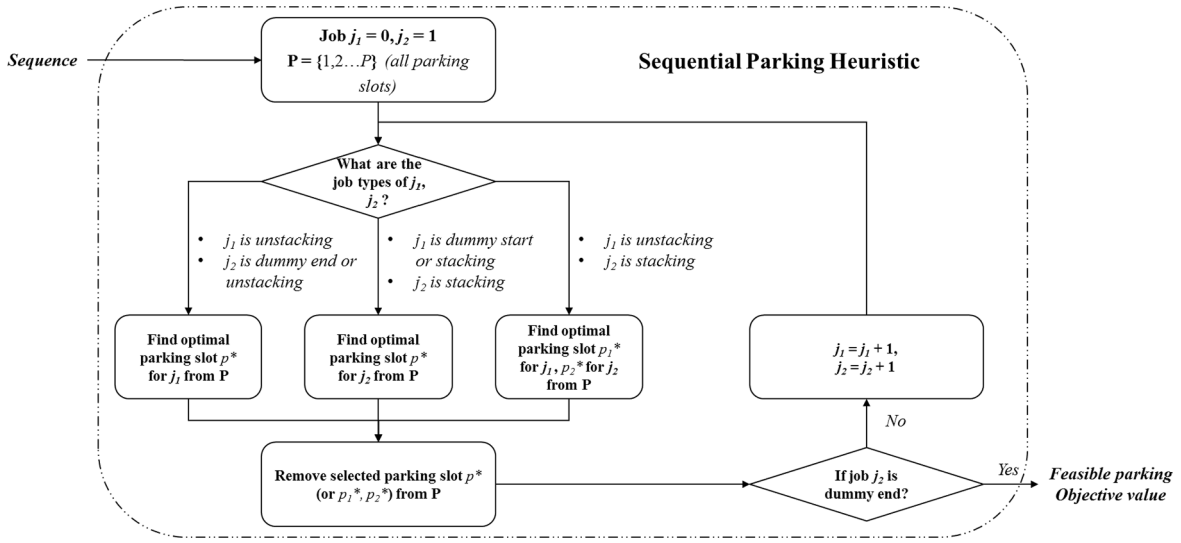

Fig. 7. Illustration of sequential parking heuristic algorithm.

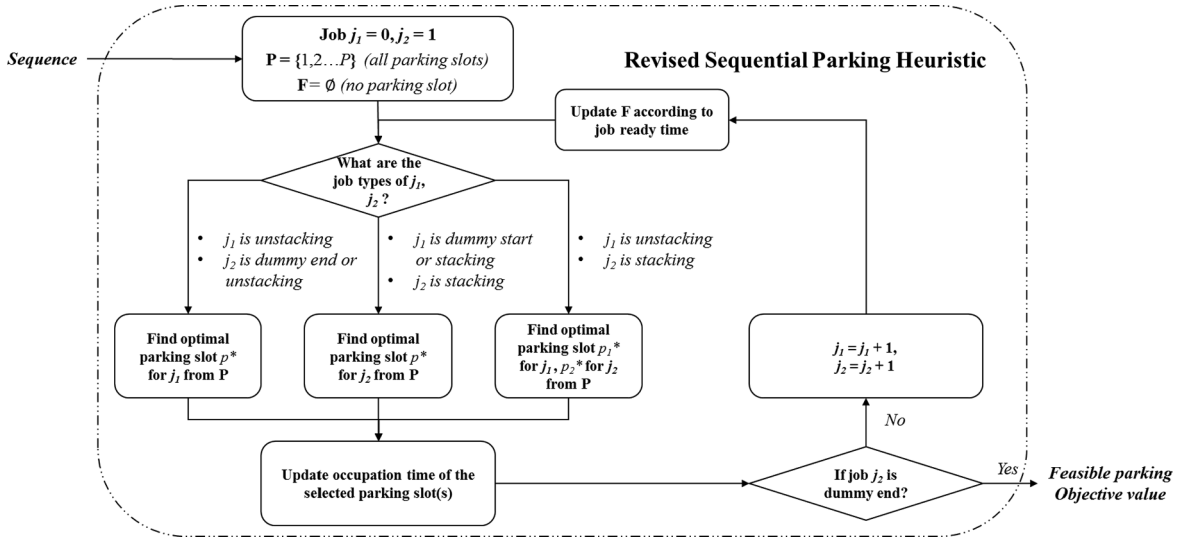

Fig. 8. Illustration of revised sequential parking heuristic algorithm.

4.3. Benchmark algorithms

Kim et al. (2003) presented several rule-based heuristics with high practicality for the job sequencing strategies, such as first-come-first-serve, nearest neighbor, and unidirectional travel as follows:

- First-Come-First-Served (FCFS): the YC serves vehicles in the order of their arrival time at the block. A tie is broken by choosing the nearest job.
- Nearest Neighbor (NN): the YC serves the job that is located nearest to its current location. A tie is broken by choosing a job for the earliest vehicle arrival.
- Uni-directional Travel (UT): The YC travels in one direction and serves vehicles until no more vehicles remain in the direction of the travel, and then the YC starts to travel and serve vehicles in the opposite direction.

FCFS and NN use a single metric – ready time and coordination – to determine the sequence. It means that FCFS ranks the jobs according to their ready time in increasing order, and NN selects the nearest job according to the horizontal coordination of the stacking slot. Due to the single metric, the YC may need to travel from one end to another in FCFS or wait for the job readiness in NN.

UT takes both ready time and distance into consideration. By referring to an example of Fig. 9, assuming that the YC current horizontal coordination is at bay 4 at time 50 and the YC is traveling from left to right, the candidates are the jobs ①, ③ and ④. The

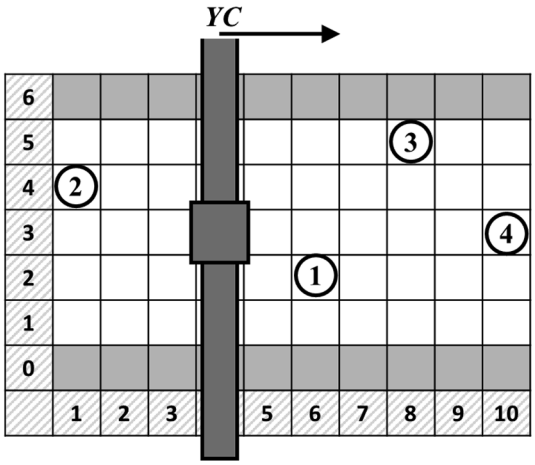

Fig. 9. An illustrated example of UT job sequence.

candidates are evaluated according to the job ready time and estimated travel time from the current YC position to the job. If the YC requires 20, 40, and 70 time-units to move to jobs ①, ③, and ④ respectively, which will be ready at time 140, 110, and 100 for jobs ①, ③, and ④, respectively,

- if serving job ①, the YC can arrive at time 70 but cannot serve until 140;
- if serving job ③, the YC can arrive at time 90 but cannot serve until 110;
- if serving job ④, the YC can serve since 100, but cannot arrive until 120.

Therefore, the YC will first serve job ③, and then job ④ since the YC is traveling from left to right. Once the YC finishes job ④, it will travel in the reverse direction and check the time of the other jobs. UT is a practical and well-known algorithm as it provides high manageability for its deterministic and consistent natures. However, the operational efficiency could be achieved by a more sophisticated algorithm, and the manageability would also be enhanced by introducing advanced supporting systems (e.g., real-time location systems, differential global positioning systems, and Internet-of-Things).

The three benchmarking heuristics cater for the job handling sequence. As parking slot determination has not been explored by literature, this study applies the sequential parking heuristic for the job sequences provided by the heuristics.

5. Numerical experiment

A set of numerical experiments is conducted to examine the efficiency and the quality of the proposed TSA algorithm by comparing the results to the optimum as well as the benchmarking algorithms.

5.1. Experiment setting

There are five methods to be compared, i.e., MIP, TSA, FCFS, NN, and UT. MIP stands for the mixed-integer programming models presented in Sections 3.2 and 3.3. The optimal solutions are obtained by IBM CPLEX 12.7 Solver with fixed computation time. The experiment is conducted for four different block layouts and the five scenarios of jobs. Note that a scenario is defined as the number of jobs to be handled, and each scenario consists of random test cases. The block layout consists of the length (the number of bays) and the width (the number of rows) of a block, which is interpreted as the coverage area of a single YC. For each batch of jobs, the experiment replicates five runs, and each run is for a set of randomly generated types of jobs and target stacking slots for the jobs. The uniform distribution is used to randomize job selection among all slots, and a Bernoulli experiment with the probability of 0.5 is conducted to each selected job to decide the job type (i.e., stacking and unstacking). The ready time is randomly generated by the discrete uniform distribution between 1 and 4, and then multiplied by 10 s. Four different block sizes are considered for a single YC consisting of (15 bays & 6 rows), (15 bays & 10 rows), (20 bays & 6 rows), and (20 bays & 10 rows). In the Figs. 10–14 shown, “B” and “R” are used to abbreviate “bays” and “rows”, respectively.

Regarding the TSA algorithm, the parameters are set as follows: $A = 0.3$; $N_{max} = 100$; $N_{neighbor} = 50$; $N_{no\_change} = 2$; and tabu list size is 30. It is also assumed that the operation time for stacking and unstacking a container is set to 5 s, and the unit travel time for both YC gantry and trolley are equally set to 1 s. The MIP models and heuristic algorithms are implemented by using C# programming language under the development environment of Visual Studio Community 2017 on a computer of Intel i5, 3.3 GHz with 8 GB RAM.

Table 1
MIP and TSA results for non-sharing parking slots.

Batch sizes	MIP-Relax: Obj.	MIP- Relax: Time	MIP: Obj.	MIX: Best bound	MIP: Time**	Gap*	TSA: Avg	TSA: Min	TSA: Max	TSA: Std. dev.	TSA: Avg. Time
5	7	0.06	93	93	3389	0	93	93	93	0	1.89
	14	0.05	95	91	7200	4%	95	95	95	0	1.91
	5	0.05	97	97	7176	0	97	97	97	0	1.89
	15	0.05	95	84	7200	12%	96	96	96	0	2.12
	19	0.06	107	107	3651	0	107	107	107	0	2.09
10	19	0.14	N/A	76	7200	N/A	164.45	162	170	2.65	3.34
	17	0.14	N/A	63	7200	N/A	175	174	176	0.32	3.26
	14	0.13	N/A	65	7200	N/A	178.75	178	185	2.15	3.04
	18	0.14	N/A	74	7200	N/A	172.90	170	175	2.22	2.89
	19	0.16	N/A	66	7200	N/A	170.35	169	173	1.35	3.04
15	10	0.33	N/A	63	7200	N/A	260.55	255	263	2.22	4.25
	19	0.36	N/A	66	7200	N/A	243.50	243	245	0.82	4.42
	17	0.33	N/A	64	7200	N/A	246.05	244	250	1.43	4.35
	19	0.33	N/A	76	7200	N/A	236.10	228	248	4.81	4.16
	14	0.28	N/A	62	7200	N/A	251.40	249	261	3.83	4.28

* Gap (%) = the gap between the best bound and the optimum
** The time is measured in seconds.

5.2. Comparison TSA against MIP models

This subsection examines the proposed MIP models of the subsections 3.3 and 3.4 together with the TSA algorithm and compares their outcomes. The linear relaxation of the proposed model is also examined by relaxing the two binary variables, s_{j_1,j_2} (the sequence variable) and x_{jp} (the parking variable). The batch sizes of 5, 10, and 15 jobs are generated under the block layout (20 bays & 10 rows). The computation time is limited to 2 h for solving the MIP models, and the TSA runs for 20 repetitions. The value of M in the MIP models is not an arbitrary number but tightened by theoretical upper bound (as introduced in Section 3.4) for each scenario. Tables 1 and 2 report the results of MIP-relaxation, MIP, and TSA for non-sharing and sharing parking slots, respectively.

The MIP models could not solve the batch size of 10 jobs in 2 h, whereas the TSA solve the problem with fast response times (i.e., less than 3.5 s) on average. When comparing the TSA makespans against MIP results in a batch size of 5 jobs, the difference is subtle, and the standard deviation of TSA solutions is 0. For the large batch sizes, the TSA provides longer makespans than those of the MIP best bounds. This result indicates that the TSA makespans might be beyond the optimality. Nonetheless, the proposed TSA provides efficient and consistent solutions evidenced by short average computational time (i.e., 5 s at most) and small ranges of standard deviations (i.e., 2.3% at most) even if the batch size becomes large. For practical consideration, the efficiency and consistency of an algorithm are the most important feature as the problem is characterized by the short-term decision-making problem with small batch sizes. The TSA outcomes are fully displayed in Appendix B.

5.3. Comparison TSA against benchmarking algorithms

Another set of experiments is conducted to compare the results (i.e., makespan) of TSA against benchmarking algorithms for both non-shared and shared parking slots over different job batches (i.e., the number of jobs) for the four-block layouts. Since the proposed

Table 2
MIP and TSA results for sharing parking slots.

Batch sizes	MIP-Relax: Obj.	MIP- Relax: Time	MIP: Obj.	MIX: Best bound	MIP: Time**	Gap*	TSA: Avg	TSA: Min	TSA: Max	TSA: Std. dev.	TSA: Avg. Time
5	7	0.03	94	76	7200	19%	92	92	92	0	2.81
	14	0.03	95	75	7200	21%	93	93	93	0	2.05
	5	0.05	97	78	7200	19%	96	96	96	0	1.98
	15	0.05	113	68	7200	4%	95	95	95	0	2.30
	19	0.06	109	107	7200	2%	106	106	106	0	1.99
10	19	0.17	N/A	76	7200	N/A	160.60	158	165	2.28	3.71
	17	0.12	N/A	63	7200	N/A	172.85	170	173	0.67	3.66
	14	0.11	N/A	65	7200	N/A	176.30	176	182	1.34	3.36
	18	0.14	N/A	74	7200	N/A	167	166	170	1.78	3.42
	19	0.17	N/A	72	7200	N/A	167.55	166	170	1.54	3.56
15	10	0.34	N/A	63	7200	N/A	254.95	253	258	2.33	5.14
	19	0.38	N/A	66	7200	N/A	235.35	234	245	3	4.86
	17	0.34	N/A	67	7200	N/A	239.25	239	241	0.55	5.14
	19	0.34	N/A	76	7200	N/A	229.85	223	242	5.24	5.17
	14	0.31	N/A	70	7200	N/A	248.25	245	256	4.44	5.09

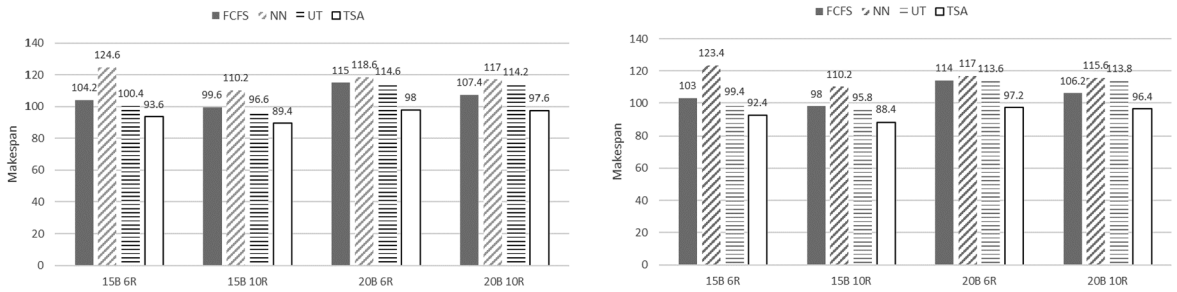

Fig. 10. Results of TSA and benchmarking algorithms for non-shared (left) and shared (right) parking slots for the batch size of 5.

benchmarking algorithms are deterministic approaches and thus always produce consistent solutions, while the TSA contains the randomness, the average outcomes of TSA for 20 repetitions are used for the comparison in this subsection.

Figs. 10–14 show that the TSA results always outperform the results of benchmarking algorithms regardless of block layouts and batch sizes, as the TSA algorithm adopts a search strategy determining the better job sequence and the parking slots for the jobs iteratively where they are interdependent with each other. Compared to the well-known heuristic UT algorithm, which outperforms the other two benchmarking algorithms, the proposed TSA reduces the makespan by 11.1%, 10.3%, 12.4%, 10.6% and 14.5% for the batches of 5, 10, 15, 20, and 25 jobs respectively, over the block layouts when no parking slot is shared among vehicles. When sharing parking slots, the makspans are reduced by 11.4%, 10.3%, 11.4%, 9.2% and 11.4% for the batches of 5, 10, 15, 20, and 25 jobs, respectively. The average reductions compared to UT over batch sizes are 13.3%, 13.7%, 8.8% and 13.5% for the block layouts of (15 bays & 6 rows), (15 bays & 10 rows), (20 bays & 6 rows), and (20 bays & 10 rows) respectively under non-shared parking slots, and 12%, 12.6%, 6.1% and 12.2% respectively under shared parking slots.

A greater effect of Chebyshev movement on travel time reduction of a YC is expected when a greater block size and a smaller batch size are considered. According to the experiment results for batch size of 5 jobs, when no parking slot is shared, the TSA results in a 2.1% increase of makespan per unit block length for the block layout of (15 bays & 6 rows) and (20 bays & 6 rows), a 1.4% increase for (15 bays & 10 rows) and (20 bays & 10 rows), a 0.4% reduction per unit block width for (15 bays & 6 rows) and (15 bays & 10 rows), and a 1.1% reduction for (20 bays & 6 rows) and (20 bays & 10 rows). The average effect of block length and width on the makespan results show that the YC travel time becomes higher by 1.78% when the block length increases by unit bay and lower by 0.75% when the block is wider by unit row. The lengthened block requires the YC to travel a longer distance leading to an increase in the gantry travel time, while the widened block provides an increased chance of travel time reduction for the YC trolley by performing Chebyshev movement. When sharing parking slots, the increase of unit block length results in an increase of the makespan by 1.97% and the increase of unit block width contributes to decreasing the makespan by 0.38% on average. The effect of YC travel time reduction for unit block size (i.e., length and width) is aligned with the results of non-shared parking slots. The weakened effect for unit block width stems from the repeated visits to a parking slot leading to the makespan reduction via Chebyshev, and thus, the widened block naturally mitigates the reduction effect on the travel time.

It is also expected that the makespan is reduced when sharing parking slots instead of non-sharing. When comparing the TSA results obtained under the conditions of non-shared and shared parking slots, the makespans are reduced by 1.11%, 2.15%, 3.12%,4.96% and 7.32% on average for the batches of 5, 10, 15, 20 and 25 jobs respectively. These makespan reductions come from the improved flexibility of the parking slot assignment when they are shared for a job compared to the cases of non-shared parking slots. The flexibility results in travel time reduction of the YC.

5.4. TSA convergence

The algorithm structure of the proposed TSA is based on the tabu search, and hence the computation efficiency needs to be examined to assess the applicability of employing the algorithm in real-time operations.

Fig. 15 plots solution changes of TSA over 50 iterations and observed that the TSA converges solutions efficiently. The TSA nearly

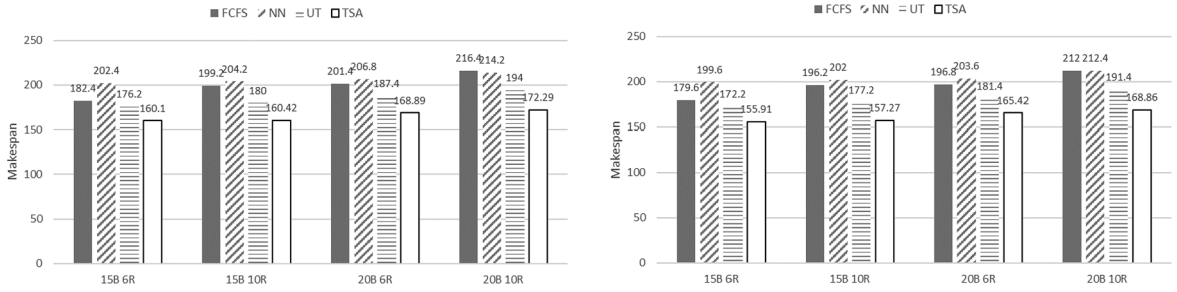

Fig. 11. Results of TSA and benchmarking algorithms for non-shared (left) and shared (right) parking slots for the batch size of 10.

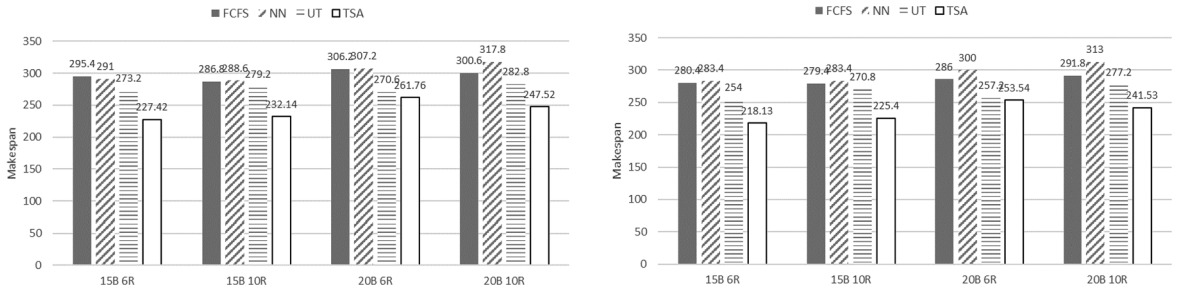

Fig. 12. Results of TSA and benchmarking algorithms for non-shared (left) and shared (right) parking slots for the batch size of 15.

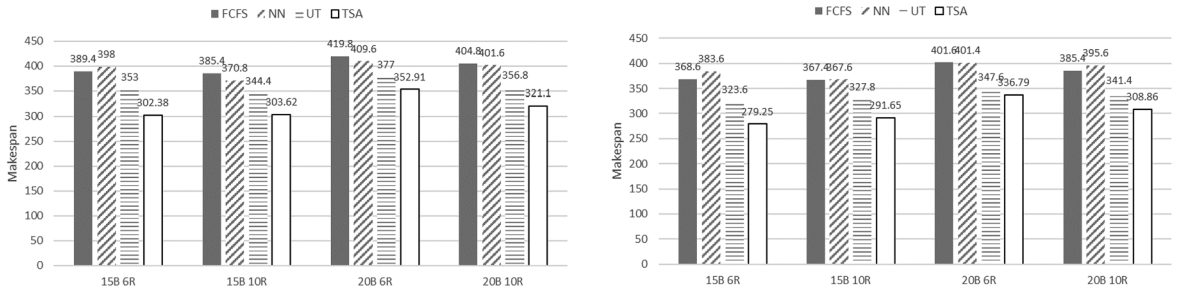

Fig. 13. Results of TSA and benchmarking algorithms for non-shared (left) and shared (right) parking slots for the batch size of 20.

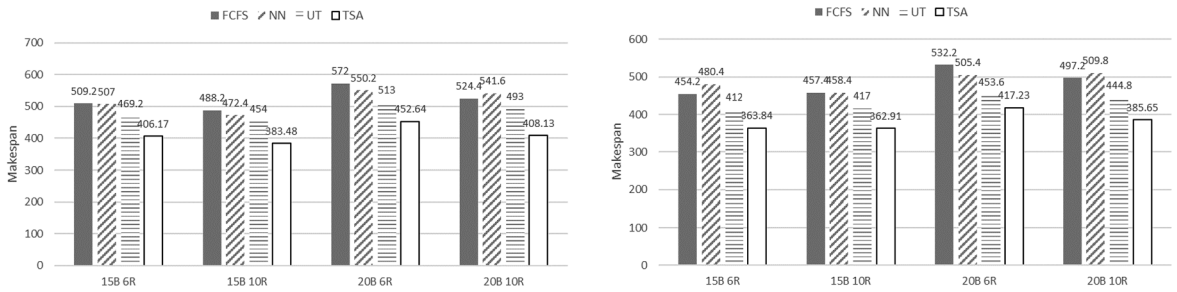

Fig. 14. Results of TSA and benchmarking algorithms for non-shared (left) and shared (right) parking slots for the batch size of 25.

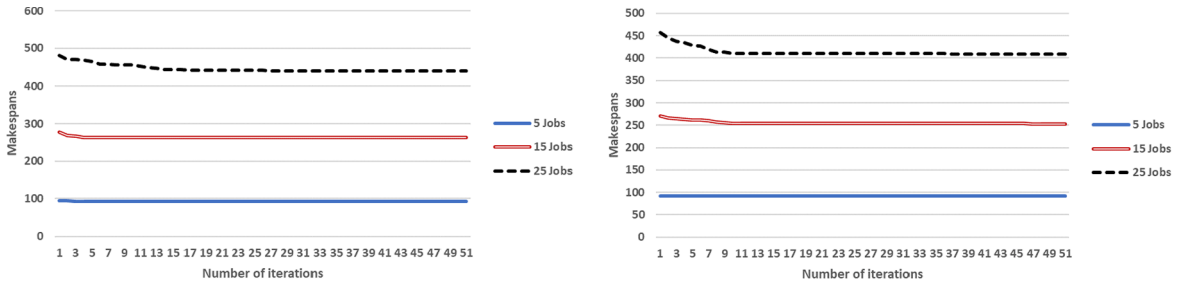

Fig. 15. Solution convergence of TSA for non-shared (left) and shared (right) parking slots under the block layout (20 bays & 10 rows).

converges the solutions less than 15 iterations for non-shared and shared parking slots, respectively, for all the block layouts. The smaller job batch size shows the earlier convergency due to the narrower search space.

The condition of non-sharing parking slots could play a role in reducing the search space. The convergency rate for the cases of non-shared parking slots is faster than those for shared parking slots. The convergency is achieved at the 4th iteration for job batch of size 15 at the block layout of (20 bays & 10 rows) when non-shared parking slots are applied, for example, whereas it is 12th iteration for sharing parking slots. It is noted that the increased variation of TSA for the increased batch size takes negative effect on the solution convergency as shown for the job batch of size 25.

6. Conclusions

This study discovered a scheduling problem required to simultaneously optimize the operation sequence of a YC and parking positions of vehicles for a batch of jobs at a side-loading block layout of a container handling yard. Since a YC travels in the Chebyshev metric, the travel time of the trolley for a job is affected by the vehicle parking position under the given position of the stacking slot. The proposed problem is clearly represented by a MIP model with discretized decision variables. Due to the computation complexity, an efficient heuristic algorithm – TSA (tabu search-based two-stage algorithm) – was developed and compared with three other benchmarking heuristics (i.e., FCFS, NN, and UT). The numerical experiment showed the outperformance of the proposed TSA algorithm in both computational efficiency and solution quality (makespan). The results also showed that the travel time reductions also manifested when sharing parking slots and enlarging the block layout. Since the practicality of the TSA algorithm would be improved with well-managed batch size allocation to the block, it is recommended that the transition between job batches to have a tiny time buffer so that the TSA algorithm is able makes a real-time decision for every small size job batch repeatedly.

In the future study, a dynamic modeling framework could be developed to cater for uncertainties from job ready time, job handling time, reshuffling, etc. Instead of scheduling all jobs together, a rolling horizon fashion could be applied to make a schedule with an improved algorithm. This study would be extended by allowing multiple YCs and other operational elements the decision-making.

CRediT authorship contribution statement

Chenhao Zhou: Methodology, Software, Formal analysis, Investigation, Writing - original draft, Writing - review & editing. **Byung Kwon Lee:** Conceptualization, Formal analysis, Investigation, Writing - original draft, Writing - review & editing. **Haobin Li:** Investigation.

Acknowledgement

This work was supported by a research grant of Singapore Maritime Institute of Singapore.

Appendix A. Myopic decision propositions on determining parking slots

This section proposes several propositions for the four combinations of two consecutive jobs, i.e. (S, S), (S, U), (U, S), and (U, U). Regarding the job combinations (S, U) and (U, S), the optimal parking slots are determined by Corollaries 1.3 and 1.2. For the job combination of (S_1, S_2) and (U_1, U_2), the sets of optimal parking slots for S_2 and U_1 can be determined by the propositions 2 and 3, whereas the parking slots for S_1 and U_2 can be determined by proposition 1. The proofs are simply shown under the assumption that vehicle assess is only available at a side of the block, but it can be relaxed for both sides of the block.

Proposition 1. *If the area surrounded by the partial contour line* (Francis et al., 1992) *by referring to the stacking slot satisfies* $T_v \geq T_h$, *when the gantry movement* (*horizontal movement, H-move*) *and trolley movement* (*vertical movement, V-move*) *are executed in Chebyshev metric, the YC travel time between a parking slot and the stacking slot is dominated by V-move time, i.e.,* $t = T_v$; *else it will be dominated by H-move, i.e.,* $t = T_h$.

Proof. Let us consider the four positions consisting of the trolley position that the previous job was completed $(0, \beta)$, an alternative

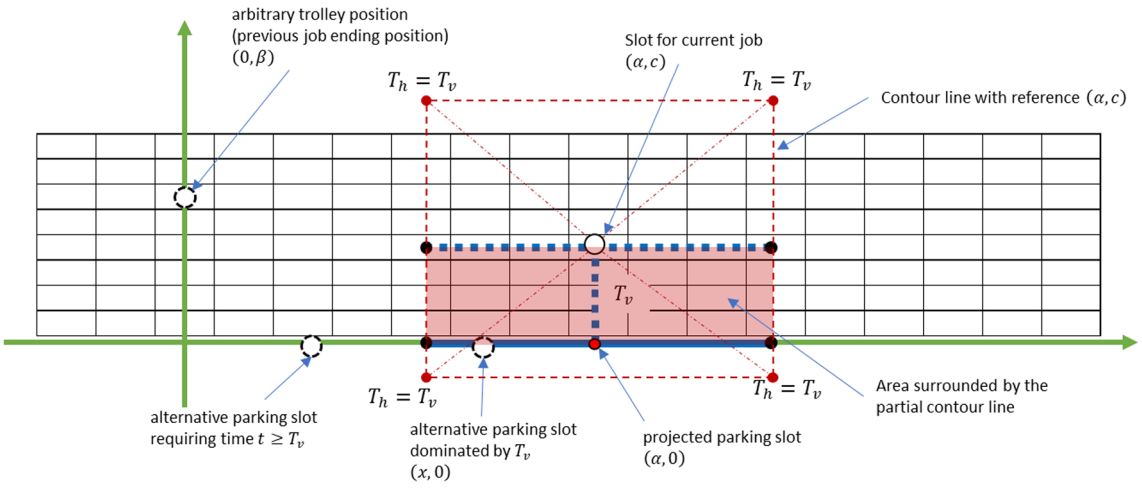

Fig. A1. Travel time dominance for vehicle parking slots under Proposition 1.

parking slot to be determined $(x, 0)$, the stacking slot for the stacking or unstacking job (α, c), and the projected parking slot $(\alpha, 0)$ depicted in Fig. A.1. The contour line with reference to the known (α, c) is limited by a constant k such as $ax(|\alpha - x|\ t_h, |c - y|\ t_v) \le k$, where x and y determine the contour line of gantry and trolley movement directions, respectively, and k should be greater than or equal to the value that the gantry movement time is equal to the time for trolley movement in Chebyshev metric. As it is expected $|\alpha - x|\ t_h \le k$ or $|c - y|\ t_v \le k$, the contour line is estimated as a rectangular form of $\alpha - \frac{k}{t_h} \le x \le \alpha$ and $c - \frac{k}{t_v} \le y \le c$, as the trolley movement is bounded by the block width and the gantry is expected to move in right-side (Fig. A.1).

The trolley speed is typically slower than the gantry speed. It implies $0 \le y \le c$, and thus $k = t_v c$. Consequently $\alpha - \frac{t_v}{t_h} c \le x \le \alpha$. It means that the V-move dominates the travel time when $T_v \ge T_h$. If the trolley speed is faster than the gantry speed, $T_v < T_h$, with $0 \le x \le \alpha$, and then $k = t_h \alpha$. Consequently, $c - \frac{t_h}{t_v}\alpha \le y \le c$. It means the travel time is dominated by H-move. Q.E.D.

Corollary 1.1. *There always exists a set of parking lots having travel time shorter than or equal to that with the parking lot projected from a target slot for a stacking job.*

Proof. By referring to the positions of Fig. A.1, we need to find x satisfying $max(t_h\alpha, t_v\beta) + t_v c \ge max(t_h x, t_v\beta) + max(t_h(\alpha - x), t_v c)$. When the YC travel time is dominated by V-move in accordance with Proposition 1, $t_h(\alpha - x) \ge t_v c$ and $t_h x \le t_v \beta$, the condition becomes $x \le \alpha + \frac{t_v}{t_h}(\beta - c) - max\left(\alpha, \frac{t_v}{t_h}\beta\right)$.

Since $x \le \alpha$ is expected, therefore, all the parking lots on $\left[\alpha + \frac{t_v}{t_h}(\beta - c) - max\left(\alpha, \frac{t_v}{t_h}\beta\right), \alpha\right]$ provide shorter travel time than or equal to that of the projected parking lot. In addition, maximum cardinality of the set of parking lots having shorter travel time than or equal to that of the projected parking lot is estimated as $max\left(\alpha, \frac{t_v}{t_h}\beta\right) - \frac{t_v}{t_h}(\beta - c)$. Q.E.D.

Corollary 1.2. *There always exists a set of parking lots having laden travel time shorter than or equal to that with the projected parking lot, when the YC performs an unstacking job.*

Proof. The empty travel time is not of interest as the empty YC trolley must visit the container slot before performing the laden movement wherever a parking lot for a vehicle is assigned. Regarding the laden travel time, by referring to Fig. A.1, we need to find x satisfying $t_v c = max(t_h\ |x - \alpha|, t_v c)$.

When the YC travel time is dominated by V-move in accordance with Proposition 1, $t_h\ |x - \alpha| \ge t_v c$. It means, therefore, all the parking lots on $\left[\alpha - \frac{t_v}{t_h}c, \alpha + \frac{t_v}{t_h}c\right]$ have trolley travel time shorter than or equal to that of the projected parking lot. Q.E.D.

Corollary 1.3. *For the same laden travel time, there always exists a set of parking lots having empty travel time shorter than or equal to that with the projected parking lot when the YC performs a stacking job.*

Proof. We need to find x satisfying the same laden travel time condition of the Eq. (27). it is obviously expected that $t_h(\alpha - x) \le t_v c$, where $x \le \alpha$. Therefore, all the parking lots on $\left[\alpha - \frac{t_v}{t_h}c, \alpha\right]$ have empty travel time shorter than or equal to the projected parking slot under the same laden travel time. Q.E.D.

Corollary 1.4. *When only considering the empty travel for a stacking job, a set of parking slots having travel time shorter than or equal to that with the projected slot is determined by the trolley position that the YC completed the previous job.*

Proof. The YC empty travel time expected as $max(t_h x, t_v\beta) \le max(t_h\alpha, t_v\beta)$, where $x \le \alpha$, becomes $max(t_h x, t_v\beta) \le t_v\beta$ according to the V-move dominance. If $t_h x \le t_v\beta$, the condition is satisfied. Therefore, all the parking slots on $\left[\frac{t_v}{t_h}\beta, \alpha\right]$ do not have the empty travel time shorter than or equal to that with the projected parking slot, when the YC performs a stacking job. Q.E.D.

Proposition 2. *For the consecutive jobs (S, S) and (U, U), when the travel time is dominated by H-move or V-move for each of the two storage (retrieval) slots as shown in Fig. A.2, a set of optimal parking slots is on the overlapped line estimated from H-move or V-move time dominances.*

Proof. By referring to the quadratic positions of Fig. A.2, the two partial contour lines can be drawn as $\alpha_1 \le x_1 \le \alpha_1 + \frac{k_1}{t_h}$ and $c_1 - \frac{k_1}{t_v} \le y_1 \le c_1$ for (α_1, c_1) and $\alpha_2 - \frac{k_2}{t_h} \le x_2 \le \alpha_2$ and $c_2 - \frac{k_2}{t_h} \le y_2 \le c_2$ for (α_2, c_2), where $\alpha_1 \le \alpha_2$. The parking slot is in between α_1 and α_2. The possibility of overlapping the two partial contour lines can be minimized when the H-move dominance for (α_1, c_1) resulting in $\alpha_1 \le x_1 \le 2\alpha_1$, $k_1 = t_h\alpha_1$, and $c_1 - \frac{t_h}{t_v}\alpha_1 \le y_1 \le c_1$, and the V-move dominance for (α_2, c_2) resulting in $0 \le y_2 \le c_2$, $k_2 = t_v c_2$, and thus $\alpha_2 - \frac{t_v}{t_h}c_2 \le x_2 \le \alpha_2$. The length of the possibly overlapped parking slots is estimated as $\alpha_1 - \left(\alpha_2 - \frac{t_v}{t_h}c_2\right)$.

Since that the length estimation is non-negative, $2\alpha_1 + \frac{t_v}{t_h}c_2 \ge \alpha_2$. Therefore, there is a set of parking slots on the overlapped partial contour lines for the consecutive jobs (S, S) and (U, U) minimizing YC travel time. Q.E.D.

Proposition 3. *For the consecutive jobs (S, S) and (U, U), when the extra travel time is needed to be dominated by H-move or V-move for each of the two storage (retrieval) slots as shown in Fig. A.3, a set of optimal parking slots is on the line in between the boundaries of H-move or V-move.*

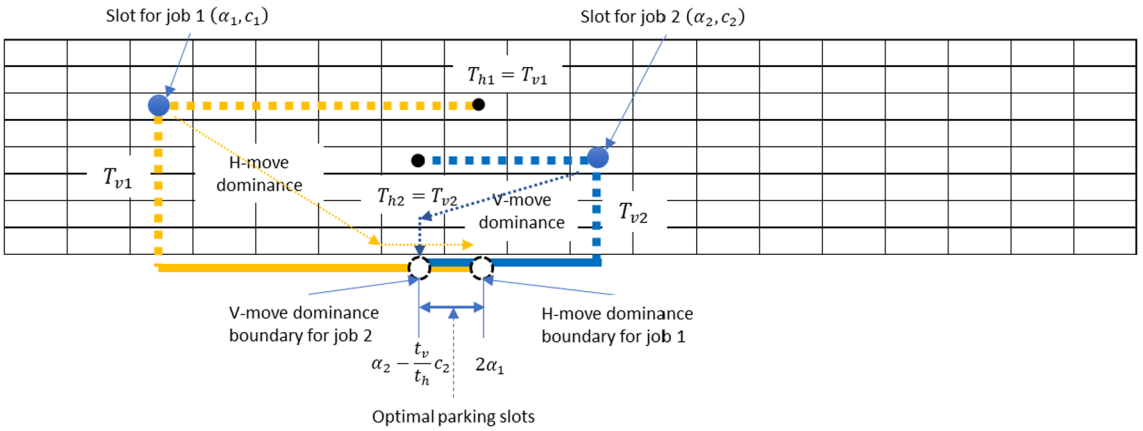

Fig. A2. Travel time dominance for vehicle parking slots under Proposition 2.

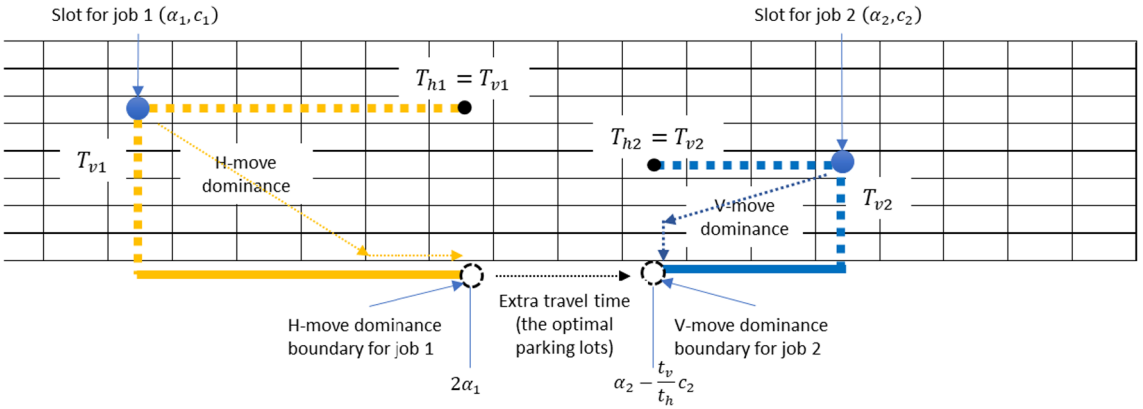

Fig. A3. Travel time dominance for vehicle parking slots under Proposition 3.

Proof. The YC needs to spend extra travel time that is not covered by H-move and/or V-move dominance when the travel distance between the two storage (retrieval) slots are far away, $2\alpha_1 + \frac{t_v}{t_h} c_2 < \alpha_2$. The necessary extra travel time is from the boundary of H-move for (α_1, c_1) should not beyond the boundary of V-move for (α_2, c_2), as any parking slot belonging to the partial contour line estimated from (α_2, c_2) is dominated by the V-move. The extra travel time is hence represented by $t_h\left(\alpha_2 - \frac{t_v}{t_h} c_2 - 2\alpha_1\right)$.

It means that the set of parking slots in between the two partial contour lines still allows the H-move or V-move dominances for (α_1, c_1) and (α_2, c_2) and minimize the YC travel time for consecutive jobs (S, S) and (U, U). Q.E.D.

Appendix B. Extended TSA results

This section examines the effect of aspiration and diversification strategies of the proposed TSA algorithm via comparing the experiment results of TSA, TSA without diversification, TSA without diversification and aspiration condition (2), and TSA without heuristic initial solutions. No parking slot is shared in this experiment. Table B.1 displays the comparison outcomes for the batch sizes of 5 and 25 jobs under the block layout of (20 bays & 10 rows). The experiment runs 20 repetitions for each instance.

As shown in the results (Table B.1), in a small batch size (5 jobs), there is no difference among various variants of TSA algorithms. The effects of diversification and aspiration strategies with criterion (2) are somewhat contributable to makespan reduction (i.e., 0.37% on average) with small supplement (i.e., 1.26 s on average) of computational time in a large batch size (25 jobs). As for the effect of heuristic initial solutions, the random sampling has little chance of discriminating the quality of initial solutions as much as that of the benchmarking algorithms due to the limited job sequence combinations.

This section also displays the full experiment results (Tables B.2–B.5) providing efficiency and consistency of the TSA algorithm for the 20 runs of each scenario.

Table B1
Experiment results for heuristic initial solutions under block layout of (20 bays & 10 rows).

	TSA					TSA without diversification				
	Avg. value	Min. value	Max. value	Standard deviation	Average time	Avg. value	Min. value	Max. value	Standard deviation	Average time
5Jobs_1	93.00	93	93	0	1.89	93.00	93	93	0	1.39
5Jobs_2	95.00	95	95	0	1.91	95.00	95	95	0	1.46
5Jobs_3	97.00	97	97	0	1.89	97.00	97	97	0	1.43
5Jobs_4	96.00	96	96	0	2.12	96.00	96	96	0	1.35
5Jobs_5	107.00	107	107	0	2.09	107.00	107	107	0	1.35
25Jobs_1	428.95	420	438	5.10	5.51	430.40	422	444	6.64	4.11
25Jobs_2	410.60	402	421	4.83	5.40	411.45	400	422	5.26	4.09
25Jobs_3	392.85	384	401	4.66	5.40	394.05	385	404	4.58	4.05
25Jobs_4	399.40	392	406	3.82	5.42	400.80	392	411	4.92	4.20
25Jobs_5	408.85	403	417	4.09	5.40	410.35	403	420	4.09	4.22
	TSA without diversification and aspiration condition (2)					TSA without heuristic initial solutions				
	Avg. value	Min. value	Max. value	Standard deviation	Average time	Avg. value	Min. value	Max. value	Standard deviation	Average time
5Jobs_1	93.00	93	93	0	1.75	93.00	93	93	0	1.83
5Jobs_2	95.00	95	97	0.62	1.72	95.00	95	97	0.62	1.84
5Jobs_3	97.00	97	97	0	1.58	97.00	97	97	0	1.93
5Jobs_4	96.00	96	96	0	1.61	96.00	96	96	0	2.04
5Jobs_5	107.00	107	107	0	1.48	107.00	107	107	0	1.76
25Jobs_1	431.15	424	442	4.76	4.16	432.15	423	449	6.52	5.71
25Jobs_2	411.90	404	422	4.44	4.22	411.85	403	425	6.72	5.30
25Jobs_3	396.60	388	412	5.60	4.10	394.75	385	415	6.39	5.42
25Jobs_4	399.55	391	407	4.48	4.26	395.80	383	407	6.28	5.98
25Jobs_5	410.20	405	417	3.05	4.29	409.85	399	417	4.98	5.59

Table B2
Experiment results for block layout of (15 bays & 6 rows).

Parking lots	Batch sizes	TSA makespans				Computational time (sec.)
		Average	Minimum	Maximum	Standard deviation	
Non-sharing	5	93.60	93.60	93.60	0	1.35
	10	160.10	158	163.40	1.98	2.16
	15	227.42	224.20	232.40	2.63	9.57
	20	302.38	296	312.20	3.94	3.43
	25	406.17	396.80	415.60	5.08	4.11
Sharing	5	92.40	92.40	92.40	0	1.46
	10	155.91	154.6	158.60	1.55	2.41
	15	218.13	215.4	222.40	2.14	11.14
	20	279.25	275.6	285	2.64	4.23
	25	363.84	358.8	369.60	3	5.74

Table B3
Experiment results for block layout of (15 bays & 10 rows).

Parking lots	Batch sizes	TSA makespans				Computational time (sec.)
		Average	Minimum	Maximum	Standard deviation	
Non-sharing	5	89.40	89.40	89.40	0	1.33
	10	160.42	158.80	163.40	1.68	2.25
	15	232.14	227.20	238.40	3.40	3.07
	20	303.62	298.80	311.80	3.58	3.65
	25	383.48	376.40	393.80	4.59	6.53
Sharing	5	88.40	88.40	88.40	0	1.38
	10	157.27	155.80	161.60	1.91	2.57
	15	225.40	220.60	231.60	3.13	3.59
	20	291.65	287	298.60	3.01	4.51
	25	362.91	355.20	372.40	4.16	8.63

Table B4
Experiment results for block layout of (20 bays & 6 rows).

Parking lots	Batch sizes	TSA makespans				Computational time (sec.)
		Average	Minimum	Maximum	Standard deviation	
Non-sharing	5	98	98	98	0	5.21
	10	168.89	167.80	172.40	1.38	3.70
	15	261.76	259	266.20	2.45	6.71
	20	352.91	347.20	360.80	4.03	4.42
	25	452.64	441.40	465.20	5.96	5.08
Sharing	5	97.20	97.20	97.20	0	5.56
	10	165.42	164.60	168.60	1.25	4.18
	15	253.54	249.60	256.20	2.05	7.62
	20	336.79	331.60	340.80	2.63	5.59
	25	417.23	411.80	422.80	3.30	6.55

Table B5
Experiment results for block layout of (20 bays & 10 rows).

Parking lots	Batch sizes	TSA makespans				Computational time (sec.)
		Average	Minimum	Maximum	Standard deviation	
Non-sharing	5	97.60	97.60	97.60	0	1.98
	10	172.29	170.60	175.80	1.74	3.11
	15	247.52	243.80	253.40	2.62	4.29
	20	321.10	317	325.40	2.56	4.86
	25	408.13	400.20	416.60	4.50	5.43
Sharing	5	96.40	96.40	96.40	0	2.22
	10	168.86	167.20	172	1.52	3.54
	15	241.53	238.80	248.40	3.11	5.08
	20	308.86	304.40	316	3.40	5.81
	25	385.65	380.60	392.20	3.03	7.09

References

Basu, S., Sharma, M., Ghosh, P.S., 2017. Efficient preprocessing methods for tabu search: an application on asymmetric travelling salesman problem. INFOR: Inf. Syst. Operational Res. 55 (2), 134–158.

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

Carlo, H.J., Vis, I.F.A., Roodbergen, K.J., 2014. Storage yard operations in container terminals: Literature overview, trends, and research directions. Eur. J. Oper. Res. 235, 412–430.

Chen, L., Bostel, N., Dejax, P., Cai, J., Xi, L., 2007. A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. Eur. J. Oper. Res. 181 (1), 40–58.

Chen, L., Langevin, A., Lu, Z., 2013. Integrated scheduling of crane handling and truck transportation in a maritime container terminal. Eur. J. Oper. Res. 225 (1), 142–152.

Dorndorf, U., Schneider, F., 2010. Scheduling automated triple cross-over stacking cranes in a container yard. OR Spectrum 32 (3), 617–632.

Edelkamp, S., Schrödl, S., 2012. Selective Search. In: Edelkamp, S., Schrödl, S. (Eds), Heuristic Search: Theory and Applications, Morgan Kaufmann, Waltham, MA, pp. 633-669. https://doi.org/10.1016/B978-0-12-372512-7.00014-6.

Gharehgozli, A.H., Laporte, G., Yu, Y., de Koster, R., 2014a. Scheduling twin yard cranes in a container block. Transportation Sci. 49 (3), 686–705.

Gharehgozli, A.H., Roy, D., de Koster, R., 2015. Sea container terminals: New technologies and OR models. Maritime Econ. Logistics 18, 103–140.

Gharehgozli, A.H., Vernooij, F.G., Zaerpour, N., 2017. A simulation study of the performance of twin automated stacking cranes at a seaport container terminal. Eur. J. Oper. Res. 261 (1), 108–128.

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

Glover, F., 1990. Tabu search: A tutorial. Interfaces 20 (4), 74–94.

Hu, Z.-H., Sheu, J.-B., Luo, J.X., 2016. Sequencing twin automated stacking cranes in a block at automated container terminal. Transportation Res. Part C: Emerging Technol. 69, 208–227.

Kim, K.H., Kim, K.Y., 1999. An optimal routing algorithm for a transfer crane in port container terminals. Transportation Sci. 33 (1), 17–33.

Kim, K.H., Lee, K.M., Hwang, H., 2003. Sequencing delivery and receiving operations for yard cranes in port container terminals. Int. J. Prod. Econ. 84 (3), 283–292.

Kim, K.Y., Kim, K.H., 2003. Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals. Naval Res. Logistics (NRL) 50 (5), 498–514.

Lee, B.K., Kim, K.H., 2010. Comparison and evaluation of various cycle-time models for yard cranes in container terminals. Int. J. Prod. Econ. 126 (2), 350–360.

Lehnfeld, J., Knust, S., 2014. Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. Eur. J. Oper. Res. 239, 297–312.

Liu, G.Y., He, Y., Fang, Y., Qiu, Y., 2003. A novel adaptive search strategy of intensification and diversification in tabu search. In: Proceedings of the 2003 International Conference on Neural Networks and Signal, Nanjing, China, 14-17 December 2003, pp. 428–431.

Lu, Y., Le, M., 2014. The integrated optimization of container terminal scheduling with uncertain factors. Comput. Ind. Eng. 75, 209–216.

Luo, J., Wu, Y., 2015. Modelling of dual-cycle strategy for container storage and vehicle scheduling problems at automated container terminals. Transportation Res. Part E: Logistics Transportation Rev. 79, 49–64.

Malek, M., Guruswamy, M., Pandya, M., Owens, H., 1989. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. Ann. Oper. Res. 21 (1), 59–84.

Saini, S., Roy, D., de Koster, R., 2017. A stochastic model for the throughput analysis of passing dual yard cranes. Comput. Oper. Res. 87, 40–51.

Salhi, S., 2002. Defining tabu list size and aspiration criterion within tabu search methods. Comput. Oper. Res. 29 (1), 67–86.
Speer, U., Fischer, K., 2016. Scheduling of different automated yard crane systems at container terminals. Transportation Sci. 51 (1), 305–324.
Vis, I.F., Carlo, H.J., 2010. Sequencing two cooperating automated stacking cranes in a container terminal. Transportation Sci. 44 (2), 169–182.
Francis, R.L., McGinnis, L.F., White, J.A., 1992. Facility Layout and Location : An Analytical Approach, 2nd eds. Prentice Hall, New Jersey.
Wu, Y., Li, W., Petering, M.E., Goh, M., de Souza, R., 2015. Scheduling multiple yard cranes with crane interference and safety distance requirement. Transportation Sci. 49 (4), 990–1005.
Zheng, F., Man, X., Chu, F., Liu, M., Chu, C., 2018. Two yard crane scheduling with dynamic processing time and interference. IEEE Trans. Intell. Transp. Syst. 19 (12), 3775–3784.