Hindawi
Journal of Advanced Transportation
Volume 2022, Article ID 5738254, 17 pages
https://doi.org/10.1155/2022/5738254

Research Article

Positioning a Handshake Bay for Twin Stacking Cranes in an Automated Container Terminal Yard Block

Zhi-Hua Hu,[1,2] **Xi-Dan Tian**,[1,2] **Yu-Qi Yin**,[2] **and Chen Wei**[2]

[1]*Qingdao Institute, Shanghai Maritime University, Shanghai 201306, China*
[2]*Logistics Research Center, Shanghai Maritime University, Shanghai 201306, China*

Correspondence should be addressed to Zhi-Hua Hu; zhhu@shmtu.edu.cn

Received 6 December 2019; Revised 12 December 2021; Accepted 27 December 2021; Published 18 January 2022

Academic Editor: Vincent F. Yu

At automated container terminals (ACTs), twin automated stacking cranes (ASCs) can carry out the tasks—store and retrieve containers simultaneously in a yard block using a handshake bay, where a primary ASC stacks the container at the handshake bay and the other crane carries it to the destination bay. Although the handshake bay increases the degree of crane utilization, the ASCs will interfere with each other at the bay, decreasing the stacking efficiency. This study formulates a mixed-integer linear program (MILP) to position the handshake bay and simultaneously schedule the twin ASCs to minimize the tasks' makespan. The proposed formulation considers the safe time interval to avoid crane collisions during adjacent crane movements. To solve the model, we developed a random-key genetic algorithm with a priority-based decoding scheme to optimize the task sequences and tasks assigned to the cranes. The priority-based GA can always generate feasible solutions by ranking the container-handling tasks. Numerical experiments prove that the safe temporal interval affects the makespan and the handshake bay's position. An optimal handshake bay reduces 35% of the makespan compared with a nonoptimal bay, and the proposed algorithm is competitive compared with the on-the-shelf MILP solver and can solve medium- and large-scale instances in short computing time with gaps lower than 5% compared with ideal solutions.

1. Introduction

Globalization promotes container transportation through seaborne shipping that undertakes 90% of the world trade [1]. In 2015–2017, the seaborne trade volume by container ships increased from 1,660 million tons to 1,834 million tons [2]. Many containers are daily delivered by a large number to container terminals [3]. The maritime transportation flows and container demand have increased over time, although the COVID-19 pandemic may slow down [4]. The increasing volume in the container shipping industry exerts much pressure on the container terminals [5]. Shanghai Port, as the global busiest container terminal, achieved a throughput of 42.01 million twenty-foot equivalent units (TEUs, a 20 ft × 8 ft × 8.5 ft container) in 2018 [6], which indicates that the terminal had to handle 110 thousand TEUs daily. In response to the massive throughput challenge, container terminals have to increase the efficiency of terminal operations, especially by utilizing advanced equipment [7]. Automated stacking crane (ASC) is such a kind of equipment that improves stacking productivity in automated container terminals (ACTs) [8]. To improve the operation efficiency of the yard, the twin automated stacking cranes are used to store and retrieve containers simultaneously with the help of the handshake bay. The prime stacking crane stacks the container into the handshake bay, and the cooperative crane transports the container to the target bay. The ASCs stack and retrieve containers in the stacking blocks under the control of terminal operating systems. The ACT operators have developed the automation technologies and operation optimization tools to schedule ASCs efficiently. Many containers are daily delivered by a large number of external trucks (ETs).

The twin ASC solution examined used two cranes that conduct container storage and retrieval tasks cooperatively in a single block [9–11]. The ASCs store and retrieve the

containers through the transfer points (for input/output (I/O)) at the ends of blocks. Generally, the two ASCs can handle containers simultaneously in the block but cannot pass each other. Figure 1 depicts a typical twin ASC layout used at Shanghai Yangshan ACT [12]. At the seaside I/O point, an ASC picks up the storage containers or drops off the retrieval containers onto the internal automated guided vehicles (AGVs). At the landside I/O point, the other ASC exchanges the containers with external container trucks. An ASC cannot crossover the other and can only serve the I/O point at its side. The conflict is notable when the twin cranes handle a batch of tasks via only the seaside or landside I/O point because the crane at the other side is blocked to reach the point and has to stay idle.

When the twin ASCs are independent, like two crossable ASCs, they can be scheduled in two separate programs. Each ASC operates according to a batch of container-handling tasks with constraints (e.g., time windows and sequence-dependent orders). While the two ASCs use the same track/orbit, they cannot cross each other, so they must synchronize along the orbit. We call this constraint time-space synchronization that demands a safety clearance distance between two ASCs. This study introduces the handshake bay as a critical bay between the two ASCs for the following reasons. First, only one ASC can simultaneously handle container storage/retrieval tasks at the handshake bay. Then, we can represent a container-handling task by two subtasks. An ASC retrieves and stores the container at the handshake bay, and then, the other ASC retrieves the container from the handshake bay and stores it to its destination. So, the ASCs must synchronize the retrieval/storage operations at the handshake bay. The handshake bay can be a fixed bay of the block or can be dynamically determinant. Its position is critical for improving efficiency. Figure 2 depicts the handshake bay's operations: a crane stacks container to the handshake bay, while another crane carries it to the destination. To store a container in a block bay, the seaside ASC picks it up, carries it to the handshake bay, drops the container, and then returns to the seaside I/O point for incoming container-handling tasks; meanwhile, the landside ASC moves to the handshake bay, picks up the container, carries the container, and finally drops it to its destination bay. By utilizing the handshake bay, the ASCs reduce the idle time and potentially save the makespan. However, additional operations are required for the cranes to hand over containers. Thus, optimizing the handshake bay and twin ASC scheduling solutions is crucial.

Under optimizing operational efficiency for a batch of storage/retrieval tasks in an ACT yard block, some studies developed models to minimize the makespan (the completion time of the last task) [13, 14]. We jointly determine the task sequences of the cranes and the location of the handshake bay in a block. The tasks whose origin and destination are the handshake bay that are operated by two ASCs sequentially.

This study contributes to three streams of studies in the literature. Firstly, we developed a mixed-integer linear program to decide the optimal handshake bay coupled with the ASC scheduling problem under synchronization constraints.

Figure 1: Twin ASC stacking blocks at Yangshan automated container terminal.

Pioneering studies aim at these problems using methods based on simulation and heuristics [14]. We developed solutions based on formal MILP models. Secondly, we devised a priority-based GA [15, 16] for the handshake bay positioning problem. GA provides us great flexibility to hybridize with domain-dependent heuristics to efficiently implement the specific problem. The priority-based GA can always generate feasible solutions by ranking the container-handling tasks. Finally, we compared the MILP (solved by an on-the-shelf solver), a greedy insert algorithm, and the GA to examine their performances in small-, medium-, and large-scale instances and validate the efficiency of the proposed methods.

Compared with other literature studies, this study takes the handshake bay as the decision variable and considers the cooperation between the handshake bay and two ASCs to avoid conflict and improve the efficiency of ASC. We design the task-based priority coding and the ASC-based priority coding rules in terms of algorithm.

In the rest of the study, Section 2 is a literature review. Section 3 illustrates the research problem and formulates the mathematical model. Section 4 presents an insert algorithm and a priority-based GA to solve the model efficiently. Section 5 discusses the experimental results by examining the performances of the proposed algorithms. Section 6 is the conclusion and highlights further research directions.

2. Literature Review

2.1. Stacking Crane Scheduling. The studies on ASC scheduling generally include storage optimization, retrieval optimization, pre-marshaling, and their combinations [17].

Rail-mounted gantry (RMG) cranes and rubber-tired gantry (RTG) cranes are essential equipment used in container yard blocks. Their efficiency is the performance bottleneck of container terminals [11]. Ng and Mak [18] devised a branch-and-bound algorithm to minimize the waiting time of tasks for a single-yard crane. Gharehgozli et al. [19] proposed a two-phase method for a single-yard crane optimization problem: first, they formulated the original scheduling problem into an assignment model; then, they developed a branch-and-bound algorithm to solve the problem. Sharif et al. [20] proposed an agent-based approach for scheduling and deploying multiple yard cranes that can travel among blocks. They examined various synchronization rules for crane assignments and schedules. Galle et al. [21] scheduled multiple yard cranes simultaneously by considering future relocations. They devised a

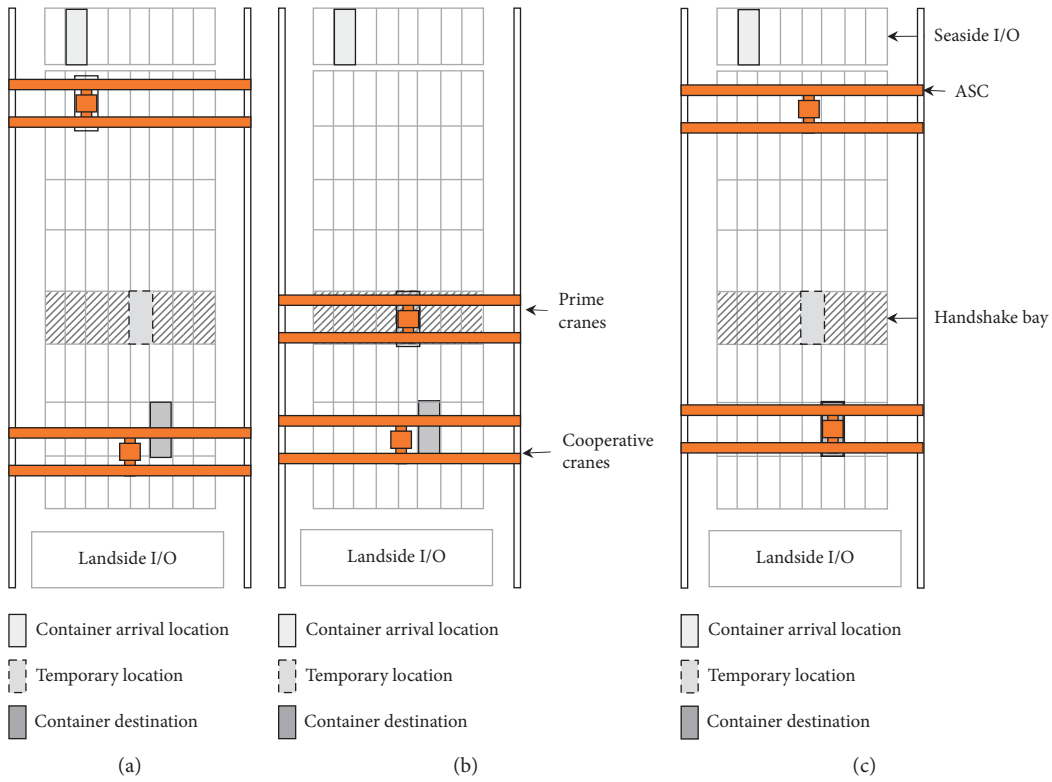

Figure 2: Twin ASCs' operations for a storage task with handshake bay. (a) Seaside ASC picks up imported container from the I/O point. (b) Seaside ASC drops off the container at the handshake bay. (c) Landside ASC carries the container to the destination.

MILP to determine the task sequences and solved the model using a relaxation method and a local search algorithm. Considering inter-crane interference, clearance distances, and time windows for simultaneous stacking and retrieving containers by two cranes, Li et al. [22] also developed a MILP that is solved by rolling horizon heuristics. Ng [23] scheduled multiple yard cranes that shared one bidirectional traveling lane and handled tasks with different ready times by a MILP and dynamic programming considering inter-crane interferences. Fotuhi et al. [24] introduced an agent-based approach to sequence multiple cranes to minimize the waiting time of external trucks. They used reinforcement learning to optimize the operations considering historical data. Sha et al. [25] scheduled multiple cranes with turning distance and operation rules under energy minimization. He et al. [26] studied the energy-saving solution for scheduling yard cranes using an integrated simulation method.

ASC is a kind of unmanned RMGs that uses automation technologies to control the crane movements and container-handling operations. Gharehgozli and Zaerpour [27] scheduled a single ASC for stacking outbound containers by considering an alternative stacking policy (stacking containers of different types at the same row to increase stacking space utilization while increasing the complexity of crane scheduling).

The twin ASC configuration introduces difficulties in modeling the cranes' interferences, affecting the algorithm development. Boysen et al. [28] developed a classification scheme for the interferences between two cranes. Some pioneering studies formulated the twin ASC scheduling problem as formal or descriptive MILPs while developing different solution algorithms, e.g., graph-based polynomial algorithm [29], rule-based heuristics [30], reward-based heuristics [31], dynamic programming, and beam search algorithm [32].

As discussed in Section 1, the handshake bay will affect the solutions for ASC task sequences. Carlo and Martínez-Acevedo [8] optimized the handshake bay by considering fourteen priority rules. Jaehn and Kress [33] scheduled the twin cranes in a single block considering handshakes based on discrete-time model and heuristic algorithms. Gharehgozli et al. [14] studied the effect of handshake bay on the makespan and waiting time in a twin ASC configuration and proposed several priority-based operations rules to simulate the stacking operations. The handshake bay is predetermined (as a constant) in the proposed models in all these studies.

Table 1 summarizes the studies on stacking crane scheduling with four aspects: synchronization (Syn), handshake (HS), model type, and solution algorithm. Most studies deal with the synchronization, while handshake bay is ignored, which simplifies the ASC scheduling problem. Moreover, most algorithms are designed based on task priority. We consider the synchronization of handshake bay and ASCs, taking the handshake bay as a decision variable and transferring the moving interference between double stacker cranes to handshake bay position.

Table 1: Pioneering studies on scheduling yard cranes in container blocks.

Paper	Syn	HS	Model	Solution methods
Gharehgozli and Zaerpour [27]	No	No	DTIP	Simulated annealing
Ng and Mak [18]	No	No	IP	Branch-and-bound algorithm
Gharehgozli et al. [19]	No	No	IP	Branch-and-bound algorithm
Ng [23]	Yes	No	IP	Dynamic program-based heuristic
Galle et al. [21]	Yes	No	IP	Approximate algorithm
Li et al. [22]	Yes	No	MILP	Rolling horizon heuristics
He et al. [26]	Yes	No	MILP	GA and particle swarm algorithm
Briskorn et al. [30]	Yes	No	MILP	Bucket brigade algorithm
Sha et al. [25]	Yes	No	MILP	Scheduling algorithm
Zhang et al. [34]	Yes	No	MILP	Approximate approach
Sharif et al. [20]	Yes	No	Simulation	ABM with DP
Fotuhi et al. [24]	Yes	No	Simulation	ABM; reinforcement learning
Briskorn and Angeloudis [29]	Yes	No	Simulation	Graphical model and polynomial algorithms
Ozcan and Eliiyi [31]	Yes	No	Simulation	Reward-based stacking algorithm
Kress et al. [32]	Yes	No	Simulation	Bounded DP
Jaehn and Kress [33]	Yes	Yes	DTIP	Bucket brigade algorithm
Carlo and Martínez-Acevedo [8]	Yes	Yes	MILP	Priority rules
Gharehgozli et al. [14]	Yes	Yes	Simulation	Priority and operation rules

Note. Syn = synchronization, HS = handshake, IP = integer program, DTIP = discrete-time IP; ABM = agent-based modeling; DP = dynamic programming.

2.2. Synchronization Constraints in Scheduling Problems. In routing and scheduling problems, synchronization occurs among devices' operations due to spatial, temporal, and capacity constraints [35–37]. The pioneering studies investigated in Section 2.1 generally conceptualized the interference considered in twin ASC scheduling as synchronization constraints that are generally similarly considered in vehicle routing problems (VRPs) [38]. Drexl [39] reviewed the VRPs with multiple synchronization constraints among distribution tasks, cargo handling operations, vehicle movements, cargo loads, and service resources. Schiffer et al. [40] reviewed VRPs and location routing problems with intermediate stops, replenishment, and unloading, where the routes are synchronized at locations. Gschwind and Irnich [41] synchronized two sequential operations by dynamic time windows in a dial-a-ride scheduling problem. Crainic et al. [42] synchronized the movements of urban vehicles and city freighters at logistic satellites in an integrated scheduling and resource management problem. The space-time synchronization constraints are formulated among operations in the two-tiered city logistic system. Agatz et al. [43] synchronized the vehicles (trucks here) and drones at the drone departure and arrival locations in a VRP with a drone to make deliveries. They formulated the synchronization problem as a MILP and solved the model by a route-first cluster-second heuristic algorithm. Derigs et al. [44] studied the synchronization effect of time windows and load transfers between trucks and trailers on the performance of routes. Belenguer et al. [45] synchronized the trucks and trailers at satellite facilities using heuristics as solution methods. Chao [46] synchronized main and sub-routes in the two-stage VRPs using cluster-first-based heuristics and a Tabu search algorithm.

Besides, synchronization is also a typical concept and constraint in many domains, e.g., chaotic systems [47], multi-agent systems [48], and machine learning [49]. Synchronization is an emergent property in a broad range of dynamical systems. Synchronization may be explained and formulated differently in other fields. However, synchronization generally represents the closeness relations in time or space dimensions in transportation and logistics.

2.3. Incremental Contribution. This study contributes to the related studies in the following aspects. First, we conceptualize the interferences between the twin ASCs as combined time-space and critical resource synchronizations. Based on the pioneering studies in Table 1, as an incremental contribution, we developed a continuous-time MILP to optimize the handshake bay and sequence stacking tasks for the ASCs. Second, the proposed MILP integrates the handshake bay's decisions, the two ASC scheduling solutions, and the crane interference avoidance used to synchronize the crane operations at the handshake bay. Based on the pioneering studies on ASC scheduling with handshake bays, this study combines the routing and handshake bay positioning problems in a single formal MILP. Finally, most conventional heuristics usually conduct a local search. The use of evolution operators makes GA very effective in performing a global search. Using the MILP as a base, we develop a random-key genetic algorithm with ASC priority and task priority coding scheme to study the impacts of handshake bay positioning on twin ASC scheduling solutions. The collision of synchronous operation is avoided from the model and algorithm, which is more practical.

3. Problem Description and the Model

3.1. Problem Description. This study investigates the problem of positioning a handshake bay and scheduling twin ASCs for a batch of containers' arrival at one I/O point in a yard block. The block consists of n_b bays, several rows, and two I/O points at its two ends. We denote the bays by a set, e.t., $\{1, 2, \ldots, n_b\}$. We denote the set of stacking bays by $B = \{0, 1, \ldots, n_b\}$. Here, 0 refers to a virtual bay representing

the I/O point of the arrival containers. The set of the twin ASCs is denoted by $K = \{p, c\}$, where p represents the crane near the I/O point of arrival containers and c represents the other crane. The ASC moves along the bays at a speed of T^m per bay and uses a trolley to grasp the containers. The time of hoist/dropping off one container is denoted by T^h.

A set of tasks are available at the beginning of the scheduling, denoted by $N = \{1, 2, \ldots, n\}$. Each task specifies a container movement by an original location (bay) O_i and a destination (bay) D_i, where $O_i, D_i \in B$ for all $i \in N$. We can determine the data (O_i, D_i) in advance. The primary purpose of this study is to optimize a handshake bay denoted by w in B, namely $w \in B$. The container carried across the handshake bay is handled by the twin ASCs sequentially: the crane p carries the container from its original location to the handshake bay, and then, the crane c takes it from the handshake bay to the destination.

The objective of scheduling the twin ASCs is to minimize the makespan, represented by the last task's completion time. The decisions involve positioning the handshake bay w and the handling sequences of ASCs. The crane p drops a container to the handshake bay before the crane c can pick up the container. The two cranes can use the handshake bay exclusively as it is a critical resource (conceptualized as critical resource synchronization). Besides, considering the crane interference, the sequences of twin ASCs have to keep a safe time interval with each other to avoid crane collision (conceptualized as time-space synchronization). Figure 3 depicts these two kinds of synchronization. Table 2 summarizes the notations mentioned above.

3.2. Mathematical Model. We formulate the handshake bay positioning and twin ASC scheduling problem as follows. The binary variable x_{ijk} formulates the tasks' allocation to cranes and their sequence: it equals 1 if ASC k handles task i directly before task j; otherwise, it is 0. The notations s_{ik} and e_{ik} denote the start and completion times of task i handled by ASC k individually. A binary variable u_{ij} synchronizes the prime crane's storage and cooperative crane's retrieval operations for avoiding the interference of the twin ASCs at the handshake bay: if the prime crane (p) finishes task i before the cooperative crane (c) starting task j, then u_{ij} equals to 1; otherwise, u_{ij} is 0. If task i involves storage/retrieval operations at the handshake bay, y_i equals to 1; otherwise, y_i equals to 0.

The model assumptions are reflected in the two aspects. First, the ASC must be handed over through a buffer. ASC shall not cross the buffer during task processing. Second, ASC can avoid collision and achieve synchronization by maintaining a safe time interval.

$$\begin{cases} \min & z \\ \text{subject} & \text{to}, \end{cases} \tag{1}$$

$$\sum_j x_{ijk} \le 1, \forall i, k, \tag{2}$$

$$\sum_{i,j} x_{ijp} = |N| - 1, \tag{3}$$

$$\sum_{i,j} x_{ijc} = \sum_i y_i - 1, \tag{4}$$

$$e_{ip} \ge s_{ip} + T^m\left(d_{ip} - b_{ip}\right) + 2T^h, \forall i, \tag{5}$$

$$e_{ic} \ge s_{ic} + T^m\left(d_{ic} - b_{ic}\right) + 2T^h + M\left(y_i - 1\right), \forall i, \tag{6}$$

$$s_{jk} \ge e_{ik} + T^m\left(d_{ik} - b_{jk}\right) + M\left(x_{ijk} - 1\right), \forall i, j, k, \tag{7}$$

$$e_{ik} \ge s_{ik}, \forall i, k, \tag{8}$$

$$z \ge e_{ik}, \forall i, k, \tag{9}$$

$$s_{jc} \ge e_{ip} + T^s + M\left(u_{ij} - 1\right), \forall i, j, \tag{10}$$

$$e_{ip} \ge s_{jc} - M \cdot u_{ij} + M\left(1 - y_j\right) + 2T^h + T^s, \forall i, j, \tag{11}$$

$$\sum_j u_{ij} \le |N| \cdot y_i, \forall i, \tag{12}$$

$$u_{ii} = y_i, \forall i, \tag{13}$$

$$b_{ip} = O_i, \forall i \tag{14}$$

$$D_i \ge d_{ip} - M \cdot y_i, \forall i \tag{15}$$

$$d_{ip} \ge D_i - M \cdot y_i, \forall i, \tag{16}$$

$$w \ge d_{ip} + M\left(y_i - 1\right), \forall i, \tag{17}$$

$$d_{ip} \ge w + M\left(y_i - 1\right), \forall i, \tag{18}$$

$$w \ge b_{ic} + M\left(y_i - 1\right), \forall i, \tag{19}$$

$$b_{ic} \ge w + M\left(y_i - 1\right), \forall i, \tag{20}$$

$$D_i \ge d_{ic} + M\left(y_i - 1\right), \forall i, \tag{21}$$

$$d_{ic} \ge D_i + M\left(y_i - 1\right), \forall i, \tag{22}$$

$$M \cdot y_i \ge d_{ip} - w, \forall i, \tag{23}$$

$$2y_i \ge \sum_j x_{ijc} + \sum_j x_{jic}, \forall i, \tag{24}$$

$$s_{ik} \ge 0, \forall i, k, \tag{25}$$

$$\max_{\forall i} D_i \ge w, \tag{26}$$

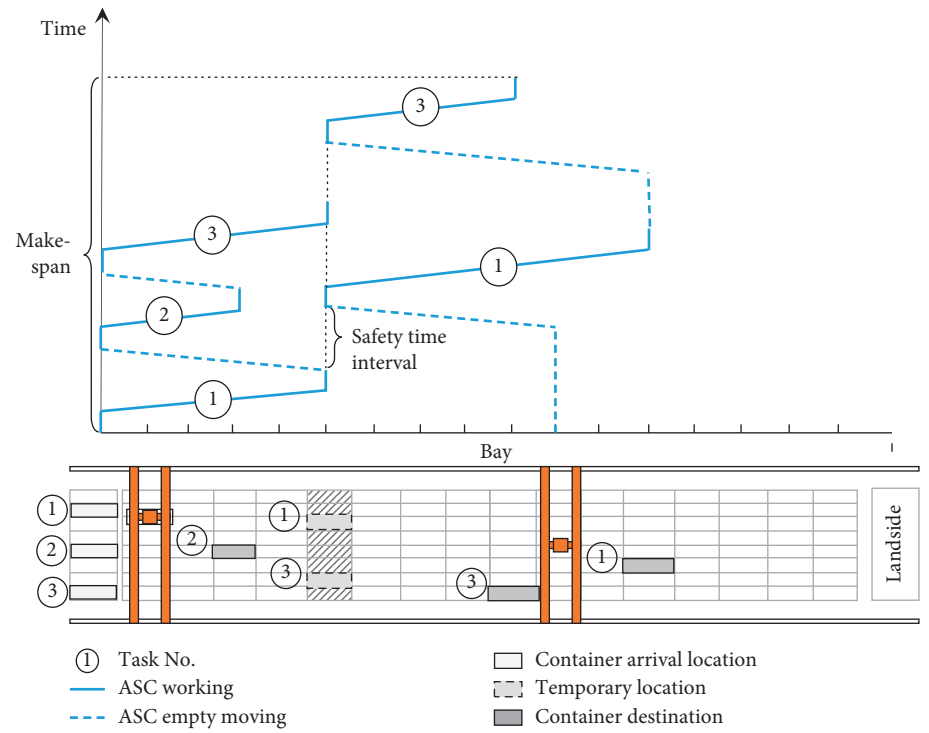

Figure 3: Time-space diagram of twin ASCs' operations with a handshake bay.

Table 2: Notations.

(1)	*Set*
$B = \{0, 1, \ldots, n_b\}$	A set of bays in yard block, where 0 is the I/O points of arrival containers
$N = \{1, \ldots, n\}$	A set of tasks, generally indexed by i, j
$K = \{p, c\}$	K consists of the prime and cooperative cranes (p, c), indexed by k
(2)	*Known data*
$O_i \in B$	Original bay of task i
$D_i \in B$	Destination bay of task i
T^s	The safe time interval between the handlings of ASCs
T^m	The time of crane moving one bay
T^h	The time of crane hoisting/dropping off one container
M	A large enough number, $M = \lvert N \rvert \cdot (T^m \cdot \lvert B \rvert + 2 \cdot T^h)$
(3)	*Decision variables*
$b_{ik} \in B$	The initial bay of task i handled by crane k
$d_{ik} \in B$	The target bay of task i handled by crane k
s_{ik}	The beginning time of crane k handling task i
e_{ik}	The finish time of crane k handling task i
$x_{ijk} \in \{0, 1\}$	1, if crane k handles task i directly before task j; 0, otherwise.
$y_i \in \{0, 1\}$	1, task i is transhipped by the handshake bay; 0, otherwise.
$u_{ij} \in \{0, 1\}$	1, if p finishes task i before c starting task j; 0, otherwise.
z	The makespan of the twin ASCs completing all the tasks
w	The handshake bay, $w \in B$

$$x_{ijk}, y_i, u_{ij} \in \{0, 1\}, \forall i, j, k. \quad (27)$$

The objective function (1) minimizes the makespan of the twin ASCs, subject to the following five groups of constraints:

(1) Sequencing constraints. Constraint (2) ensures that each task has at most one successor task. As constrained by (3) and (4), the cranes finish all the tasks. These constraints imply a flow balance, ensuring that the two cranes handle the tasks individually once and in sequence.

(2) Time constraints. We compute the start time and end time of prime crane (p) near the I/O point of arrival containers handling task i by the moving time and loading/unloading time, given in (5). The start time

and end time of the cooperative crane (c) handling the container at the handshake bay are computed by (6). The direct travel time of the crane from task i to task j is given in (7). As constrained by (8), the end time of the task must be later than the start time. In (9), the makespan is later than the end time of every task.

(3) Synchronization constraints. The safe time interval between the handlings of cranes at the handshake bay helps avoid crane collision. In (10), if the prime crane (p) handles task i before the cooperative crane handling task j, task j must start later than the end time of task i plus the safe time interval. Reversely, in (11), if we handle the task j before task i, the end time of task i must be later than the start time of task i plus safe time interval and the loading time of the two tasks. If j is staked at the handshake bay ($y_j = 1$) and the task i is not performed just before j ($u_{ij} = 0$), the inequality, $e_{ip} \geq s_{jc} + 2T^h + T^s$, should be met, which is studied in Figure 3. In (12), the cranes handle the containers at the handshake bay in a sequence. In (13), the prime crane (p) always handles the tasks before the cooperative crane (c).

(4) Handshake bay location constraints. In (14), the prime crane starts a task at the original location of the container. In (15) and (16), if the container is not stacked at the handshake bay, the prime crane finishes the task at its destination; otherwise, the prime crane finishes the task at the handshake bay, as constrained in (17) and (18). Constraints (19)–(22) ensure that the cooperative crane starts a temporary stacked task at the handshake bay and finishes it at its destination. In (23), if the prime crane stacks the container of a task at the handshake bay, its destination must be on the other side of the handshake bay. In (24), the cooperative crane transports the container at the handshake bay to its destination.

(5) Domain constraints. In (25)-(26), the start times of tasks are nonnegative, and the handshake bay must locate before the destinations of tasks.

Notably, the distances, $(d_{ik} - b_{ik})$ and $(d_{ip} - w)$, in (5)-(7) and (23), should be $(b_{ik} - d_{ik})$ and $(d_{ip} - w)$ for retrieval tasks to ensure that their values are nonnegative.

Proposition 1. *The problem of scheduling twin ASCs and optimizing the handshake bay is NP-hard.*

Proof. By relaxing the decision on the handshake bay's location, the twin ASC scheduling model determines the optimal task sequences of the cranes. Since the prime crane handles all the tasks one by one, we can obtain the prime crane's task sequence by solving a traveling salesman problem (TSP) where a traveler sequentially visits a set of vertices. As for the cooperative crane, the sequence consists of the tasks using the handshake bay for staking containers temporarily. We can also solve it with a TSP. Thus, the relaxed model is NP-hard because the TSP is such [50]. □

4. Solution Algorithms

The MILP is hard to be solved for a practical problem with more than ten tasks (see Section 5) because of the complexity of integrating handshake bay positioning and twin ASC scheduling. We developed an insert algorithm with a greedy strategy and priority-based GA [15, 16] with better computational efficiency.

4.1. Greedy Insert Algorithm. Inspired by scheduling experiences from the Yangshan ACT, the algorithm inserts tasks to the handling sequences of the ASCs to minimize the crane idle time. First, the algorithm selects one bay in the block as the handshake bay. Second, the algorithm alternately inserts a long-time task and a short-time task to the ASC close to the I/O point of arrival containers until all the tasks are assigned. Third, the algorithm adds the tasks using the handshake bay to tranship the containers to the cooperative ASC, considering the safety time interval. Fourth, the algorithm computes the makespan, as denoted by z_i. Fifth, the algorithm enumerates the available bays as handshake bay, computes the makespan, and then outputs the bay with the minimal makespan and the corresponding scheduling as the solution. Algorithm 1 presents the outline of the insert algorithm.

Proposition 2. *The computational complexity of Algorithm 1 is $O(m \cdot n^2)$, where n is the number of tasks and m denotes the number of bays in a block.*

Proof. We can decompose Algorithm 1 into two procedures that sequence the tasks by sorting according to their operation times. The worst sorting time is $n(n + 1)/2$, when all the tasks are temporarily stacked. So, the computational complexity of calculating the makespan once is $O(n^2)$. Through the algorithm, the makespan is calculated m times according to the number of candidate handshake bay. Thus, the whole computational complexity is $O(m \cdot n^2)$. □

4.2. The Genetic Algorithm. The greedy insert algorithm (Algorithm 1) can produce a feasible solution within a short computing time. However, the greedy search generates a single solution due to limited search space. We developed a GA using a real number encoding scheme and a priority-based decoding strategy. The GA is a typical representative intelligent algorithm and can incorporate many other successful heuristics developed for other algorithms, e.g., simulated annealing and various neighbor searches.

The GA derives from evolutionary computation and follows natural selection as the principle for searching for improved global optimization solutions in various applications [51], such as scheduling [52, 53] and routing [54]. The critical components include (1) encoding solution into individuals, (2) repetitively performing crossover and mutation operations to generate new individuals, and (3) reserving individuals with high-performance value by the fitness function. A GA organizes a population of individuals

to devise solutions and form generations. New individuals are produced based on the existing individuals and the next generation is composed, with some elite individuals selected from the current generation. Over massive generations and superior selection, the population evolves toward the best individuals. The crossover operator guarantees the local improvement during the generations, and mutation ensures that the GA searches widely in the feasible region. Crossover fraction and mutation rate are control parameters to exploit and explore optimal solutions.

4.2.1. The Algorithm Procedures. Algorithm 2 presents the procedures of GA. The algorithm randomly generates individuals to form the initial population and then improves the population iteratively. At each generation, firstly, the individuals are decoded into task sequences for the twin ASCs based on the decoding scheme; secondly, the fitness function computes the solutions' performances as the makespan of their tasks. Thirdly, the process transfers some elite individuals with the highest fitness value to the next generation. Fourthly, the process follows the crossover fraction to select some individuals as parents to generate child individuals. Fifthly, the process mutates the rest of the current individuals, and finally, the process decides on the excellent child individuals coupled with the elite individuals to form the next population. The algorithm terminates when it achieves the maximal generations or maintains the optimal solution for limited iterations.

4.2.2. Encoding Scheme. We use random keys [55] in a real number vector v to represent the priorities of tasks for permuting the task in the ASC handling sequences. The vector v contains $2 \cdot |N|$ elements, namely $v = [v_i]_{1\times(2|N|)}$ and $v_i \in [0, 1]$. The vector values represent the priorities of N tasks of the prime ASC and N tasks of the cooperative ASC. Figure 4 illustrates an individual containing five tasks.

A high value of v_i indicates a high priority of corresponding task i for insertion into the sequence. The algorithm adds the candidate task with the highest priority to the tail of the corresponding ASC's task sequence.

4.2.3. Decoding Scheme. The decoding scheme interprets a value vector into a solution and calculates the raw fitness value to provide evaluation value for further GA operators. Derived from the MILP, we designed Algorithm 3 to interpret a vector into feasible sequences for the two ASCs while considering the synchronization constraints. By the vector's values, the priorities of the ASC to the tasks are determined sequentially in the decoding scheme, which extends the solution method using constant crane priority [8].

To simulate the yard crane performances, Carlo and Martínez-Acevedo [8] and Gharehgozli et al. [14] developed schemes for sequencing the yard crane tasks using crane priorities. In this study, we implement this method in Algorithm 4 as a decoding scheme for comparison studies (see Section 5). Algorithms 3 and 4 use the priorities as decoding orders to construct a sequence of tasks gradually. However, Algorithm 3 uses the random keys' task priorities in a GA, while Algorithm 4 sequences the tasks according to the given yard cranes. In practice, the operators generally use crane-based priorities to schedule the crane operations, possibly because the method is intuitive. However, it cannot utilize the empty time. The task-based priorities provide a way to utilize the cranes by considering the empty time. However, it is not always true. So, we will study them through computational experiments.

4.2.4. Initialization. The GA generates a group of solutions as the initial population. As introduced in the decoding scheme (Section 4.2.3), we decode any real value vector v into the task sequences as a feasible solution. To start with a good initialization, we add the solution of the greedy insert algorithm (Algorithm 1) as one of the initial solutions. Therefore, the initial population is randomly generated in the initialization process, with one solution resulting from Algorithm 1.

4.2.5. Crossover and Mutation. As mentioned in 4.2.3, the decoding scheme allows various crossover and mutation operators designed for both discrete and continuous GAs. To get better performance of the decoding scheme, we test the operators proposed from literature and applied in practical solvers such as MATLAB. Finally, we select the two-point crossover and Gaussian mutation [56] to outperform other operators in tests among various designed operations. The two-point crossover selects two vector positions of one parent pair and exchanges the corresponding vector values between the two individuals to form a new individual. Gaussian mutation adds a random number taken from a Gaussian distribution with a mean 0 to each value of the parent vector and then scales them to form a new individual.

4.2.6. Fitness Evaluation. We use the fitness value to evaluate the performance of the solution represented by the individual. As formulated in Section 3, we prefer lower-makespan solutions; therefore, a low makespan z returned by the decoding scheme is of high performance. We convert the makespan z to the raw fitness value that is reciprocal to z. We use a rank-based method for scaling the fitness values. The method ranks the individuals based on the raw fitness in descending order and sets the fitness value of individual ranked at position n_d equal to $\sqrt{n_d}$ [57].

4.2.7. Elite and Selection. The selection method chooses high fitness values with high possibilities to form parents of the next generation. We use a roulette selection scheme to select the candidate parents by making up a roulette wheel. The section size of the wheel corresponding to an individual is in proportion to the individual's fitness value and then selecting several individuals with a probability as the parents. The probability (α) determines the fraction of individuals operated by crossover.

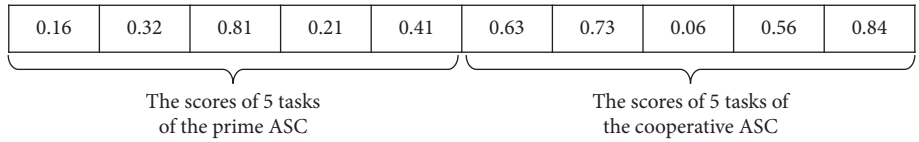

Figure 4: An individual representation of five tasks for the twin ASCs.

5. Experiments

We conduct extensive numerical experiments to evaluate the formulated model's effectiveness, the greedy insert algorithm (Algorithm 1), the crane priority-based GA, and the task priority-based GA (Algorithm 2). The model and algorithms were coded and implemented in the MATLAB 2016a platform on a computer with Intel (R) Core i7-5500U CPU Dual 2.4 GHz, 8 GB memory, and Windows 10 operating system. We solve the MILP with Gurobi 725 (http://www.gurobi.com) using MATLAB.

5.1. Data and Experimental Settings. We use the following data to construct the instances and settings in the experiments. The data are set based on the investigations to the Shanghai Yangshan ACT. The authors provided mathematical modeling courses and assistants to the ACT designers, managers, and operators. The block consists of 28 bays and two I/O points at the ends. The travel speed of ASC is set to 2.5 m/s. That is, an ASC takes 6 s to travel a bay. The hoist/drop speed of ASC is set to 2 m/s. In practice, ASC needs a calibration time of 25–30 s for orienting the container location to be operated. Therefore, we set the hoist/drop time to 30 s. The task number n is set to 5, 10, 20, 50, and 100. For each number of tasks, we generate four scenarios of task distributions (denoted as "s," "c," "l," and "u") so that tasks are distributed from the I/O point of arrival container to 1/3 length of the block, from 1/3 length of the block to 2/3 length of the block, from 2/3 length of the block to the end of the block, and distributed uniformly in the block. The safe time interval between two crane handlings is set to 9 s, indicating a minimal half-bay distance between twin ASCs. In summary, we represent the model parameters described above by a value vector, $(n_b, T^m, T^h, T^s) = (28, 6, 30, 9)$.

The parameters of GA include maximal generation G, stall generation φ, population size ρ, crossover fraction α, and mutation ratio β. In the parameter tuning experiment, we adjust one parameter and fix other parameters simultaneously. Section 5.2.1 presents more details of tuning the parameters. The default values of the parameters are $(G, \varphi, \rho, \alpha, \beta) = (1000, 450, 20, 0.6, 0.55)$.

In Table 3, we devise three experiments with purposes and parameter settings.

5.2. Experimental Result

5.2.1. Parameter Tuning. We demonstrate the parameter tuning experiments using instances with 50 tasks uniformly distributed in a container yard block. Figure 5 depicts the results for each tuned parameter. The objective value decreases dramatically at the beginning of execution, and the decreased speed slows down after 100 generations. The algorithm result converges to an optimal value and holds stable after 800 generations. The algorithm terminates after the stall generation, and the best value is 450. In Figure 5(c), the lowest objective value incurs at population size 20; after that point, computation time increases, while the objective value is unchanged. The crossover fraction (α) and mutation ratio (β) jointly determine the GA's progress. Figure 5(d) produces a contour plot for these two parameters and their corresponding makespan. The objective value drops to the lowest when the crossover fraction reaches 0.6 and the mutation ratio to 0.55. We mark two regions of combinations that lead to good performances by grey colors in Figure 5(d).

The parameters are sensitive to the number of tasks and even their distributions. Here, we mainly demonstrate the processes and typical results for tuning the parameters. In application scenarios, after testing the distribution of the tasks and the size of a batch of tasks in scheduling, we can use it to determine the parameters before regular production.

5.2.2. Algorithm Performances. We evaluate the proposed model's performance and algorithms in computation time (CT) and optimal objective value (Obj). It is found in the experiments that a feasible solution of MILP solved by the on-the-shelf solver takes several hours. So, it is not applicable to use the MILP solver in practice. In the following experiments, we set the time limit of two hours for solving the MILP. We conduct 500 runs of the priority-based GA and take the final objective values as ideal objective values, which are "ideal" bounds when the MILP solver cannot solve the model within two hours.

As presented in Table 4, the MILP solver can solve optimally for small-scale instances with five tasks. The performance of the MILP solver decreases for the medium-scale instances with ten to twenty tasks since the running time reaches the limit. For the instances of more than 50 tasks, the MILP solver cannot find feasible solutions within the time limit (remarked as the notation "–" in the table). The greedy insert algorithm (Algorithm 1) can find a solution in less than 0.2 seconds, and its gaps between the ideal solutions narrow down when the problem scales grow. The crane priority-based GA (Algorithm 1 using the decoding scheme in Algorithm 4) is robust in computation time. The task priority-based GA improves the greedy algorithm and outperforms other contenders in objective value. For small-scale instances, the task priority-based GA found the optimal results equal to those from the MILP solver. In the medium- and large-scale instances, the task priority-based GA can reduce the gaps between the ideal solutions by 5%, even to

Inputs $O_i, D_i, T^s, T^m, T^h, N, K$
Outputs $b_{ik}, d_{ik}, s_{ik}, e_{ik}, z, w$
Steps
Step 1 Initialize $z = |N| \cdot (T^m \cdot |B| + 2 \cdot T^h)$
Step 2 Initialize a set of candidate handshake bays, denoted by $\Omega = \left\{\min_i(O_i), \min_i(O_i)+1, \ldots, \max_i(D_i)\right\}$. Use Ω_j to index the jth element in Ω
Step 3 For each Ω_j
  Step 3.1 Initialize two sets N^p and N^c, and two sequence vectors V_p and V_c
  Step 3.2 Let $l = \Omega_j$
  Step 3.3 *For* each $i \in N$
    Step 3.3.1 Set $b^*_{ip} = O_i$
    Step 3.3.2 If $D_i > l$, then set $d^*_{ip} = l, b^*_{ic} = l, d^*_{ic} = D_i$, add i to N^p and to N^c
    Step 3.3.3 else, set $d^*_{ip} = D_i, b^*_{ic} = -1, d^*_{ic} = -1$, and add i to N^p
  Step 3.3.4 *End for*
  Step 3.4 Initialize a set of tasks as $\Theta = N^p \bigcup N^c$
  Step 3.5 While Θ is not empty
    Step 3.5.1 Select the task with the longest distance $|D_i - O_i|, i \in \Theta$, and append i to V_p
    Step 3.5.2 If $b^*_{ic} > 0$
      Calculate (s^*_{ic}, e^*_{ic}) by the safe time interval T^s, and append i to V_c
    End If
    Step 3.5.3 Remove i from Θ
    Step 3.5.4 Select the task i with the shortest distance $|D_i - O_i|, i \in \Theta$, and append i to V_p
    Step 3.5.5 If $b^*_{ic} > 0$
      Calculate (s^*_{ic}, e^*_{ic}) by T^s, and append i to V_c
    End If
    Step 3.5.6 Remove i from Θ
    Step 3.5.7 End while
  Step 3.6 Calculate $z^* = \max_{i \in N, k}(e^*_{ik})$
  Step 3.7 If $z^* < z$
    $b_{ik} \leftarrow b^*_{ik}, d_{ik} \leftarrow d^*_{ik}, s_{ik} \leftarrow s^*_{ik}, e_{ik} \leftarrow e^*_{ik}, z \leftarrow z^*, w \leftarrow l, \forall i, k$
    End If
  Step 3.8 End For
Step 4 Output $b_{ik}, d_{ik}, s_{ik}, e_{ik}, z, w, \forall i, k$

Algorithm 1: Greedy insert algorithm.

0.06%. The computation times show that the efficiency of the task priority-based GA is related to the number of tasks because the synchronization of the two crane scheduling tasks takes much time to determine the feasible task sequences avoiding crane interference.

Figure 6 presents the results of solving the model and algorithms for seven uniformly distributed tasks. All methods generated an optimal handshake location at the bay of No. 11. The crane priority-based GA found a near-optimal solution with an extra six seconds compared with optimal makespan, and the greedy insert algorithm (Algorithm 1) is uncompetitive. The MILP and task priority-based GA optimized the identical objective value as the optimal result, while the crane handlings' sequences obviously differ.

5.2.3. Parameter Sensitivity. We examine the effect of the synchronization strength (length of the safe time interval, T^s) and handshake bay positions on the solutions in the following.

Figure 7 displays the impact of safe time intervals on the incremental makespan of the tasks in the four scenarios. With the makespan of safe time of $T^s = 6$ seconds as the benchmark, the makespan rises to 13.5% after T^s increasing to 30 seconds. We can observe a notable makespan increment in the scenario of tasks distributed in the block seaside, and the curve is steep during the safe time prolonging.

Figure 8 displays the incremental makespan of various handshake bays compared with the optimal handshake bay, which locates at the bay of No. 5 in this instance. The makespan decreases when the handshake bay moves close toward the optimal bay. After the handshake bay reaches optimal, we can observe a dramatic increment in makespan, and the prime crane carries most tasks to their destination, while the other crane cooperates with fewer tasks. Compared with the incremental makespan, the optimal buffer location can save 35% working time than the handshake bay's worst assignment.

The following practical insights are evaluated based on the results shown in Figures 7 and 8. Firstly, the impact of synchronization strength on the makespan relates to the distribution of tasks. An extensive time delay is needed to synchronize the two cranes when the cranes have to work in a compact space of dense tasks. Secondly, a proper handshake bay helps to relieve the time of delay caused by the synchronized operations. The temporal space can reassign each device's tasks, therefore balancing the workload

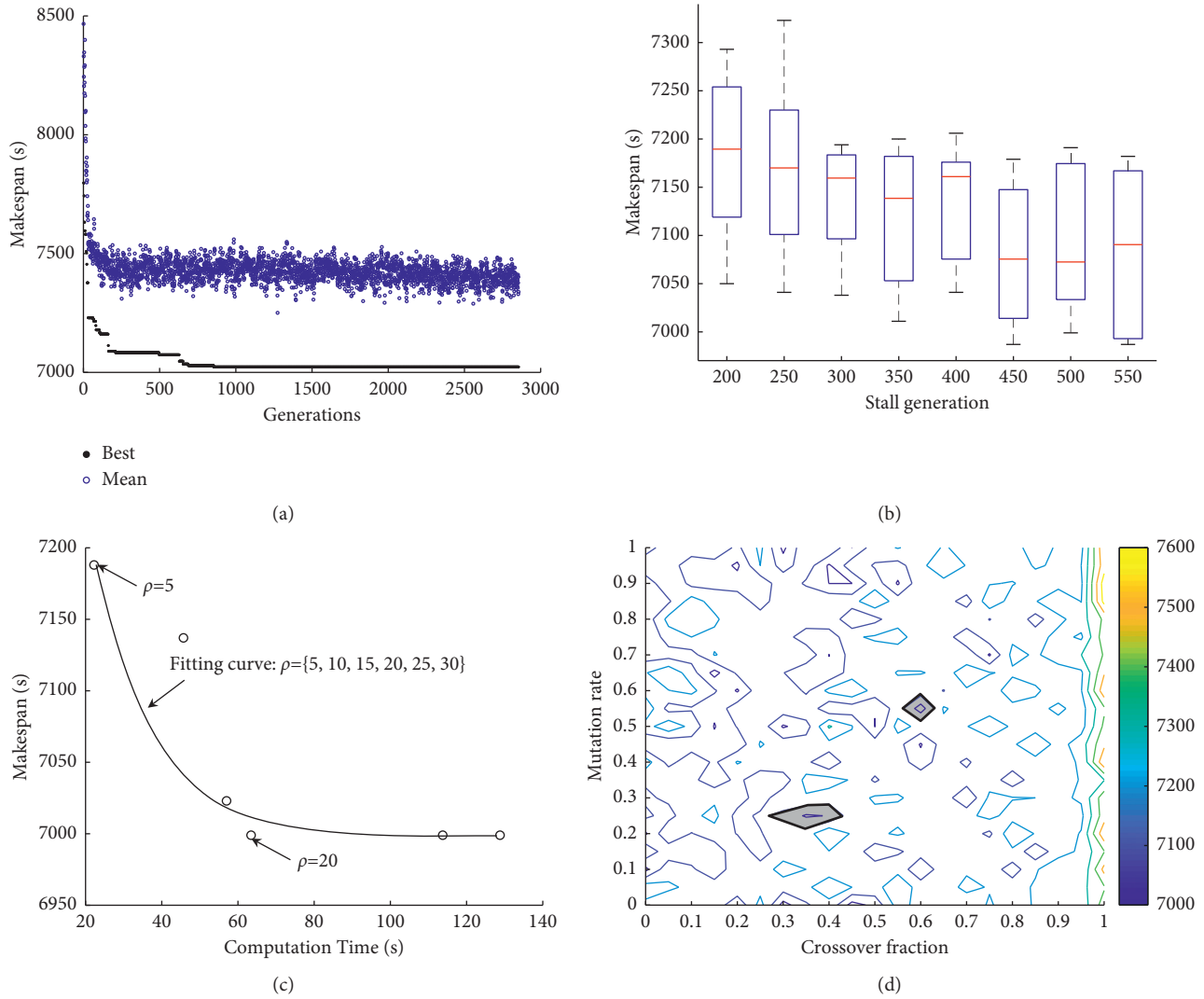

Figure 5: Impacts of the GA parameters on solutions. (a) Objective value of GA. (b) Tuning of stall generation. (c) The effect of population sizes (ρ) on makespan versus computation time. (d) The outperformed regions of combinations of crossover fraction and mutation ratio.

Steps
Step 1 $g \leftarrow 0$
Step 2 Generate the initial population $P(g)$
Step 3 Evaluate $P(g)$ and select the elite solutions $E(g)$
Step 4 While (termination criteria are not satisfied) do
Step 4.1 Crossover $P(g)$ to generate $C^X(g)$
Step 4.2 Mutate $P(g)$ to generate $C^M(g)$
Step 4.3 Merge $C^X(g)$ and $C^M(g)$ as $C(g)$
Step 4.4 Improve $C(g)$ by crossover and mutation operator
Step 4.5 Evaluate $C(g)$ based on a decoding scheme
Step 4.6 Move best solutions from $P(g)$ and $C(g)$ to $E(g)$
Step 4.7 Select $P(g+1)$ from $P(g)$ and $C(g)$ by roulette wheel selection
Step 4.8 $g \leftarrow g+1$
Step 4.9 End While

Algorithm 2: Genetic algorithm.

Inputs $O_i, D_i, T^s, T^m, T^h, c, N, K$
Outputs $b_{ik}, d_{ik}, s_{ik}, e_{ik}, z, w$
Steps
Step 1 Initialize $z = |N| \cdot (T^m \cdot |B| + 2 \cdot T^h)$ and a set $N^W = \varnothing$ containing uncompleted tasks and the corresponding cranes
Step 2 Select the bays between $\min_i(O_i)$ and $\max_i(D_i)$ to form a candidate bay set Ω. Use Ω_j to index the jth element in Ω
Step 3 For each Ω_j
  Step 3.1 Let $l = \Omega_j$, initialize sequence $V_k, k \in K$, initialize an occupied time set $\Gamma = \varnothing$. Use Γ_j to index the jth element in Γ.
  Step 3.2 Insert all the tasks of the prime crane into N^W, i.e., $N^W = \{(1, p), \ldots, (n, p)\}$.
  Step 3.3 While N^W is not empty
    Step 3.3.1 Select (i, k) from N^W with the highest vector value in v, and insert i to the sequence tail of V_k
    Step 3.3.2 Calculate $b^*_{ik}, d^*_{ik}, s^*_{ik}, e^*_{ik}$
    Step 3.3.3 If k is the prime crane
      Step 3.3.3.1 If e^*_{ik} violates the safe time interval, namely $e^*_{ik} \in \Gamma_j, \exists j$
        Delay e^*_{ik} to a conflict-free time value, and add $[e^*_{ik} - T^h - T^s, e^*_{ik} + T^s]$ to Γ
      End If
      Step 3.3.3.2 Add (i, c) to N^W if task i is stacked at l for operated by the other crane.
    Step 3.3.4 else
      Step 3.3.4.1 If s^*_{ic} violates the safe time interval, namely, $s^*_{ic} \in \Gamma_j, \exists j$
        delay s^*_{ic} to a conflict-free time value, and add $[s^*_{ic} - T^s, s^*_{ic} + T^h + T^s]$ to Γ
      End If
    Step 3.3.5 End If
  Step 3.4 End While
  Step 3.5 Set $z^* = \max_{i \in N, k}(e^*_{ik})$
  Step 3.6 If $z^* < z$
    $b_{ik} \leftarrow b^*_{ik}, d_{ik} \leftarrow d^*_{ik}, s_{ik} \leftarrow s^*_{ik}, e_{ik} \leftarrow e^*_{ik}, z \leftarrow z^*, w \leftarrow l, \forall i, k$
  End If
Step 4 End For
Step 5 Output $b_{ik}, d_{ik}, s_{ik}, e_{ik}, z, w, \forall i, k$

Algorithm 3: Decoding scheme based on task priorities.

Inputs $O_i, D_i, T^s, T^m, T^h, c, K, N = \{1, \ldots, n\}$
Outputs $b_{ik}, d_{ik}, s_{ik}, e_{ik}, z, w$
Steps
Step 1 Initialize $z = |N| \cdot (T^m \cdot |B| + 2 \cdot T^h)$
Step 2 Select the bays between $\min_i(O_i)$ and $\max_i(D_i)$ to form a candidate bay set Ω. Use Ω_j to index the jth element in Ω
Step 3 For each Ω_j
  Step 3.1 Let $l = \Omega_j$, initialize sequence V_p and V_c, initialize an occupied time set $\Gamma = \varnothing$. Use Γ_j to index the jth element in Γ.
  Initialize two sets N^p and N^c containing the task for prime crane and split task for cooperative
  Step 3.2 Sort $v_i, i = 1, \ldots, n$ in descending order, and insert i into V_p sequentially.
  Step 3.3 Calculate $b^*_{ip}, d^*_{ip}, s^*_{ip}, e^*_{ip}$ according to V_p, T^m, T^h.
  Step 3.4 Insert the occupied time of each split task at handshake bay into Γ
  Step 3.5 Sort $v_{i+n}, i = 1, \ldots, n$ in descending order, and insert i into V_c sequentially.
  Step 3.6 Calculate $b^*_{ip}, d^*_{ip}, s^*_{ip}, e^*_{ip}$ according to V_p, T^m, T^h, Γ
  Step 3.7 Set $z^* = \max_{i \in N, k}(e^*_{ik})$
  Step 3.8 If $z^* < z$
    $(b_{ik} \leftarrow b^*_{ik}, d_{ik} \leftarrow d^*_{ik}, s_{ik} \leftarrow s^*_{ik}, e_{ik} \leftarrow e^*_{ik}, z \leftarrow z^*, w \leftarrow l), \forall i, k$
  End If
  Step 3.9 End For
Step 4 Output $(b_{ik}, d_{ik}, s_{ik}, e_{ik}, z, w), \forall i, k$

Algorithm 4: Decoding scheme based on crane priorities.

between devices to reach an optimal result in a whole scheduling perspective. The location of the temporal space affects the assignment of tasks to cranes and the moving distances of the crane handling tasks. Therefore, the location is significant in balancing the cranes' workloads and optimizing the tasks' makespan. Finally, the handshake bay positioning and ASC scheduling must be optimized as integrated decisions because they are mutually affected and

Table 3: Experimental settings.

No.	Purposes	Parameter settings
1	Tune the GA to determine the robust parameter value	(1) Set max generations (G) to 5000; (2) Set stall generations (φ) to 200, 300, 400, 500, and 600, and run the GA for 20 times individually; (3) Set the crossover fraction (α) to 0.05, 0.10, . . ., 0.95, and set the mutation rate (β) to 0.05, 0.10, . . ., 0.95. Use these values for cross experiments and then run the GA for each combination of crossover fraction and mutation rate five times.
2	Compare the optimality and computation time of the MILP and proposed algorithms	(1) Use the settings in No. 1; (2) Set the run time limitation of the MILP solver to 900 s; (3) Use the MILP solver, the greedy insert algorithm, and GA (Algorithms 1 and 2) to solve the problem.
3	Examine the impacts of synchronization constraints and handshake location on the makespan and ASC operations	(1) Apply GA with its default settings; (2) Set the safe time interval (T^s) to 6, 12, 18, 24, and 30 s; (3) Set a handshake bay (w) to an available bay.

Table 4: Objective value and computation time of solving the model and algorithms.

Instance	MILP		Greedy insert			Crane priority-based GA			Task priority-based GA			Ideal
	Obj.	CT/s	Obj.	CT/s	Gap[b] (%)	Obj.	CT/s	Gap[b] (%)	Obj.	CT/s	Gap[b] (%)	Obj.[c]
5s	495	5.05	555	0.12	12.12	504	6.76	1.82	**495**	3.87	0.00	**495**
5c	753	1.67	765	0.02	1.59	759	7.26	0.80	**753**	3.88	0.00	**753**
5l	1065	4.85	1065	0.01	0.00	1071	8.02	0.56	**1065**	3.78	0.00	**1065**
5u	630	1.65	747	0.01	18.57	633	7.63	0.48	**630**	4.83	0.00	**630**
10s	999	7200.00[a]	1035	0.01	13.18	1020	5.58	11.54	933	6.35	2.02	**915**
10c	1569	7200.00[a]	1611	0.01	9.91	1557	9.46	6.22	1479	6.08	0.90	**1466**
10l	2145	7200.00[a]	2019	0.01	4.00	2067	9.15	6.47	1989	6.09	2.45	**1941**
10u	1419	7200.00[a]	1527	0.00	15.12	1395	12.83	5.17	1380	6.50	4.04	**1326**
20s	2376	7200.00[a]	1983	0.01	11.08	1968	5.80	10.24	1824	15.85	2.18	**1785**
20c	4305	7200.00[a]	3141	0.00	8.32	3306	9.60	14.01	2919	20.01	0.67	**2900**
20l	6159	7200.00[a]	4317	0.01	7.34	4323	16.44	7.49	4113	13.64	2.27	**4022**
20u	4398	7200.00[a]	3207	0.01	7.42	3228	8.23	8.12	3045	17.33	1.99	**2986**
50s	—	7200.00[a]	4731	0.01	6.61	5322	7.90	19.93	4632	42.50	4.38	**4438**
50c	—	7200.00[a]	7845	0.01	6.46	9936	22.33	34.84	7419	35.34	0.68	**7369**
50l	—	7200.00[a]	10689	0.01	9.10	12429	22.52	26.85	10269	69.93	4.81	**9798**
50u	—	7200.00[a]	7335	0.01	2.97	8688	10.12	21.97	7152	86.12	0.40	**7123**
100s	—	7200.00[a]	9195	0.01	0.67	11052	12.40	21.01	9201	137.12	0.74	**9133**
100c	—	7200.00[a]	15681	0.01	5.44	20574	31.98	38.34	15111	164.95	1.61	**14872**
100l	—	7200.00[a]	21285	0.02	3.98	26259	45.37	28.28	20673	161.15	0.99	**20471**
100u	—	7200.00[a]	14733	0.01	2.91	20040	17.12	39.97	14325	219.40	0.06	**14317**

Note. [a]Terminated by Gurobi after 2 hours, "—" indicates no feasible solution found. [b] Gap = ((Method Obj.) − (Ideal Obj.))/(Ideal Obj.) × 100%. [c]Ideal Obj. is obtained by running the task priority-based GA for 500 times.

jointly affect the solution optimality. The position of the handshake bay affects the reassignment of tasks to ASCs, the moving distances, and the working time of each task by the ASCs.

5.3. Discussion

(1) Based on the algorithms' comparisons (Table 4), the greedy insert algorithm succeeds in computing time and space complexities. When many tasks are involved, the computing performance is still competitive compared with the MILP solver and GAs. The MILP solvers (e.g., Gurobi and Cplex) are suitable for solving small-scale instances, and generally, optimal solutions are achievable within an acceptable time. The GA takes more computation time than that of the greedy insert algorithm. Nevertheless, the GA can stop using the number of maximal generations as a termination criterion. So, the optimality and computing time of using GA can be balanced, while the solution method based on the MILP solver may not find a feasible solution within acceptable times. Comparatively, the greedy insert algorithm can only find a feasible solution by experience-based rules. The MILP solver can find optimal solutions optimistically, while it cannot find a feasible solution always due to computing complexities (in time or space requirements). The GA can find improved solutions, ideally when it runs for enough generations. In summary, three solution methods are capable of different scales of problem instances. As revealed by the experiments, the task

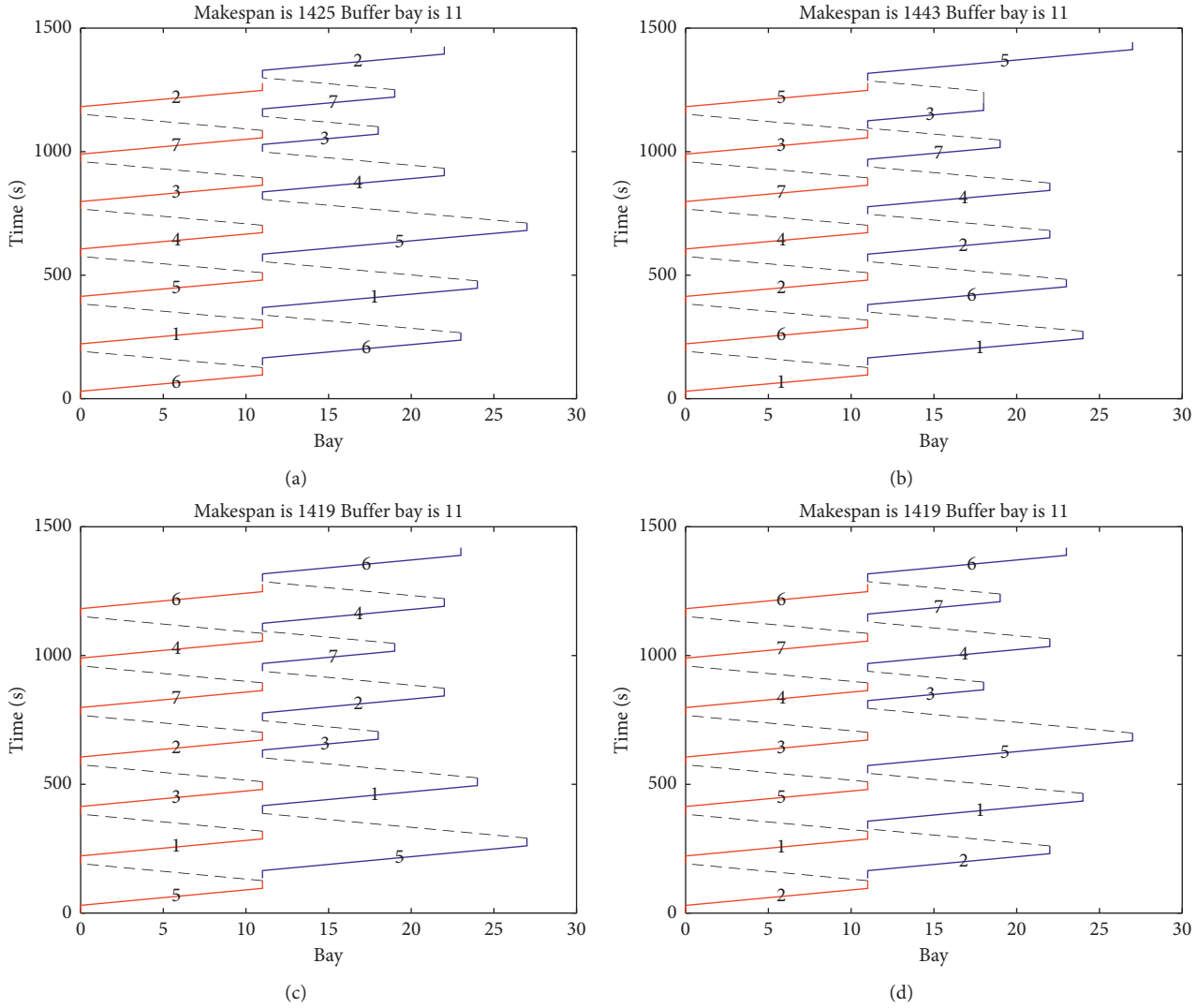

Figure 6: Time-space diagrams of seven tasks scheduled by the proposed methods. (a) Crane priority-based GA. (b) Greedy insert. (c) MILP. (d) Task priority-based GA.

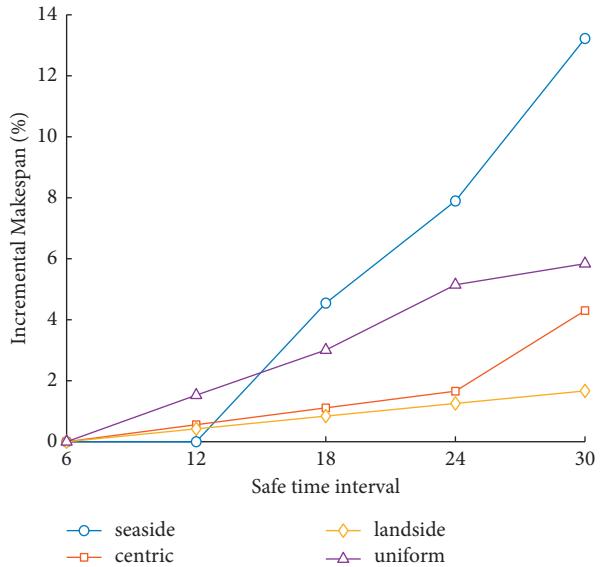

Figure 7: Impact of safe time interval on makespan.

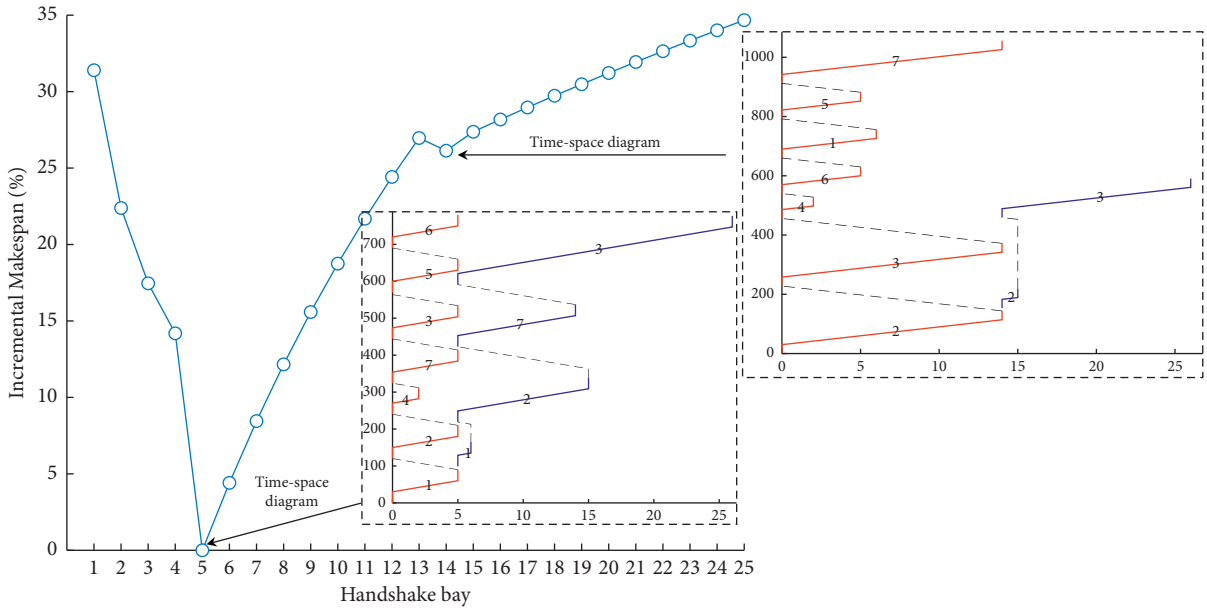

Figure 8: Impact of handshake bay on the solutions.

priority-based GA is competitive for the medium- and large-scale instances in practical and operational environments. The terminal can choose the corresponding solution method according to different problem instances to improve the operation efficiency of an automatic container terminal.

(2) Real-world logistic optimization problems usually involve multi-vehicle-type multistage operations. All the vehicles and strategies function within the constrained working places. Synchronization constraints are general forms representing the complex relations among facilities, vehicles, stages, and operations in the dimensions of time, space, and tasks. The synchronization relation or constraint becomes a conceptualization, classification, and modeling tool. In ACTs, automated quay cranes, ACTs, and AGVs are different types of vehicles in different operations stages, while they interact and even interconnect with each other within the limited space of the terminals; meanwhile, we must coordinate them with the operation facilities (e.g., berths and yard blocks). Although this study mainly conceptualized the time-space and critical resource synchronizations considering the impacts of the handshake bay, various synchronization relations constrain the operations in ACTs.

6. Conclusion

This study investigates an operational problem of positioning a handshake bay and scheduling twin ASCs in a single container yard block at ACTs and, consequently, discusses the management insights and operational improvements. A handshake bay is temporary storage to coordinate the twin ASCs that sequentially stack and retrieve containers in the block to increase the utilization degree of cranes. We identified the time-space and critical resource synchronizations in using a handshake bay to avoid crane collisions. Then, we formulate a MILP to integrate the decisions of the handshake bay and ASC scheduling, where we represent the synchronization constraints by a safe time interval to avoid crane collision. A handshake task is operated by the two cranes sequentially: the cooperative crane can retrieve a container from the handshake bay after the prime crane drops the container to the handshake bay. The position of the handshake bay determines the travel distance of ASCs. The MILP solver is valid and efficient for the small-scale instances; however, it is hard for the solver to optimize the medium- and large-scale instances. Therefore, the greedy algorithm is proposed to search for a solution efficiently, and the task priority-based GA can sequence the handlings of twin ASCs with a dynamic priority for the crane handlings to search for optimal solutions globally. The computational experiments validated the efficiency of the algorithms for large-scale instances. For the large-scale instances, the proposed GA can find the near-optimal results within a 5% gap compared with the ideal solution and outperforms the MILP solver, the greedy insert, and the algorithm using constant crane priorities. In addition to the methodological studies, we conduct experiments to analyze the impacts of stacking operations and synchronization constraints on the makespan, such as the safe time interval and handshake bay position. The optimal handshake bay can save 35% working time compared with choosing the worst handshake bay.

As for future research directions, the handshake bay and even the ASC operations are not independent of various operations in ACTs. First, we have formulated the handshake bay positioning and scheduling model for the containers unloaded from the vessels, while the ASCs service for the container-handling tasks from and to vessels, and from and to hinterlands. To consider the combinations of these

container-handling tasks, we must revise the proposed model and develop new algorithms. Second, as elucidated in Section 3, we assume that all these parameters are constants while uncertain and even dynamically changing. Because of the uncertain environment of ACTs, we will investigate new stochastics or robust optimization models in the future. Third, we use "ideal" objective values as the lower bounds for algorithmic comparisons. We will try to formulate the mathematical tight lower bounds in the future. The challenges of operation optimization in container terminals generally relate to integrating various vehicles and resources, which produces various synchronization constraints. We will combine the model algorithms developed in this study with our previous and pioneering studies in the literature.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was partially supported by the National Nature Science of China (71871136).

References

[1] M. F. Gorman, J. P. Clarke, A. H. Gharehgozli, M. Hewitt, R. de Koster, and D. Roy, "State of the practice: a review of the application of OR/MS in freight transportation," *Interfaces*, vol. 44, no. 6, pp. 535–554, 2014.

[2] Statista, "International Seaborne Trade Carried by Container Ships from 1980 to 2017," 2017, https://www.statista.com/statistics/253987/international-seaborne-trade-carried-by-containers/.

[3] A. Azab, A. Karam, and A. Eltawil, "A simulation-based optimization approach for external trucks appointment scheduling in container terminals," *International Journal of Modelling and Simulation*, vol. 40, no. 5, pp. 321–338, 2019.

[4] J. Pasha, M. A. Dulebenets, A. M. Fathollahi-Fard et al., "An integrated optimization method for tactical-level planning in liner shipping with heterogeneous ship fleet and environmental considerations," *Advanced Engineering Informatics*, vol. 48, 2021.

[5] O. Theophilus, M. A. Dulebenets, J. Pasha, Y. Y. Lau, A. M. Fathollahi-Fard, and A. Mazaheri, "Truck scheduling optimization at a cold-chain cross-docking terminal with product perishability considerations," *Computers & Industrial Engineering*, vol. 156, 2021.

[6] K. Si, "Shanghai Retains crown of World's Busiest Container Port," 2019, http://www.seatrade-maritime.com/news/asia/shanghai-keeps-crown-of-world-s-busiest-container-port.html.

[7] A. Karam, A. Eltawil, and K. H. Reinau, "Energy-efficient and integrated allocation of berths, quay cranes, and internal trucks in container terminals," *Sustainability*, vol. 12, 2020.

[8] H. J. Carlo and F. L. Martínez-Acevedo, "Priority rules for twin automated stacking cranes that collaborate," *Computers & Industrial Engineering*, vol. 89, no. 1, pp. 23–33, 2015.

[9] H. J. Carlo and I. F. A. Vis, "New initiatives in stacking cranes configurations," *Port Technology International*, vol. 44, no. 1, pp. 32–36, 2009.

[10] U. Dorndorf and F. Schneider, "Scheduling automated triple cross-over stacking cranes in a container yard," *Spectrum*, vol. 32, no. 3, pp. 617–632, 2010.

[11] I. F. A. Vis and H. J. Carlo, "Sequencing two cooperating automated stacking cranes in a container terminal," *Transportation Science*, vol. 44, no. 2, pp. 169–182, 2010.

[12] Straitstimes, "World's Largest Automated Container Terminal Opens in Shanghai," 2017, https://www.straitstimes.com/asia/east-asia/worlds-largest-automated-container-terminal-opens-in-shanghai.

[13] A. H. Gharehgozli, G. Laporte, Y. Yu, and R. De Koster, "Scheduling twin yard cranes in a container block," *Transportation Science*, vol. 49, no. 3, pp. 686–705, 2015.

[14] A. H. Gharehgozli, F. G. Vernooij, and N. Zaerpour, "A simulation study of the performance of twin automated stacking cranes at a seaport container terminal," *European Journal of Operational Research*, vol. 261, no. 1, pp. 108–128, 2017.

[15] C. Chitra and P. Subbaraj, "A nondominated sorting genetic algorithm solution for shortest path routing problem in computer networks," *Expert Systems with Applications*, vol. 39, no. 1, pp. 1518–1525, 2012.

[16] M. Gen, F. Altiparmak, and L. Lin, "A genetic algorithm for two-stage transportation problem using priority-based encoding," *Spectrum*, vol. 28, no. 3, pp. 337–354, 2006.

[17] J. Lehnfeld and S. Knust, "Loading, unloading and premarshalling of stacks in storage areas: survey and classification," *European Journal of Operational Research*, vol. 239, no. 2, pp. 297–312, 2014.

[18] W. C. Ng and K. L. Mak, "Yard crane scheduling in port container terminals," *Applied Mathematical Modelling*, vol. 29, no. 3, pp. 263–276, 2005.

[19] A. H. Gharehgozli, Y. Yu, R. de Koster, and J. T. Udding, "An exact method for scheduling a yard crane," *European Journal of Operational Research*, vol. 235, no. 2, pp. 431–447, 2014.

[20] O. Sharif, N. Huynh, M. Chowdhury, and J. M. Vidal, "An agent-based solution framework for inter-block yard crane scheduling problems," *International Journal of Transportation Science and Technology*, vol. 1, no. 2, pp. 109–130, 2012.

[21] V. Galle, C. Barnhart, and P. Jaillet, "Yard Crane Scheduling for container storage, retrieval, and relocation," *European Journal of Operational Research*, vol. 271, no. 1, pp. 288–316, 2018.

[22] W. Li, Y. Wu, M. E. H. Petering, M. Goh, and R. d. Souza, "Discrete time model and algorithms for container yard crane scheduling," *European Journal of Operational Research*, vol. 198, no. 1, pp. 165–172, 2009.

[23] W. C. Ng, "Crane scheduling in container yards with inter-crane interference," *European Journal of Operational Research*, vol. 164, no. 1, pp. 64–78, 2005.

[24] F. Fotuhi, N. Huynh, J. M. Vidal, and Y. Xie, "Modeling yard crane operators as reinforcement learning agents," *Research in Transportation Economics*, vol. 42, no. 1, pp. 3–12, 2013.

[25] M. Sha, T. Zhang, Y. Lan et al., "Scheduling optimization of yard cranes with minimal energy consumption at container terminals," *Computers & Industrial Engineering*, vol. 113, no. 1, pp. 704–713, 2017.

[26] J. He, Y. Huang, and W. Yan, "Yard crane scheduling in a container terminal for the trade-off between efficiency and energy consumption," *Advanced Engineering Informatics*, vol. 29, no. 1, pp. 59–75, 2015.

[27] A. Gharehgozli and N. Zaerpour, "Stacking outbound barge containers in an automated deep-sea terminal," *European Journal of Operational Research*, vol. 267, no. 3, pp. 977–995, 2018.

[28] N. Boysen, D. Briskorn, and F. Meisel, "A generalized classification scheme for crane scheduling with interference," *European Journal of Operational Research*, vol. 258, no. 1, pp. 343–357, 2017.

[29] D. Briskorn and P. Angeloudis, "Scheduling co-operating stacking cranes with predetermined container sequences," *Discrete Applied Mathematics*, vol. 201, no. 1, pp. 70–85, 2016.

[30] D. Briskorn, S. Emde, and N. Boysen, "Cooperative twin-crane scheduling," *Discrete Applied Mathematics*, vol. 211, no. 1, pp. 40–57, 2016.

[31] S. Ozcan and D. T. Eliiyi, "A reward-based algorithm for the stacking of outbound containers," *Transportation Research Procedia*, vol. 22, no. 1, pp. 213–221, 2017.

[32] D. Kress, J. Dornseifer, and F. Jaehn, "An exact solution approach for scheduling cooperative gantry cranes," *European Journal of Operational Research*, vol. 273, no. 1, pp. 82–101, 2019.

[33] F. Jaehn and D. Kress, "Scheduling cooperative gantry cranes with seaside and landside jobs," *Discrete Applied Mathematics*, vol. 242, no. 1, pp. 53–68, 2018.

[34] A. Zhang, W. Zhang, Y. Chen, G. Chen, and X. Chen, "Approximate the scheduling of quay cranes with non-crossing constraints," *European Journal of Operational Research*, vol. 258, no. 3, pp. 820–828, 2017.

[35] M. Fink, G. Desaulniers, M. Frey, F. Kiermaier, R. Kolisch, and F. Soumis, "Column generation for vehicle routing problems with multiple synchronization constraints," *European Journal of Operational Research*, vol. 272, no. 2, pp. 699–711, 2019.

[36] R. Liu, Y. Tao, and X. Xie, "An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits," *Computers & Operations Research*, vol. 101, pp. 250–262, 2019.

[37] R. Soares, A. Marques, P. Amorim, and J. Rasinmäki, "Multiple vehicle synchronisation in a full truck-load pickup and delivery problem: a case-study in the biomass supply chain," *European Journal of Operational Research*, vol. 277, no. 1, pp. 174–194, 2019.

[38] M. Drexl, "Applications of the vehicle routing problem with trailers and transshipments," *European Journal of Operational Research*, vol. 227, no. 2, pp. 275–283, 2013.

[39] M. Drexl, "Synchronization in vehicle routing-A survey of VRPs with multiple synchronization constraints," *Transportation Science*, vol. 46, no. 3, pp. 297–316, 2012.

[40] M. Schiffer, M. Schneider, G. Walther, and G. Laporte, "Vehicle routing and location routing with intermediate stops: a review," *Transportation Science*, vol. 53, 2019.

[41] T. Gschwind and S. Irnich, "Effective handling of dynamic time windows and its application to solving the dial-a-ride problem," *Transportation Science*, vol. 49, no. 2, pp. 335–354, 2015.

[42] T. G. Crainic, N. Ricciardi, and G. Storchi, "Models for evaluating and planning city logistics systems," *Transportation Science*, vol. 43, no. 4, pp. 432–454, 2009.

[43] N. Agatz, P. Bouman, and M. Schmidt, "Optimization approaches for the traveling salesman problem with drone," *Transportation Science*, vol. 52, no. 4, pp. 965–981, 2018.

[44] U. Derigs, M. Pullmann, and U. Vogel, "Truck and trailer routing-Problems, heuristics and computational experience," *Computers & Operations Research*, vol. 40, no. 2, pp. 536–546, 2013.

[45] J. M. Belenguer, E. Benavent, A. Martínez, C. Prins, C. Prodhon, and J. G. Villegas, "A branch-and-cut algorithm for the single truck and trailer routing problem with satellite depots," *Transportation Science*, vol. 50, no. 2, pp. 735–749, 2016.

[46] I.-M. Chao, "A tabu search method for the truck and trailer routing problem," *Computers & Operations Research*, vol. 29, no. 1, pp. 33–51, 2002.

[47] L. M. Pecora and T. L. Carroll, "Synchronization in chaotic systems," *Physical Review Letters*, vol. 64, no. 8, pp. 821–824, 1990.

[48] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[49] S. Wang, Y. Cao, T. Huang, Y. Chen, and S. Wen, "Event-triggered distributed control for synchronization of multiple memristive neural networks under cyber-physical attacks," *Information Sciences*, vol. 518, pp. 361–375, 2020.

[50] C. H. Papadimitriou, "The Euclidean travelling salesman problem is NP-complete," *Theoretical Computer Science*, vol. 4, no. 3, pp. 237–244, 1977.

[51] C. K. H. Lee, "A review of applications of genetic algorithms in operations management," *Engineering Applications of Artificial Intelligence*, vol. 76, no. 1, pp. 1–12, 2018.

[52] Y. Hou, N. Wu, M. Zhou, and Z. Li, "Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 3, pp. 517–530, 2017.

[53] H. Yuan, J. Bi, and M. Zhou, "Multiqueue scheduling of heterogeneous tasks with bounded response time in hybrid green IaaS clouds," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 10, pp. 5404–5412, 2019.

[54] L. Wang and J. Lu, "A memetic algorithm with competition for the capacitated green vehicle routing problem," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 516–526, 2019.

[55] J. F. Gonçalves and M. G. C. Resende, "Biased random-key genetic algorithms for combinatorial optimization," *Journal of Heuristics*, vol. 17, no. 5, pp. 487–525, 2011.

[56] R. Hinterding, "Gaussian mutation and self-adaption for numeric genetic algorithms," in *Proceedings of the IEEE Conference on Evolutionary Computation*, pp. 384–388, Perth, Australia, December 1995.

[57] N. M. Razali and J. Geraghty, "Genetic algorithm performance with different selection strategiesin solving TSP," in *Proceedings of the World Congress on Engineering 2011, WCE 2011*, pp. 1134–1139, London, U.K., July 2011.