Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 4739376, 16 pages
https://doi.org/10.1155/2019/4739376

Research Article

MILP Model and a Rolling Horizon Algorithm for Crane Scheduling in a Hybrid Storage Container Terminal

Kai Yu and Jingcheng Yang

Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai 200030, China

Correspondence should be addressed to Jingcheng Yang; yangjingcheng@sjtu.edu.cn

Received 28 September 2018; Revised 6 December 2018; Accepted 17 December 2018; Published 8 January 2019

Academic Editor: Ning Sun

This paper investigates the yard crane scheduling problem of a hybrid storage container terminal whose import containers and export containers are stored together in each block. The combination of containers improves the space utilization of a container terminal while it also creates new challenges for the yard crane scheduling. To formulate this problem, we propose a mixed integer linear programming (MILP) model, which jointly optimizes trucks' waiting costs and penalty costs caused by exceeding waiting time thresholds. Considering the NP-completeness of this scheduling problem, we develop an efficient rolling horizon algorithm based on some heuristics to reduce the computation time. Finally, computational studies are carried out to evaluate the performance of our method and the solutions obtained by CPLEX solver are used for benchmarking purposes.

1. Introduction

Containerised transportation, where containers are transferred between different modes of transportation both on the sea side and on the land side, has been playing a significant role in international trade transportation for several decades. Accordingly, yard cranes (YC) are the primary equipment to store and retrieve the containers in each block, which is hard to schedule due to the complexity of operation system. Hence, well-planned YC operations can create many competitive advantages [1].

At typical container terminals, the import containers include the storage requests by the internal trucks and retrieval requests by the external trucks. Meanwhile, the export containers include the retrieval requests by the internal trucks and the storage requests by the external trucks. Commonly, the two types of containers are separately stored in different container blocks in East Asia. However, this separated mode fails to make the most efficient use of precious yard space. As the scale of global trade expanding, the storage capacities of some East Asian container terminals are becoming more and more tight during peak import/export seasons. This may cause huge economic losses and reputational damage to the container terminals. To relieve this problem, a hybrid storage mode (in port operational practice, this mode has been well used by some automated container terminals in western countries, such as *London Gateway* terminal and *Euromax* terminal), under which import containers and export containers are stored together in each container block, is being gradually adopted by some container terminals in East Asia. In practice, this mode makes the scheduling of yard cranes more complex while improving the space utilization of container yards. Therefore, the issue of yard crane scheduling in hybrid storage container terminals needs to be solved urgently, and it has a great research necessity.

As illustrated in Figure 1, it shows a hybrid storage container terminal from an aerial view. A hybrid storage block is commonly equipped with more than one YC. These YCs need to serve the trucks with different requests, including internal and external ones, as fast as possible to decrease truck waiting. The internal trucks are to store or retrieve containers between the block and the quay cranes (QCs), and the external trucks are for the storage and retrieval containers between the gate side and the block. In the hybrid storage block, import and export containers are arranged in a number of rows and bays as shown in Figure 2. During a planning period, the storage requests may be generated by internal

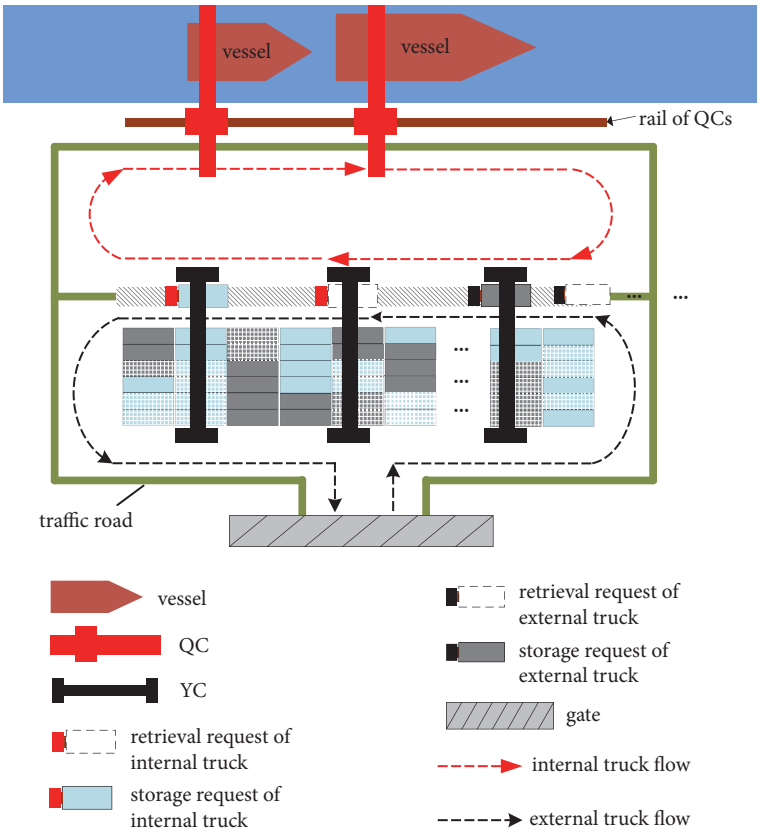


Figure 1: The bird's eye view of a hybrid storage container terminal.

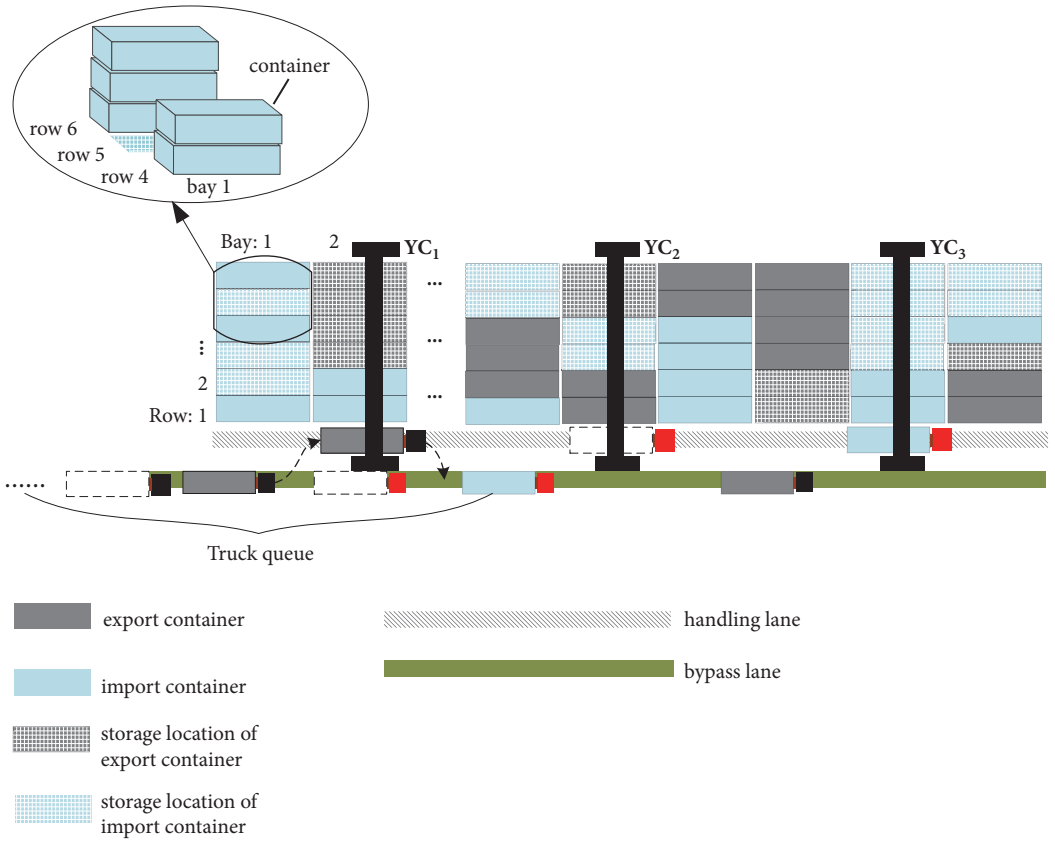


Figure 2: The enlarged view of a hybrid storage block.

or external trucks besides the retrieval requests in the truck queue.

In fact, compared with the crane scheduling in traditional East Asian blocks, the crane scheduling in hybrid storage block (CSHSB) has a novelty. Yard cranes will confront four types of requests instead of just two types. Moreover, for a hybrid storage block, there are more truck types and quantities, which makes lane congestion more likely to occur. Therefore, in the CSHSB problem, each type of requests should be treated differentially and lane congestion should be mitigated. Specifically, we introduce a novel objective function with differential waiting cost coefficients to minimize the waiting cost of requests. Besides, two waiting time thresholds are also proposed to prevent trucks from staying in lanes for too long.

The main contributions of this paper are summarized as follows. First, we formally formulate the CSHSB problem as a novel mixed integer linear programming (MILP) model, where both internal truck requests and external truck requests are considered simultaneously. Second, due to the NP-completeness of the CSHSB problem, we develop an efficient rolling horizon algorithm, which takes advantage of the properties of the problem, to obtain satisfactory solutions (near optimal solutions) in a reasonable amount of time. As the problem size increases, the complexity of the proposed algorithm only grows linearly. Finally, a series of computational studies are performed to demonstrate the efficiency and practicality of our algorithm.

The rest of our paper is organized as follows: Section 2 explains the contributions of this work in light of an extensive literature review of YC scheduling problems. On the basis of problem description, Section 3 formulates our problem as a mixed integer linear programming (MILP) and states some properties about the mathematical model. Section 4 builds upon the problem characteristics to develop a rolling horizon algorithm. This algorithm is tested through computational studies in Section 5. The concluding remarks and future directions are given in Section 6.

2. Literature Review

In this section, we review previous work focusing on the problems of scheduling yard crane(s). This topic has captured increasing attention over the last decade (see, e.g., the literature reviews by Carlo et al. [2]; Gharehgozli et al. [3]; Gharehgozli & Zaerpour [4]; Wang et al. [5]). Firstly, we briefly review some related work dealing with interblock crane scheduling. Then, the crane(s) scheduling within a single container block, which is closely related to our paper, is reviewed comprehensively.

2.1. Interblock YC Scheduling. In practice, for the situation with multiple yard cranes and multiple blocks, jobs in container blocks need to be assigned to the available yard cranes. Therefore, some studies concentrate on the interblock scheduling of yard cranes. Legato et al. [6] propose an optimization model for crane interblock movement, so as to satisfy the crane capacity requirements and minimize the total cost for block matching and crane activation. Moreover, in order to find the best policy for dispatching yard cranes to blocks, they introduce the architecture of an integrated framework, which includes both optimization and simulation techniques. He et al. [7] formulate the yard crane scheduling among container blocks as a goal programming. A hybrid algorithm, which employs some heuristic rules and a parallel genetic algorithm, is proposed to solve the interblock scheduling problem efficiently. Besides, in Chen and Langevin [8], this problem is treated as a mixed integer programming and a taboo (tabu) search algorithm is proposed to quickly obtain near optimal solutions. Furthermore, considering the shortcomings of single-period scheduling, Chang et al. [9] develop a multiperiod model for interblock crane scheduling.

Recently, green operation in maritime ports has received a lot of attention. In fact, yard cranes are large-scale device, so that yard cranes need a lot of energy to move between container blocks. In order to seek a trade-off between efficiency and energy consumption, He et al. [10] convert the interblock yard crane scheduling problem into a vehicle routing problem with soft time windows and propose a combined algorithm that integrates the genetic algorithm and particle swarm optimization algorithm to find the satisfied solutions. Similarly, Sha et al. [11] present an integer programming model with the purpose of minimizing the total energy consumption to optimize the interblock yard crane scheduling.

2.2. YC Scheduling within a Single Block. In container yard operations, yard crane scheduling problem within a single block is very relevant to this paper. Incipiently, some researchers start with dispatching a single crane in a container block. Kim and Kim [12] firstly propose a mixed integer programming for the single crane scheduling problem, which is based on the configuration of Asian container terminal (see Figure 1). The retrieval schedule is given by groups of containers and the objective is to minimize crane travel time through the bays. Narasimhan and Palekar [13] show the NP-completeness of the problem, prove some structural properties on the optimal solution, and suggest a branch and bound approach. The best solution of Kim and Kim [14] uses encoding and decoding procedures embedded in a neighbourhood beam search. Later, Lee et al. [15] consider the same problem for two blocks, where there is only one crane in each container block. Moreover, a simulated annealing algorithm is introduced for this problem solving.

Ng and Mak [16] extend the single crane scheduling problem by including storage requests and differentiating the ready time of each request. For the same problem, Guo et al. [17] present an A$*$ search algorithm and a backtracking algorithm to efficiently find a solution that minimizes the average vehicle waiting time. Recently, Gharehgozli et al. [18] consider a unique crane to carry out storage and retrieval requests while minimizing total crane travel time. Due to specific problem properties, they provide a continuous time integer program and design a two-phase approach for the problem. Gharehgozli et al. [19] show that this problem can be solved in polynomial time and develop different polynomial time algorithms for different storage and retrieval systems. Zou et al. [20] consider a compact storage and retrieval

system by using both dedicated and shared storage policies to minimize dual command throughput time. Considering container relocations are not addressed in single crane scheduling problems, Yuan and Tang [21] incorporate a single yard crane scheduling problem with relocations to reduce crane travel and relocations. da Silva et al. [22] only focus on the block retrieval problem to minimize the number of relocations and the unproductive moves of hindering containers. Moreover, Galle et al. [23] integrate the storage, retrieval, and relocation requests into an integer program, so as to jointly optimize current crane travel time and future relocations.

However, in the realistic operations of container terminals, a container block may contain multiple yard cranes. The problem of scheduling multiple cranes within a single block is closest to our paper. Compared to the single crane scheduling within a single block, the multicrane handling system is more complex. Firstly, Lee et al. [24] extend the work of Kim and Kim [12] by simultaneously scheduling several cranes in a container block. An integer programming model with the objective of minimizing the makespan is developed for scheduling multiple yard cranes. Considering the NP-completeness of the problem, they also design a scheduling heuristic to quickly obtain a satisfactory solution. Li et al. [25] develop a continuous-time model with the objective of minimizing the truck waiting time for the problem of scheduling multiple yard cranes. They incorporate several realistic constraints into the scheduling model and offer a heuristic algorithm to quickly obtain the near optimal solution. Similarly, in order to find the near optimal solutions, Wu et al. [26] offer a clustering-reassigning approach for multicrane scheduling. Based on the properties of twin automated cranes system, Gharehgozli et al. [27] formulate the multicrane scheduling problem as a multiple asymmetric generalized traveling salesman problem and develop an adaptive large neighbourhood search heuristic to solve it. In addition, Saini et al. [28] present a stochastic model for two passing dual yard cranes and obtain closed-form expressions for the crane throughput capacity with interference delays. Besides, for both balanced and unbalanced stack configuration, an approximate model is developed to estimate the expected throughput times. Lashkari et al. [29] investigate the problem of scheduling a dual-spreader crane when lifts are subject to a weight limit. They propose a fast method for computing a lower bound on the optimal value and design an efficient heuristic approach to solve the problem. Speer and Fischer [30] focus on the crane cycle times and the comparison between double and triple crane systems.

Finally, besides the mentioned optimization models and algorithms, some other methodologies are also used for yard crane scheduling problems. Yan et al. [31] establish a knowledge-based system to schedule yard cranes. Petering and Murty [32] develop a simulation model to evaluate different crane scheduling strategies. Similarly, for the problem of block crane scheduling, Stahlbock and Voβ [33] introduce a simulation study to test different online algorithms. Recently, Fotuhi et al. [34] propose an agent-based model by using Netlogo to optimize the service efficiency of yard cranes, so as to minimize the total waiting time of external trucks. For crane scheduling, Abourraja et al. [35] present a multiagent simulation model, including an improved strategy which is inspired by ant colony approach.

Overall, papers on yard crane scheduling with different scenarios are abundant. However, no optimization model or algorithm has yet been proposed for the yard crane scheduling in hybrid storage container terminals. Compared with the extant literature, the crane scheduling in hybrid storage block (CSHSB) has some novel characteristics. More specifically, there are more truck and request types in the CSHSB problem. Furthermore, different types need to be treated differently. These novel characteristics bring challenges to problem modelling and solving. In this paper, a MILP model with a set of practical constraints is proposed for the CSHSB problem. Our model incorporates workload balance constraints and waiting time thresholds. To obtain the promising solutions of large-scale instances in a reasonable amount of time, an efficient rolling horizon algorithm is developed. In summary, the problem considered in this article and the methods for solving it appear to be unique.

3. Problem Modelling

3.1. Problem Description. In the CSHSB problem, we suppose that there is a given set of truck requests, including the internal and external ones, with different release times. Here, the release time of a truck request is the moment when the corresponding truck arrives at its handling location. Furthermore, all locations of the truck requests, which are predetermined, are represented by the bay numbers in hybrid storage block (see Figure 2). We consider the arrival information of trucks can be predicted for a relatively short planning period, which is a common setting in the literature [16, 17]. In practice, the advanced traffic information systems (ATIS) that can provide road-users and traffic managers with accurate and reliable real-time information have been widely studied and applied in recent years [17].

In today's yard operations, several cranes are often placed in a container block to handle a series of truck requests. Yard cranes (YCs) need to provide services for successive truck requests as fast as possible to alleviate waiting. This helps in supporting unblocked truck flows to the quay side and gate side to achieve higher productivity. For the CSHSB problem, the waiting costs (per unit time) of internal and external trucks are commonly different. Moreover, to avoid the lane congestion of the hybrid storage block (see Figure 2), trucks are not allowed to wait too long in lanes. In this paper, two thresholds are introduced to control the waiting times of internal and external truck requests; then penalty costs will occur if their waiting times exceed the corresponding thresholds. In summary, we aim to minimize the total waiting cost formed by the sum of the general waiting costs and the penalty costs.

In the CSHSB problem, we assume that the number of YCs is smaller than the number of requests (i.e., $m < n$), which is the common situation in reality. Considering the scheduling efficiency, the serious workload imbalance among YCs should be avoided. Next, based on an instance with $m = 2$ and $n = 6$, a solution is shown in Figure 3. In Figure 3,

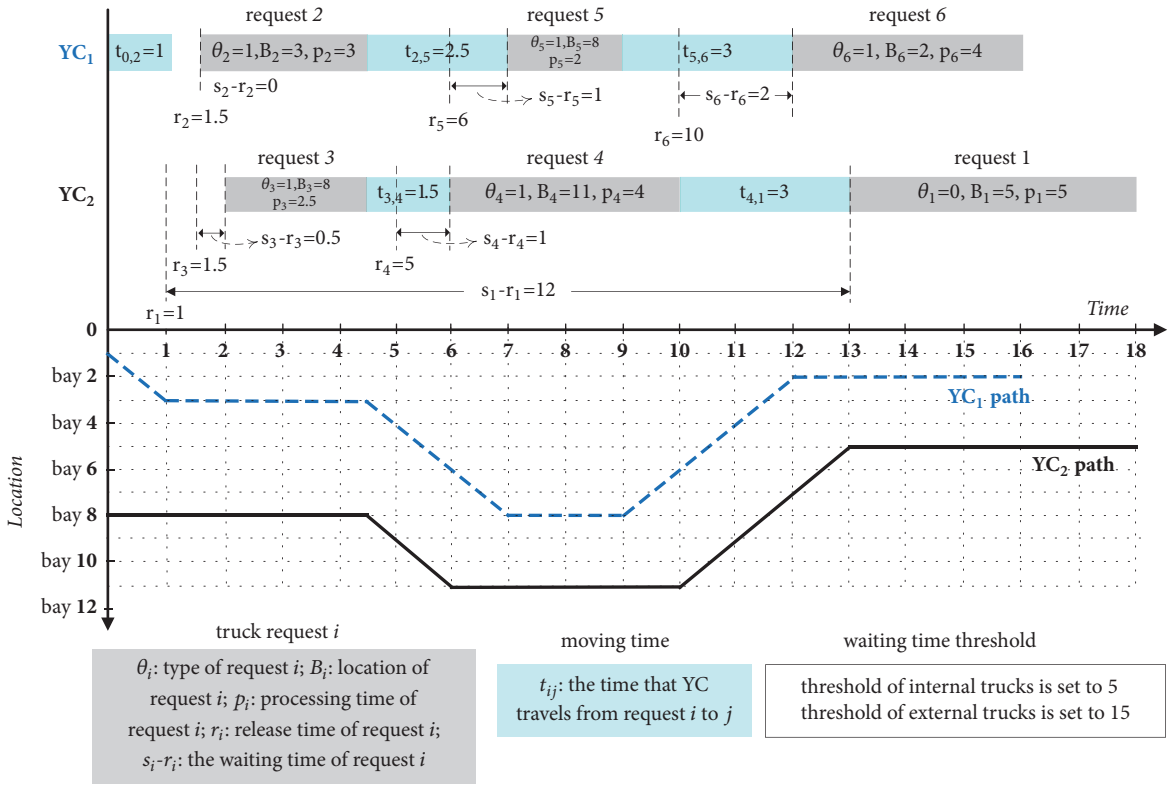


Figure 3: A feasible solution for an instance with 2 cranes and 6 requests.

$\theta_i \in \{0, 1\}$ represents the type of truck request i, where $\theta_i = 1$, if truck request i is internal truck; $\theta_i = 0$, otherwise. r_i and s_i represent the release time and start time of request i, respectively; p_i is the processing time of request i; t_{ij} denotes the moving time that YC travels from requests i to j. As shown in Figure 3, the solution is YC_1: request *2*⟶ request *5*⟶ request *6*; YC_2: request 3⟶ request *4*⟶ request *1*. Moreover, the waiting times of internal and external trucks are less than their thresholds. Both YC_1 and YC_2 are assigned three requests, so the workload balance is perfectly satisfied. Therefore, this solution is feasible. Finally, the paths of two YCs are also plotted in Figure 3.

3.2. Mixed Integer Linear Programming (MILP). In this section, we provide a MILP model to formalize the CSHSB problem.

3.2.1. Assumptions. The following assumptions are made in the mathematical model:

(1) A truck request can be handled by a YC only if the corresponding truck arrives at its handling location; i.e., each request can only be handled by a YC after the request has been released.

(2) Once a YC starts handling a request, the YC will not handle another request until it completes the request.

(3) The YC moving time between two request locations could be predicted with high accuracy as YC speed is usually quite consistent.

3.2.2. Notations. The notations used in the model are listed as follows.

Parameters

N: the set of all truck requests at the beginning of the planning period (indexed by i, j);

M: the set of YCs that can be scheduled at the beginning of the planning horizon (indexed by k);

n: the total number of the requests during the planning period;

m: the total number of the YCs at the beginning of the planning period;

r_i: the release time of truck request i, $i \in N$;

α: the maximum imbalance level of workload assignment, where $0 < \alpha < 1$;

θ_i: a binary parameter which represents the type of truck request i; $\theta_i = 1$, if request i is internal; $\theta_i = 0$, otherwise;

p_i: the processing time of truck request i, $i \in N$;

B_i: the location (bay number) of request i, $i \in N$; $B_0 = 0$ represents the initial location of each YC;

t_{ij}: the moving time that a YC travels from request i to request j; it can be obtained by the equation $t_{ij} = |B_i - B_j|\delta/v$, $i, j \in N, i \neq j$; t_{0j} denotes the time it takes for a YC to travel from the initial location to the location of truck request j;

δ: the length of each bay;

v: the moving speed of a YC;

CI: unit waiting time cost for each internal truck request;

CE: unit waiting time cost for each external truck request;

$\overline{WTI}$: the waiting time threshold of each internal truck request;

$\overline{WTE}$: the waiting time threshold of each external truck request;

PC_1: unit penalty cost incurs when the waiting time of an internal request exceeds its threshold;

PC_2: unit penalty cost incurs when the waiting time of an external request exceeds its threshold.

Decision Variables

x_{ij}^k: a binary variable satisfying $x_{ij}^k = 1$ if request i precedes request j on YC k, otherwise $x_{ij}^k = 0$; $i, j \in N,\ i \neq j, k \in M$;

s_i: the start time of truck request $i, i \in N$.

3.2.3. Mathematic Formulation

[CSHSB]

$$\begin{aligned}\min \quad & f \\ &= CI \sum_{i \in N} \theta_i \left(s_i - r_i\right) + CE \sum_{i \in N} \left(1 - \theta_i\right) \left(s_i - r_i\right) \\ &\quad + PC_1 \sum_{i \in N} \theta_i \times \max\left\{s_i - r_i - \overline{WTI}, 0\right\} \\ &\quad + PC_2 \sum_{i \in N} \left(1 - \theta_i\right) \times \max\left\{s_i - r_i - \overline{WTE}, 0\right\}\end{aligned} \tag{1}$$

S.T.

$$\sum_{k \in M} \sum_{i \in \{0\} \cup N \setminus j} x_{ij}^k = 1, \quad \forall j \in N, \tag{2}$$

$$\sum_{k \in M} \sum_{j \in N} x_{ij}^k \leq 1, \quad \forall i \in N, \tag{3}$$

$$\sum_{k \in M} x_{0j}^k \leq 1, \quad \forall j \in N, \tag{4}$$

$$\sum_{j \in N} x_{0j}^k = 1, \quad \forall k \in M, \tag{5}$$

$$\sum_{h \in \{0\} \cup N \setminus \mathrm{i}} x_{hi}^k \geq \sum_{j \in N \setminus i} x_{ij}^k, \quad \forall i \in N,\ \forall k \in M, \tag{6}$$

$$x_{ij}^k + x_{ji}^k \leq 1, \quad \forall i, j \in \{0\} \cup N,\ i \neq j,\ \forall k \in M, \tag{7}$$

$$s_i + p_i + t_{ij} \leq s_j + G\left(1 - x_{ij}^k\right), \quad \forall i, j \in \{0\} \cup N,\ i \neq j,\ \forall k \in M, \tag{8}$$

$$s_i \geq r_i, \quad \forall i \in N, \tag{9}$$

$$\sum_{i \in \{0\} \cup N} \sum_{j \in N \setminus i} x_{ij}^k \leq (1 + \alpha) \frac{n}{m}, \quad \forall k \in M, \tag{10}$$

$$\sum_{i \in \{0\} \cup N} \sum_{j \in N \setminus i} x_{ij}^k \geq (1 - \alpha) \frac{n}{m}, \quad \forall k \in M, \tag{11}$$

$$\begin{aligned} s_0 &= 0, \\ r_0 &= 0, \\ p_0 &= 0, \end{aligned} \tag{12}$$

$$s_i \geq 0, \quad \forall i \in N, \tag{13}$$

$$x_{ij}^k \in \{0, 1\}, \quad \forall i \in \{0\} \cup N,\ j \in N,\ i \neq j,\ \forall k \in M. \tag{14}$$

The objective function (1) is to minimize the total waiting cost, including the waiting and penalty costs of all requests. Constraints (2) and (3) ensure that every truck request is assigned to exactly one YC and has exactly one predecessor. Constraints (4) and (5) jointly ensure that all YCs must travel from the initial location to different request locations. Constraint (6) is to control that every truck request has only one successor except the last one. Constraint (7) ensures that the handling orders between any two truck requests are unidirectional. Constraint (8) ensures the consistency of start time and service sequence on each YC. More specifically, for any truck request, the start time of its successor cannot be earlier than its start time plus its processing time and the YC moving time. Constraint (9) ensures that each truck request can be handled by one YC only after the request is released. The workload balance constraint is ensured by constraints (10) and (11). Constraint (12) states that the beginning time of the planning period is zero and ensures that each YC must travel to its first task at time 0. Constraint (13) defines the nonnegative variables, and Constraint (14) defines the binary variables.

Since (1) contains maximum value forms, we can linearize the objective function by substituting $\max\{s_i - r_i - \overline{WTI}, 0\}$ and $\max\{s_i - r_i - \overline{WTE}, 0\}$ with auxiliary variables y_i and z_i, respectively. Then, the above model is equivalently rewritten as a standard Mixed Integer Linear Programming (MILP).

MILP for [CSHSB]

$$\begin{aligned}\min \quad & f \\ &= CI \sum_{i \in N} \theta_i \left(s_i - r_i\right) + CE \sum_{i \in N} \left(1 - \theta_i\right) \left(s_i - r_i\right) \\ &\quad + PC_1 \sum_{i \in N} \theta_i y_i + PC_2 \sum_{i \in N} \left(1 - \theta_i\right) z_i\end{aligned} \tag{15}$$

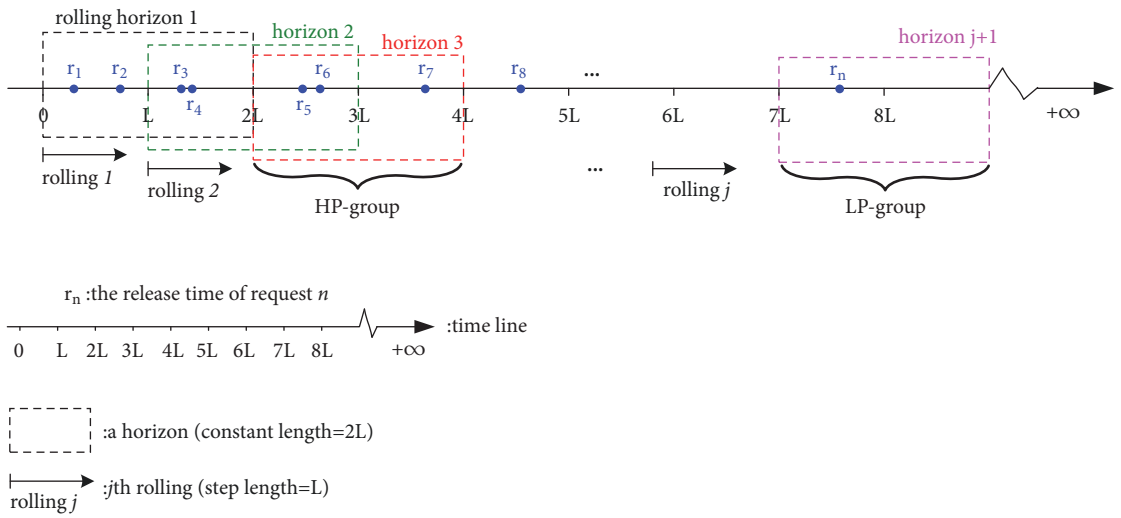


Figure 4: The rolling procedure.

S.T. constraints (2) – (14) and

$$y_i \geq s_i - r_i - \overline{WTI}, \quad \forall i \in N, \tag{16}$$

$$z_i \geq s_i - r_i - \overline{WTE}, \quad \forall i \in N, \tag{17}$$

$$y_i \geq 0,$$

$$z_i \geq 0, \tag{18}$$

$$\forall i \in N.$$

Constraints (16) and (17) indicate that if $s_i - r_i - \overline{WTI} \geq 0$ or $s_i - r_i - \overline{WTE} \geq 0$, variable y_i or z_i, must be greater than or equal to $s_i - r_i - \overline{WTI}$ or $s_i - r_i - \overline{WTE}$, respectively. Otherwise, variable y_i or z_i must be less than or equal to zero.

Note that if the scheduling method obtained by MILP cannot work well in a container terminal, then a feedback is necessary. The operational defects of the obtained scheduling method can be returned to the decision-makers in order to modify MILP's system configuration (e.g., CI, CE, $\overline{WTI}$, $\overline{WTE}$, etc.). This feedback adjustment can guarantee that the further scheduling method works well in practical operations.

Totally, this MILP model contains n^2m binary variables, n continuous variables, and $2n^2m+nm+3m+4n$ constraints. This MILP model will be solved by the commercial optimization software CPLEX in Section 5's computational studies.

As we all know, even the problem of single machine scheduling with diverse release times is NP-complete [36]. Compared to this problem, the scheduling problem we proposed is a more complex multiple machines system. Hence, the CSHSB problem is also NP-complete. In the practical operations of container terminals, the scheduling plans need to be given in time. Therefore, in order to quickly find satisfactory solutions for the CSHSB problem, we develop an efficient rolling horizon algorithm in Section 4.

4. Rolling Horizon Algorithm (RHA)

According to the properties of the CSHSB problem, we intuitively know that an excellent scheduling should try to ensure the waiting times of truck requests below their thresholds, so as to eliminate the penalty costs. Consequently, if a request β is released much later than a request α, then β is likely to be handled later than α; otherwise, the waiting time of the request α will exceed its threshold. Hence, we can divide the release times of all truck requests (the release time of a truck request is the moment that the corresponding truck arrives at its handling location) into multiple horizons. More precisely, the truck requests, whose release times are close with each other, will be assigned to the same horizon. Although the requests in each horizon still need to be sequenced, the computation time of each one is very short. Moreover, the requests in the same horizon with later release times are called low priority (LP) group, and the ones with earlier release times can be named high priority (HP) group. HP-group is more likely to be handled prior to LP-group. Figure 4 illustrates the primary idea of the RHA. As aforementioned, the whole planning period is defined as many short rolling horizons. Accordingly, the whole problem is decomposed into many small subproblems as the rolling proceeds.

The optimal scheduling solution of the whole problem can be obtained by repeatedly invoking the heuristic sequencing of the RHA, which is used to solve each subproblem. In order to propose the RHA clearly, we will introduce several necessary notations in the following section.

4.1. Notations for RHA. Some notations are used to describe the RHA, as follows.

Notations Used in the "Sequencing" Module (See Figure 5). A_j is the set of the requests in horizon j. For $j = 1$, A_1 includes the original requests in horizon 1. For $j \geq 2$, A_j includes the latter half of P_{j-1} and the new requests which are released in the latter half of horizon j.

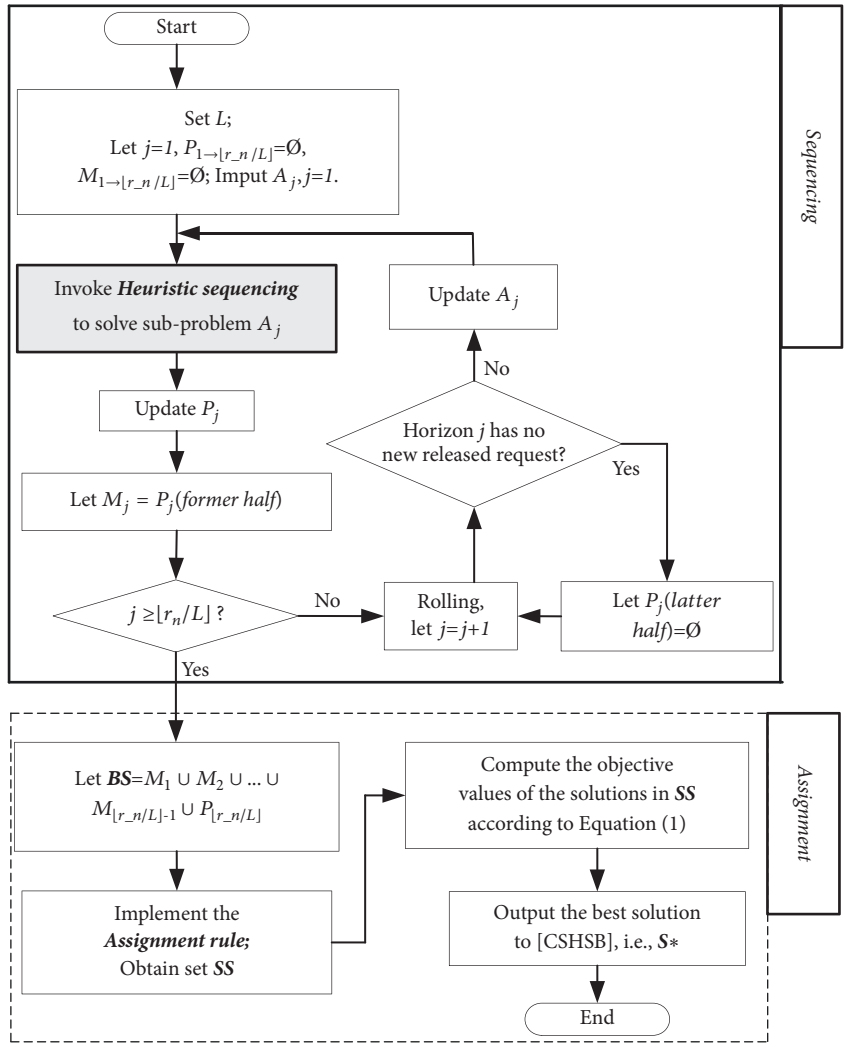


Figure 5: The flowchart of the RHA.

Note that optimizing the handling sequence of the requests in A_j is defined as "**subproblem** A_j".

P_j is the local optimal sequence, which is obtained by using *heuristic sequencing* (see Section 4.3) to solve subproblem A_j.

M_j is the former half of P_j, i.e., $M_j = P_j$ (former half).

BS is the optimal sequence for one YC to handle all requests. More specifically, $= \bigcup_{j=1}^{\lfloor r_n/L \rfloor - 1} M_j \bigcup P_{\lfloor r_n/L \rfloor}$.

The subscript $j = 1, 2, \ldots, \lfloor r_n/L \rfloor$; the number of all subproblems is $\lfloor r_n/L \rfloor$; i.e., the number of loops of the RHA is $\lfloor r_n/L \rfloor$, where r_n is the release time of the last request n (i.e., the arrival time of the last truck), and L is the step length of each rolling. Here, $\lfloor r_n/L \rfloor$ represents the largest integer which is less than or equal to r_n/L.

Notations Involved in the "Assignment" Module. ***SS*** is the set of the satisfactory solutions to CSHSB, which is obtained by imposing the *assignment rule* on ***BS***. The *assignment rule* is elaborated in Section 4.4.

$S*$ is the best solution to the CSHSB problem.

4.2. Flowchart of RHA. Figure 5 shows the flowchart of the RHA. Firstly, the rolling step length L is set. Let both sets $P_{1 \longrightarrow \lfloor r_n/L \rfloor}$ ($P_{1 \longrightarrow \lfloor r_n/L \rfloor}$ is a kind of shorthand notation, which is equivalent to "$P_1, P_2, P_3, \ldots, P_{\lfloor r_n/L \rfloor}$") and $M_{1 \longrightarrow \lfloor r_n/L \rfloor}$ equal null and $j = 1$. After the information of all truck requests is loaded, we invoke the *heuristic sequencing*, which is presented in Section 4.3, to solve the subproblem related to A_j. Next, update P_j by using the solution of subproblem A_j. Furthermore, we record the former half of sequence P_j into M_j, that is, $M_j = P_j$ (former half). As long as $j < \lfloor r_n/L \rfloor$, the rolling horizon will be rolled forward once (i.e., let $j = j + 1$), and the current A_{j+1} is also updated. Again, we invoke the *heuristic sequencing* to solve the subproblem related to A_{j+1}, and then P_{j+1} and M_{j+1} can be obtained.

Once j reaches $\lfloor r_n/L \rfloor$, the rolling will be stopped. Hence, $\forall\ j$, both M_j and P_j are obtained. Then, we shift to the *"Assignment"* module of the RHA. First, the $M_1, M_2, \ldots, M_j$ and $P_{\lfloor r_n/L \rfloor}$ are chained together, so we get the sequence ***BS***; i.e., $\boldsymbol{BS} = \bigcup_{j=1}^{\lfloor r_n/L \rfloor - 1} M_j \bigcup P_{\lfloor r_n/L \rfloor}$. By imposing the *assignment rule* (see Section 4.4) on the sequence ***BS***, we can provide

the CSHSB with the satisfactory solutions, which are saved to the set ***SS***. Moreover, according to (1), we evaluate all the solutions in ***SS*** and pick out the best solution $\boldsymbol{S}*$. Finally, we output the best solution of CSHSB and terminate the RHA.

Here, we briefly analyze the complexity of the proposed RHA. Due to the small scale of the subproblems during each rolling, solving each subproblem usually takes several seconds. Although the decrease in size is accompanied by an increase in the number of subproblems to solve, the total solution time still reduces dramatically. Therefore, it is much better to solve many subproblems than to solve the MILP as a whole. To facilitate theoretical analysis, we define the average time interval between two adjacent requests as $\lambda = r_n/n$, where r_n denotes the release time of the last request n. Then, the number of the requests released in each rolling horizon is $\lfloor L/\lambda \rfloor$; i.e., the average size of each subproblem is $\lfloor L/\lambda \rfloor$. Moreover, the total number of horizons is $\lfloor r_n/L \rfloor = \lfloor n(\lambda/L) \rfloor$; i.e., the total number of subproblems equals $\lfloor n(\lambda/L) \rfloor$. As the whole problem size (n) increases, the total solution time only increases linearly in n, i.e., $O(\lfloor n(\lambda/L) \rfloor * e^{\lfloor L/\lambda \rfloor}) \approx O(n(\lambda/L) * e^{L/\lambda})$. With L/λ much smaller than n, the exponential term, $e^{L/\lambda}$, is much smaller compared to e^n.

Next, we will introduce the *heuristic sequencing* and the *assignment rule*, which are the core parts of the RHA.

4.3. Heuristic Sequencing. In order to solve a subproblem related to A_j, we develop a *heuristic sequencing* based on evolutionary ideas. Each part of the *heuristic sequencing* is as follows.

4.3.1. Solution Representation. We denote $|A_j|$ as the number of the requests in set A_j. The natural number coding is adopted, and then a $|A_j|$-dimensional vector is used to represent a solution of the subproblem related to A_j. The mark numbers of the requests are sorted by their release times. Obviously, the repeated values are forbidden since any truck request can be handled only once, for instance, the four requests in the rolling horizon 1 of Figure 4, that is, A_1= {request 1, request 2, request 3, request 4}. A solution to this subproblem can be represented as follows.

4-dimensional vector: (3, 1, 4, 2).

This implies that the handling sequence is 3 $\longrightarrow$ 1 $\longrightarrow$ 4 $\longrightarrow$ 2.

4.3.2. Initial Solutions. Let G represent the size of the solution set, and g is used to count the number of the initial solutions. The procedure of generating the initial solution set is as follows.

Step 1. Read-in the set A_j, and let $g = 0$.

Step 2. Randomly generate a sequence for handling the requests in A_j.

Step 3. Record an initial solution into the set of the initial solutions and let $g = g + 1$.

Step 4. Repeat Step 2 if $g < G$. Otherwise, output the initial solution set.

4.3.3. Solution Evaluation. A function to evaluate the solutions of each subproblem is required by our *heuristic sequencing*. Since the objective of each subproblem is to minimize the total waiting cost, a solution with smaller objective value implies a superior handling sequence. Therefore, we present an evaluation function as follows:

$$F\left(X_{\widehat{g}}\right) = 1 - \frac{f\left(X_{\widehat{g}}\right)}{\sum_{g=1}^{G} f\left(X_g\right)}, \quad \widehat{g} = 1, 2, \ldots, G, \tag{19}$$

where $f(X_g)$ denotes the objective function value of the solution X_g.

4.3.4. New Solution Creation. To create new solutions for each subproblem, the following procedure is proposed.

(1) Selection. Firstly, we need to choose some solutions from the current solution set. Here, we apply "roulette wheel approach [37]" to select old solutions. As shown in (19), the solutions with higher evaluation score are more likely to be chosen to create new solutions. The procedure of selection is as follows.

Step 1. We calculate the probability of each solution, which is determined by the following equation:

$$Pr\left(X_{\widehat{g}}\right) = \frac{F\left(X_{\widehat{g}}\right)}{\sum_{g=1}^{G} F\left(X_g\right)}, \quad \widehat{g} = 1, 2, \ldots, G \tag{20}$$

Step 2. We calculate the cumulative probability of each solution and randomly pick G numbers from $[0, 1]$.

Step 3. If a randomly picked number is between the cumulative probabilities of two adjacent solutions, the solution with larger cumulative probability is selected.

(2) Generating New Solutions. After selection procedure is finished, we use the following procedure to explore new solutions. Accordingly, a demo is visually shown in Figure 6.

Step 1. In order to mark a position, $d1$ and $d2$ are randomly chosen from *solution1* and *solution2*. Then, the two fragments between $d1$ and $d2$ are preserved, and two incomplete solutions are recorded as a and b.

Step 2. For *solution1* and *solution2*, we implement the cycle-shift from position $d2$.

Step 3. The two fragments between $d1$ and $d2$ are deleted from cycle-shifted *solution2* and *solution1*, respectively. Then, we get two shorter strings, b' and a'.

Step 4. We replace the "×" in a with the elements in b' and the "×" in b with the elements in a'. Finally, two new solutions are generated.

(3) Avoiding Local Optimum. An approach is required to maintain the solution diversification and prevent *heuristic*

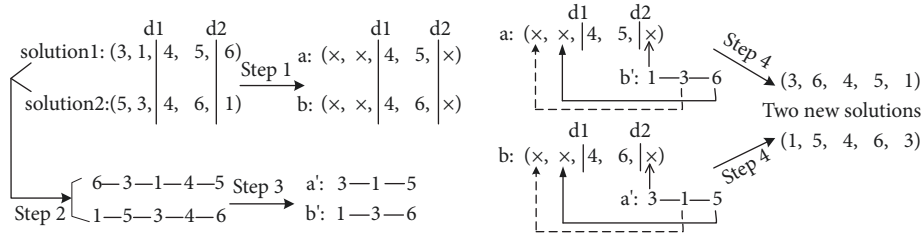


Figure 6: The creation of new solutions.

sequencing from falling into local optimum. In our paper, the "two-point swap" mutation [38–40] is used for the handling sequence of the requests.

4.3.5. Stopping Rules. In order to balance the completeness and efficiency of the *heuristic sequencing*, two stopping rules, "limit iteration" and "no improvement", are proposed.

Let τ_1 denote the number of iterations and τ_2 denote the number of iterations in which the current best solution remains unchanged. A preset maximum number of iterations can be expressed as *MaxI*. Furthermore, "limit iteration" implies the *heuristic sequencing* must be stopped once $\tau_1 \geq MaxI$. The "no improvement" means the *heuristic sequencing* will be stopped if $\tau_2 \geq MaxI/3$ (here, for "no improvement" stopping rule, we set 1/3 as the critical level; the critical level can be adjusted based on different preferences).

4.3.6. Procedure of the Heuristic Sequencing. Based on the discussion above, the whole procedure of the *heuristic sequencing* is summarized as follows.

Step 1. Initialize the solution set of a subproblem, $\tau_1 = 1$ and $\tau_2 = 0$.

Step 2. Evaluate each solution, which belongs to the current solution set, through (19).

Step 3. Update the current best solution and record its evaluation score. Let $\tau_2 = 0$ if the evaluation score of the current best solution is increased; otherwise, let $\tau_2 = \tau_2 + 1$.

Step 4. Terminate the *heuristic sequencing* and output the solution of the subproblem if $\tau_1 \geq MaxI$ or $\tau_2 \geq MaxI/3$. Otherwise, let $\tau_1 = \tau_1 + 1$ and turn to Step 5.

Step 5. Implement the selection, creation, and two-point swap techniques to create a new solution set. Then, return to Step 2.

4.4. Assignment Rule. As shown in Figure 5, we can see that once "*Sequencing*" module is stopped, the RHA will switch to "*Assignment*" module. The ***BS*** is given at the start of the "Assignment" module. However, the workload assignment of each YC is not clear, so we present a rule to solve the assignment problem

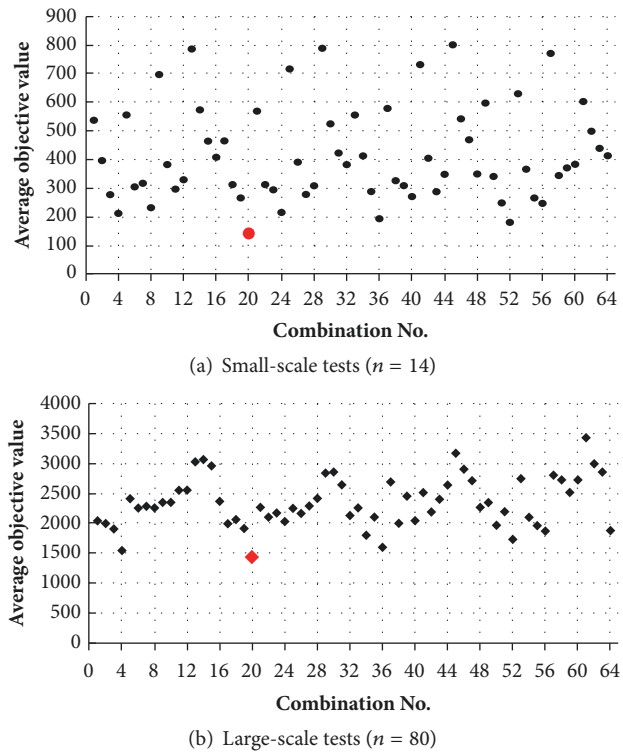
Assignment Rule. Separate the sequence ***BS*** into $\lceil n/m \rceil$ equal-sized pieces, and then the requests in each piece are parallelly assigned to m YCs (Note: $\lceil n/m \rceil$ equals the smallest integer which is larger than n/m). If n/m is not an integer value, then the number of requests in the last piece is smaller than $\lceil n/m \rceil$. In this case, we only need to discretionarily choose Γ YCs from m YCs, and then the requests in the last piece will be assigned to the chosen YCs, where $\Gamma = n - m(\lceil n/m \rceil - 1)$.

After we implement the assignment rule, a series of satisfactory solutions to CSHSB are obtained. These satisfactory solutions are saved in set ***SS***. Clearly, all the solutions in set ***SS*** perfectly meet the workload balance constraints (10) and (11). Finally, we can find the best solution ***S***$*$ from set ***SS*** according to (1), and then the RHA is terminated.

5. Computational Studies

In this section, computational studies are conducted to testify the performance of proposed algorithms. We first describe the test instances generation process. Then, the parameter settings of the RHA are discussed. Finally, the computational results and CPU times are listed, and some analyses are conducted. More specifically, we compare each of the solutions obtained by the proposed RHA, the improved genetic algorithm (IGA), and directly solving the MILP model using CPLEX solver. (So far, the paper most related to our work is that by Zheng et al. [41]. Therefore, the improved genetic algorithm (IGA) they proposed is used as a benchmark. In their study, the IGA is proposed by adopting the adaptive crossover and mutation which are developed by Yan [42]. In the IGA, the range of adaptive crossover probability is set to $[0.4, 0.6]$, the range of adaptive mutation probability is set to $[0.06, 0.1]$, the population size is set to 500, and the IGA is terminated if the number of iterations reaches 1000.) Similarly, their CPU times are also compared to each other. For CPLEX, the default parameter settings are used and the time limit for computing is set to 18000 seconds (i.e., 5 hours). We list the incumbent solutions when the time limits are met. All procedures and algorithms are coded in MATLAB. All computational studies are executed on a PC with CORE i5 CPU of 2.50GHz and 4 GB RAM.

5.1. Generation of Test Instances. In this section, all test instances are randomly generated based on the real scenario of Asian container terminals. A hybrid storage block is set to be a rectangular region with the size of 30 bays and 2 YCs. Both the initial locations of the two YCs are set to *zero*; i.e., $B_0 = 0$. In addition, we set $\delta = 8$ metres (about 26ft) due to the length of the 20ft ISO container. The ratio between the external and internal trucks is equal to 1: 4. The bay numbers



(a) Small-scale tests (n = 14)

(b) Large-scale tests (n = 80)

FIGURE 7: The comparative tests of the RHA parameter settings.

of all requests are generated by a discrete uniform distribution denoted as $U(1, 1, 30)$ (here, U (min, γ, max) is the discrete uniform distribution between min and max with step size γ); i.e., $B_i \sim U(1, 1, 30)$. Considering the storage height and crane technology in East Asia ports, we assume $p_i \sim U(2, 8)$. Additionally, $v = 1$ m/s, $\alpha = 0.2$, $\overline{WTI} = 15$ minutes, $\overline{WTE} = 60$ minutes, $CI = 10$, $CE = 5$, and $PC_1 = PC_2 = 50$.

For small-scale test instances, there are n = 8, 14, 20, 24, and 28, respectively. Moreover, n = 50, 80, and 110 are considered as large-scale. For each scale, the release time r_i (i.e., truck arrival time) follows $U(0, 4n)$. Based on the above scenario and distributions, the proposed algorithm is tested under unfavourable circumstances. Other distributions do not cause significant deviations.

5.2. RHA Parameter Settings. In this section, some tests are necessary to ensure that the RHA parameter settings could yield better performance. Due to the small size of each subproblem, we adequately set G = 60 and $MaxI$ = 100. The rolling step length, the solution creation rate, and the mutation rate are set based on a series of comparative tests. Considering test costs, the rolling step length is limited to 5, 15, 25, or 35; the solution creation rate is limited to 0.20, 0.40, 0.60, or 0.80; the mutation rate is limited to 0.01, 0.05, 0.10, or 0.15. Moreover, we continuously run RHA 20 times under each parameter combination to get the average objective values.

The comparative results of the RHA parameter settings are shown in Figure 7. In Figure 7, the horizontal axis denotes each parameter combination, and the vertical axis represents the average objective value of the corresponding parameter combination. The results indicate that the dominant parameter combination is "*combination 20*". The "*combination 20*" is (*rolling step length*, *solution creation rate*, *mutation rate*) = (15, 0.20, 0.15). Note that there is no guarantee that our choices of parameters are always the best. However, they work well on the numerous test instances. The details of all parameter combinations are placed in the Appendix.

5.3. Computational Results and Analysis. In this section, both the small-scale and large-scale instances are analyzed in the computational studies. We compare the objective values and CPU time obtained by the proposed RHA, the IGA, and CPLEX. As shown in Tables 1 and 2, Gap^1 illustrates the gaps between the results obtained from the proposed RHA and those obtained from CPLEX. Gap^2 represents the gaps between the results obtained from the proposed RHA and the results obtained from the IGA. Moreover, to show the speed advantage of the proposed RHA, we also measure the gaps between the computation times used by the RHA and those used by the IGA. These gaps are listed in the column Gap^{CPU} of Tables 1 and 2.

5.3.1. Small-Scale Instances. For the small-scale (n=8, 14, 20, 24, and 28) instances shown in Table 1, CPLEX usually can obtain the exact solutions by directly solving the MILP we formulated. As illustrated in Table 1, the average gap (Gap^1) between the objective values of CPLEX solutions and the objective values of the RHA is 1.14%. The average gap (Gap^2) between the objective values of the IGA and the objective values of the RHA is 7.30%. Moreover, the average gap (Gap^{CPU}) between the computation times of the IGA and the proposed RHA is 91.24%. Besides, from Table 1, we also can see that the proposed RHA is much faster than CPLEX.

The above results imply that, for small-scale problems, the proposed RHA performs better than the IGA and CPLEX in terms of computation speed and solution quality, on average.

5.3.2. Large-Scale Instances. In this section, the large-scale (n=50, 80, and 110) instances are randomly generated to further compare the results and CPU time obtained by the proposed RHA, the IGA, and CPLEX. For large-scale instances, CPLEX fails to obtain optimal solutions within 5 hours. Therefore, the objective values of the incumbent solutions could be used as a benchmark. From Table 2, we find that the average gap (Gap^1) between the objective values of the RHA and the objective values of the incumbent solutions (when CPLEX runs for 5 hours) is 17.49%. Moreover, the gaps between the IGA solutions and the RHA solutions are also analyzed. As shown in Table 2, the average Gap^2 is 17.39%. Furthermore, the average gap (Gap^{CPU}) between the CPU times of the IGA and the proposed RHA is 58.58%.

We observe that, similarly with the results of the small-scale instances, the proposed RHA also performs much better than the IGA and CPLEX. Besides, for large-scale CSHSB problems, the proposed RHA can drastically improve the efficiency and effectiveness of solving the CSHSB problem.

Table 1: Computational results for small-scale instances.

No.	n	CPLEX		IGA		RHA		Gap^1	Gap^2	Gap^{CPU}
		Obj^{Cplex}	CPU^{Cplex}	Obj^{IGA}	CPU^{IGA}	Obj^{RHA}	CPU^{RHA}			
1	8	5.99	168.16	6.09	50.15	6.09	3.38	-1.67	0.09	93.27
2	8	8.43	107.48	8.53	49.38	8.53	2.97	-1.13	0.00	93.99
3	8	7.28	98.24	7.42	52.12	7.41	3.19	-1.73	0.14	93.89
4	8	8.88	79.36	9.09	46.27	9.10	2.77	-2.48	-0.11	94.02
5	8	6.75	147.35	6.83	53.99	6.83	3.52	-1.08	0.00	93.48
6	8	8.15	198.83	8.28	47.59	8.26	2.80	-1.28	0.25	94.12
Average		**7.58**	**133.24**	**7.71**	**49.92**	**7.70**	**3.10**	**-1.56**	**0.06**	**93.79**
7	14	13.86	3083	15.07	174.36	14.31	9.15	-3.18	5.06	94.75
8	14	13.95	2809	14.79	167.79	14.25	7.95	-2.16	3.62	95.26
9	14	14.16	2987	15.02	171.09	14.60	9.88	-3.14	2.77	94.23
10	14	11.72	3108	13.09	157.08	12.19	10.11	-4.01	6.88	93.56
11	14	14.46	2968	16.26	176.76	14.98	9.12	-3.57	7.88	94.84
12	14	12.05	2876	12.74	170.07	12.47	11.27	-3.44	2.12	93.38
Average		**13.37**	**2971.83**	**14.49**	**169.53**	**13.80**	**9.58**	**-3.25**	**4.72**	**94.34**
13	20	18.46	18000	21.54	346.66	19.35	21.12	-4.81	10.18	93.91
14	20	26.74	16001	32.44	340.99	28.58	19.67	-6.85	11.91	94.23
15	20	31.77	15283	33.39	332.07	32.70	20.84	-2.94	2.07	93.72
16	20	20.80	17854	23.17	345.02	22.14	18.30	-6.41	4.48	94.70
17	20	37.89	14354	43.83	344.16	40.06	19.30	-5.72	8.61	94.39
18	20	31.30	14532	36.50	352.40	33.09	21.92	-5.69	9.35	93.78
Average		**27.83**	**16004.00**	**31.81**	**343.55**	**29.32**	**20.19**	**-5.40**	**7.77**	**94.12**
19	24	48.72	18000	54.33	504.98	46.11	44.84	5.36	15.13	91.12
20	24	46.73	18000	47.65	518.91	42.42	47.27	9.23	10.98	90.89
21	24	56.13	18000	59.38	505.72	52.14	48.35	7.11	12.19	90.44
22	24	56.66	18000	57.83	498.56	55.07	51.84	2.80	4.77	89.60
23	24	43.42	18000	44.43	522.57	39.41	48.44	9.22	11.30	90.73
24	24	44.19	18000	46.65	511.85	42.59	50.63	3.61	8.71	90.11
Average		**49.31**	**18000.00**	**51.71**	**510.43**	**46.29**	**48.56**	**6.22**	**10.51**	**90.48**
25	28	122.39	18000	142.24	702.98	113.57	119.67	7.20	20.15	82.98
26	28	118.42	18000	116.60	681.32	102.88	106.94	13.13	11.77	84.30
27	28	105.11	18000	104.30	682.48	97.45	112.43	7.29	6.57	83.53
28	28	149.28	18000	156.44	680.04	139.27	120.50	6.71	10.98	82.28
29	28	121.37	18000	123.49	676.77	104.94	104.81	13.53	15.02	84.51
30	28	140.07	18000	149.61	724.38	125.47	121.95	10.42	16.13	83.16
Average		**126.11**	**18000.00**	**132.11**	**691.33**	**113.93**	**114.38**	**9.71**	**13.44**	**83.46**
Average								**1.14**	**7.30**	**91.24**

Obj: objective function value; CPU: computation time (second).
$Gap^1 = (Obj^{Cplex}\text{-}Obj^{RHA})/Obj^{Cplex}$ (%).
$Gap^2 = (Obj^{IGA}\text{-}Obj^{RHA})/Obj^{IGA}$ (%).
$Gap^{CPU} = (CPU^{IGA}\text{-}CPU^{RHA})/CPU^{IGA}$ (%).

5.3.3. The Effect of Problem Scale (n) on RHA Performance. Based on the results summarized in Tables 1 and 2, we further analyze how the scale of the CSHSB problem affects the performance of the proposed RHA. As shown in Figure 8, the trends of Gap^1 and Gap^2 are increasing, which indicates that, with the increase of the problem scale (n), the proposed RHA performs better in terms of solution quality than both CPLEX and the IGA. Moreover, the Gap^{CPU} is decreasing in problem scale (n), but the Gap^{CPU} is still higher than 35%. This implies that the proposed RHA saves at least one-third of the IGA's computation time.

In summary, compared to the deterministic optimizer CPLEX, our algorithm is efficient and effective, especially for large-scale CSHSB problems. Besides, compared to an improved genetic algorithm, the proposed RHA also performs well on speed and accuracy.

Table 2: Computational results for large-scale instances.

No.	n	CPLEX	IGA		RHA		Gap^1	Gap^2	Gap^{CPU}
		Obj^{Cplex}	Obj^{IGA}	CPU^{IGA}	Obj^{RHA}	CPU^{RHA}			
31	50	330.39	346.78	1726.18	298.02	410.82	9.80	14.06	76.20
32	50	267.69	256.25	1588.04	241.12	361.38	9.93	5.91	77.24
33	50	295.96	295.31	1877.48	272.04	452.41	8.08	7.88	75.90
34	50	334.29	353.70	1525.58	293.54	405.75	12.19	17.01	73.40
35	50	349.08	436.53	1818.01	319.23	405.36	8.55	26.87	77.70
36	50	289.87	291.69	1605.26	265.20	416.74	8.51	9.08	74.04
Average		**311.21**	**330.04**	**1690.09**	**281.52**	**408.75**	**9.51**	**13.47**	**75.75**
37	80	449.74	437.34	3234.23	389.17	1333.57	13.47	11.01	58.77
38	80	452.70	417.70	3174.24	380.19	1124.53	16.02	8.98	64.57
39	80	476.53	507.52	2895.86	424.92	1291.56	10.83	16.27	55.40
40	80	476.24	472.40	3136.88	414.14	1195.63	13.04	12.33	61.88
41	80	413.72	440.01	3482.26	358.67	1246.72	13.30	18.49	64.20
42	80	458.94	465.64	3104.30	360.62	1088.79	21.42	22.55	64.93
Average		**454.64**	**456.77**	**3171.29**	**387.95**	**1213.47**	**14.68**	**14.94**	**61.62**
43	110	796.92	740.54	4563.32	527.78	3094.35	33.77	28.73	32.19
44	110	635.96	606.28	4378.75	476.83	2769.33	25.02	21.35	36.76
45	110	765.26	711.67	4974.93	539.26	3085.58	29.53	24.23	37.98
46	110	709.63	608.44	4903.29	492.82	2640.43	30.55	19.00	46.15
47	110	801.08	844.62	4663.09	586.76	3042.50	26.75	30.53	34.75
48	110	676.89	632.56	4837.26	514.52	2788.55	23.99	18.66	42.35
Average		**730.95**	**690.68**	**4720.11**	**522.99**	**2903.46**	**28.27**	**23.75**	**38.36**
Average							**17.49**	**17.39**	**58.58**

Obj: objective function value; CPU: computation time (second).
$Gap^1 = (Obj^{Cplex}\text{-}Obj^{RHA})/Obj^{Cplex}$ (%).
$Gap^2 = (Obj^{IGA}\text{-}Obj^{RHA})/Obj^{IGA}$ (%).
$Gap^{CPU} = (CPU^{IGA}\text{-}CPU^{RHA})/CPU^{IGA}$ (%).

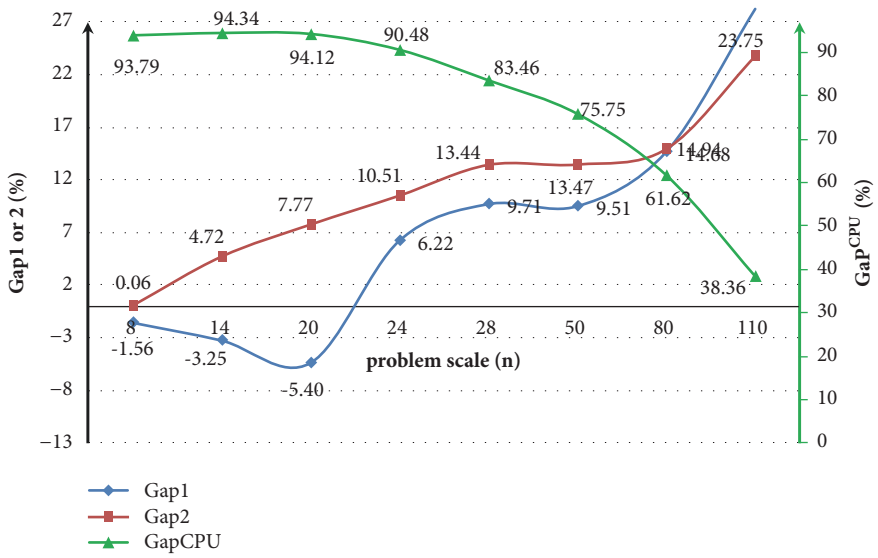


Figure 8: How problem scale (n) influences the performance of the RHA.

Table 3

CombNo.	*L, Rc, Rm*	*CombNo.*	*L, Rc, Rm*	*CombNo.*	*L, Rc, Rm*	*CombNo.*	*L, Rc, Rm*
1	5, 0.2, 0.01	17	15, 0.2, 0.01	33	25, 0.2, 0.01	49	35, 0.2, 0.01
2	5, 0.2, 0.05	18	15, 0.2, 0.05	34	25, 0.2, 0.05	50	35, 0.2, 0.05
3	5, 0.2, 0.1	19	15, 0.2, 0.1	35	25, 0.2, 0.1	51	35, 0.2, 0.1
4	5, 0.2, 0.15	**20**	**15, 0.2, 0.15**	36	25, 0.2, 0.15	52	35, 0.2, 0.15
5	5, 0.4, 0.01	21	15, 0.4, 0.01	37	25, 0.4, 0.01	53	35, 0.4, 0.01
6	5, 0.4, 0.05	22	15, 0.4, 0.05	38	25, 0.4, 0.05	54	35, 0.4, 0.05
7	5, 0.4, 0.1	23	15, 0.4, 0.1	39	25, 0.4, 0.1	55	35, 0.4, 0.1
8	5, 0.4, 0.15	24	15, 0.4, 0.15	40	25, 0.4, 0.15	56	35, 0.4, 0.15
9	5, 0.6, 0.01	25	15, 0.6, 0.01	41	25, 0.6, 0.01	57	35, 0.6, 0.01
10	5, 0.6, 0.05	26	15, 0.6, 0.05	42	25, 0.6, 0.05	58	35, 0.6, 0.05
11	5, 0.6, 0.1	27	15, 0.6, 0.1	43	25, 0.6, 0.1	59	35, 0.6, 0.1
12	5, 0.6, 0.15	28	15, 0.6, 0.15	44	25, 0.6, 0.15	60	35, 0.6, 0.15
13	5, 0.8, 0.01	29	15, 0.8, 0.01	45	25, 0.8, 0.01	61	35, 0.8, 0.01
14	5, 0.8, 0.05	30	15, 0.8, 0.05	46	25, 0.8, 0.05	62	35, 0.8, 0.05
15	5, 0.8, 0.1	31	15, 0.8, 0.1	47	25, 0.8, 0.1	63	35, 0.8, 0.1
16	5, 0.8, 0.15	32	15, 0.8, 0.15	48	25, 0.8, 0.15	64	35, 0.8, 0.15

CombNo.: combination No.; *L*: rolling step length; *Rc*: creation rate; *Rm*: mutation rate.

6. Conclusions

In this paper, we introduce a new yard crane scheduling problem inspired by the hybrid storage mode, an emerging storage mode that has been adopted at some East Asian container ports. The efficient scheduling of the cranes in hybrid storage blocks can improve the space utilization and operational efficiency of a container yard system. We formulate the problem as a mixed integer linear programming (MILP) with workload balance constraints and waiting time thresholds. Because of the NP-completeness of the problem, it can only be solved exactly for small-scale instances. Therefore, we develop an efficient rolling horizon algorithm (RHA) capable of quickly obtaining satisfactory solutions for realistically sized instances. As the problem size increases, the complexity of the RHA only grows linearly.

Our computational studies demonstrate the performance of our methods. For very small-scale instances that can be optimally solved by CPLEX (e.g., n = 8, 14, 20), the gaps between the RHA and optimal solutions are very small. However, for all small instances, the proposed RHA is on average better than CPLEX; moreover the RHA can obtain about 1.14% better results. For large-scale instances, compared to CPLEX truncated after five hours, the RHA converges very quickly and obtains up to 17.49% better results, on average. Besides, for small instances, the proposed RHA also yields results that are 7.30% better compared with a less sophisticated heuristic (i.e., the improved genetic algorithm, IGA). For large instances, our RHA can yield results that are 17.39% better compared with the IGA. Finally, the effect of problem scale (n) on the RHA's performance is explored. We find that the larger the problem scale, the more efficient the algorithm.

Future research could be carried out by considering the container relocations, which is closer to the realistic setting. Another challenge is to optimize the coordination between yard cranes and quay cranes in hybrid storage container terminals. These issues are deserved to be further studied.

Appendix

Details of Algorithm Parameter Combinations

See Table 3.

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 work was supported by the National Natural Science Foundation of China [grant number 71531010].

Supplementary Materials

The supplementary material contains all the test results of the RHA parameter settings. It totally includes 2560 running results. *(Supplementary Materials)*

References

[1] C. Zhang, Y.-W. Wan, J. Liu, and R. J. Linn, "Dynamic crane deployment in container storage yards," *Transportation Research Part B: Methodological*, vol. 36, no. 6, pp. 537–555, 2002.

[2] H. J. Carlo, I. F. A. Vis, and K. J. Roodbergen, "Storage yard operations in container terminals: literature overview, trends, and research directions," *European Journal of Operational Research*, vol. 235, no. 2, pp. 412–430, 2014.

[3] A. H. Gharehgozli, D. Roy, and R. De Koster, "Sea container terminals: New technologies and or models," *Maritime Economics & Logistics*, vol. 18, no. 2, pp. 103–140, 2016.

[4] 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.

[5] K. Wang, L. Zhen, S. Wang, and G. Laporte, "Column generation for the integrated berth allocation, quay crane assignment, and yard assignment problem," *Transportation Science*, vol. 52, no. 4, pp. 812–834, 2018.

[6] P. Legato, P. Canonaco, and R. M. Mazza, "Yard crane management by simulation and optimisation," *Maritime Economics & Logistics*, vol. 11, no. 1, pp. 36–57, 2009.

[7] J. He, D. Chang, W. Mi, and W. Yan, "A hybrid parallel genetic algorithm for yard crane scheduling," *Transportation Research Part E: Logistics and Transportation Review*, vol. 46, no. 1, pp. 136–155, 2010.

[8] L. Chen and A. Langevin, "Multiple yard cranes scheduling for loading operations in a container terminal," *Engineering Optimization*, vol. 43, no. 11, pp. 1205–1221, 2011.

[9] D. Chang, Z. Jiang, W. Yan, and J. He, "Developing a dynamic rolling-horizon decision strategy for yard crane scheduling," *Advanced Engineering Informatics*, vol. 25, no. 3, pp. 485–494, 2011.

[10] 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.

[11] 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, pp. 704–713, 2017.

[12] K. H. Kim and K. Y. Kim, "An optimal routing algorithm for a transfer crane in port container terminals," *Transportation Science*, vol. 33, no. 1, pp. 17–33, 1999.

[13] A. Narasimhan and U. S. Palekar, "Analysis and algorithms for the transtainer routing problem in container port operations," *Transportation Science*, vol. 36, no. 1, pp. 63–78, 2002.

[14] K. Y. Kim and K. H. Kim, "Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals," *Naval Research Logistics (NRL)*, vol. 50, no. 5, pp. 498–514, 2003.

[15] D.-H. Lee, Z. Cao, and Q. Meng, "Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm," *International Journal of Production Economics*, vol. 107, no. 1, pp. 115–124, 2007.

[16] 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.

[17] X. Guo, S. Y. Huang, W. J. Hsu, and M. Y. H. Low, "Dynamic yard crane dispatching in container terminals with predicted vehicle arrival information," *Advanced Engineering Informatics*, vol. 25, no. 3, pp. 472–484, 2011.

[18] 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.

[19] A. H. Gharehgozli, Y. Yu, X. Zhang, and R. De Koster, "Polynomial time algorithms to minimize total travel time in a two-depot automated storage/retrieval system," *Transportation Science*, vol. 51, no. 1, pp. 19–33, 2017.

[20] B. Zou, R. de Koster, and X. Xu, "Operating policies in robotic compact storage and retrieval systems," *Transportation Science*, vol. 52, no. 4, pp. 788–811, 2018.

[21] Y. Yuan and L. Tang, "Novel time-space network flow formulation and approximate dynamic programming approach for the crane scheduling in a coil warehouse," *European Journal of Operational Research*, vol. 262, no. 2, pp. 424–437, 2017.

[22] M. D. M. D. Silva, G. Erdoğan, M. Battarra, and V. Strusevich, "The Block Retrieval Problem," *European Journal of Operational Research*, vol. 265, no. 3, pp. 931–950, 2018.

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

[24] D.-H. Lee, Z. Cao, J. Chen, and J. Cao, "Load scheduling of multiple yard crane systems in container terminal with buffer areas," *Transportation Research Record*, no. 2097, pp. 70–77, 2009.

[25] W. Li, M. Goh, Y. Wu, M. E. H. Petering, R. De Souza, and Y. C. Wu, "A continuous time model for multiple yard crane scheduling with last minute job arrivals," *International Journal of Production Economics*, vol. 136, no. 2, pp. 332–343, 2012.

[26] Y. Wu, W. Li, M. E. H. Petering, M. Goh, and R. De Souza, "Scheduling multiple yard cranes with crane interference and safety distance requirement," *Transportation Science*, vol. 49, no. 4, pp. 990–1005, 2015.

[27] 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.

[28] S. Saini, D. Roy, and R. de Koster, "A stochastic model for the throughput analysis of passing dual yard cranes," *Computers & Operations Research*, vol. 87, pp. 40–51, 2017.

[29] S. Lashkari, Y. Wu, and M. E. H. Petering, "Sequencing dual-spreader crane operations: Mathematical formulation and heuristic algorithm," *European Journal of Operational Research*, vol. 262, no. 2, pp. 521–534, 2017.

[30] U. Speer and K. Fischer, "Scheduling of different automated yard crane systems at container terminals," *Transportation Science*, vol. 51, no. 1, pp. 305–324, 2017.

[31] W. Yan, Y. Huang, D. Chang, and J. He, "An investigation into knowledge-based yard crane scheduling for container terminals," *Advanced Engineering Informatics*, vol. 25, no. 3, pp. 462–471, 2011.

[32] M. E. H. Petering and K. G. Murty, "Effect of block length and yard crane deployment systems on overall performance at a seaport container transshipment terminal," *Computers & Operations Research*, vol. 36, no. 5, pp. 1711–1725, 2009.

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

[34] 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.

[35] M. N. Abourraja, M. Oudani, M. Y. Samiri et al., "A Multi-Agent Based Simulation Model for Rail–Rail Transshipment: An Engineering Approach for Gantry Crane Scheduling," *IEEE Access*, vol. 5, pp. 13142–13156, 2017.

[36] J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker, "Complexity of machine scheduling problems," *Annals of Discrete Mathematics*, vol. 1, no. C, pp. 343–362, 1977.

[37] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, Inc, New York, NY, USA, 2007.

[38] D. Lee, H. Q. Wang, and L. Miao, "Quay crane scheduling with non-interference constraints in port container terminals," *Transportation Research Part E: Logistics and Transportation Review*, vol. 44, no. 1, pp. 124–135, 2008.

[39] M. M. El-Sherbiny, "Particle swarm inspired optimization algorithm without velocity equation," *Egyptian Informatics Journal*, vol. 12, no. 1, pp. 1–8, 2011.

[40] M. M. El-Sherbiny, "Alternate mutation based artificial immune algorithm for step fixed charge transportation problem," *Egyptian Informatics Journal*, vol. 13, no. 2, pp. 123–134, 2012.

[41] H.-X. Zheng, K. Yu, F.-F. Li, and Y. Wang, "Multi-yard cranes scheduling in mixed storage port container terminals considering external container trucks," *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*, vol. 20, no. 12, pp. 3161–3169, 2014.

[42] T. Yan, "An Improved Genetic Algorithm and Its Blending Application with Neural Network," in *Proceedings of the 2010 2nd International Workshop on Intelligent Systems and Applications (ISA)*, pp. 1–4, Wuhan, China, May 2010.