Journal of *Marine Science and Engineering*

MDPI

Article

An Iterative Re-Optimization Framework for the Dynamic Scheduling of Crossover Yard Cranes with Uncertain Delivery Sequences

Yao-Zong Wang and Zhi-Hua Hu *

Logistics Research Center, Shanghai Maritime University, Shanghai 201306, China; 202040510010@stu.shmtu.edu.cn
* Correspondence: zhhu@shmtu.edu.cn

Abstract: In yard-crane scheduling problems, as loading operations take priority over unloading, the delivery sequence of unloading from the quaysides to the yard is uncertain. The delivery sequence changes may make crane scheduling more difficult. As a result, the crane operations schedules developed statically become suboptimal or even infeasible. In this paper, we propose a dynamic scheduling problem considering uncertain delivery sequences. A mixed-integer linear program is developed to assign tasks to cranes and minimize the makespan of crane operations. We propose an iterative solution framework in which the schedules are re-optimized whenever the delivery sequence change is revealed. A genetic algorithm is proposed to solve the problem, and a greedy algorithm is designed to re-optimize and update the solution. To make the updated solution take effect as soon as possible, regarding batch-based task assignment, the tasks in the scheduling period are divided into several batches. In this case, the instant requests arising from the delivery sequence change are added to corresponding batch tasks and re-optimized together with the tasks of this batch. In addition, a relaxation model is formulated to derive a lower bound for demonstrating the performance of the proposed algorithm. Experimental results show that the average gap between the algorithm and the lower bound does not exceed 5%. The greedy insertion algorithm can re-optimize the instant requests in time. Therefore, the proposed iterative re-optimization framework is feasible and has the advantages (necessity) of batch-based task assignment.

Keywords: crossover yard crane; uncertain delivery sequence; dynamic scheduling; genetic algorithm; logistics management

Citation: Wang, Y.-Z.; Hu, Z.-H. An Iterative Re-Optimization Framework for the Dynamic Scheduling of Crossover Yard Cranes with Uncertain Delivery Sequences. *J. Mar. Sci. Eng.* **2023**, *11*, 892. https://doi.org/10.3390/jmse11050892

Academic Editor: M. Dolores Esteban

Received: 9 March 2023
Revised: 19 April 2023
Accepted: 20 April 2023
Published: 22 April 2023

1. Introduction

In automated container terminals (ACTs), the container yard is an important operation area that connects the terminal quayside and inland [1] . The yard is configured with several blocks, each with one or two yard cranes for loading and unloading tasks. However, the complex operations in the yard limit the overall production efficiency of the terminal. To improve efficiency, the ACT configures a crossover yard crane to cooperate with loading and unloading tasks in the yard block. A schematic diagram of a block is shown in Figure 1. The crossover yard cranes, one larger (Crane 1) and one smaller (Crane 2), run on different rail tracks. They can pass each other and move freely from the seaside and landside and in the reverse direction. Hence, the advantage of the crossover yard crane is that both cranes can pass each other and handle any tasks on the range of the yard block.

However, the two cranes are not allowed to perform operations (i.e., drop and lift a container) simultaneously in a certain bay; otherwise, interferences will occur [2]. In addition, the small crane cannot cross the larger crane if the larger one is working. As a result, interferences increase the complexity of modeling the crossover yard-crane scheduling problem. Specifically, the objective is not only to determine the crane scheduling and crane routing but also to prevent crane interferences (i.e., conflict-free scheduling). We

will refer to our problem as the crossover yard-crane scheduling problem. Typically, the terminal divides the area of the yard block into several groups with various attributes for stacking containers Jiang and Jin (2017) [3]. For example, the export containers are stacked close to the seaside for easy loading on vessels, and the import containers are close to the landside for easy exiting. In this paper, we dispatch the crossover yard crane to handle the unloading operations (import tasks).

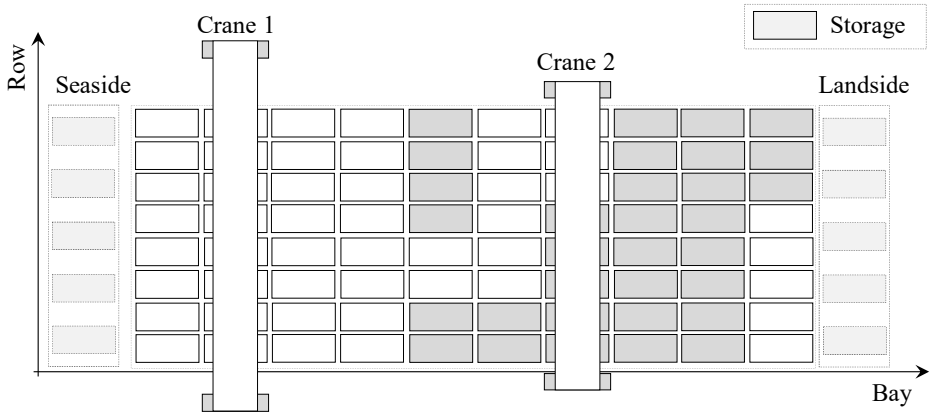


Figure 1. Schematic diagram of a yard block with crossover yard crane.

The priority of loading and unloading operations affects the delivery sequences of import containers to the yard block. In ACTs, the loading and unloading operations are performed simultaneously while vessels are berthing [4]. To optimize the operating time to guarantee the shipping date, in general, the loading operations are given higher priority than unloading (i.e., loading-first). To ensure continuous loading, cranes in the yard should alternately deliver containers to quay cranes [5]. However, continuous loading operations are difficult to guarantee due to too many random factors in the terminal operations. Therefore, the terminals try to relax the other optimization objectives to achieve continuous loading operations as far as possible, e.g., the delivery sequence in the unloading scheme is allowed to be adjusted frequently. As a result, the planned delivery sequence is subject to uncertainty due to the loading-first operational requirement. For example, the task may be handled behind or ahead of the planned schedule, or a container planned to be delivered to the yard block may be delivered to other blocks. Figure 2 illustrates the delivery sequences of the two yard blocks before and after adjustment. In the upper part of Figure 2, ten containers are distributed on a vessel in two tiers. The quay crane first handles the upper tier and then the lower tier. In the scheme, the ten containers are delivered to two yard blocks, and the delivery sequences are shown in Figure 2a. However, in loading-first, the continuous delivery of the export containers from yard block 2# may cause it to be unable to handle import container 5 in time. However, container 5 must be handled so that container 10 can be handled successfully, whereas waiting would cause other exceptions. Hence, the delivery of container 5 to yard block 2# is a feasible solution, as shown in Figure 2b. Therefore, in field operations of ACTs, the delivery sequence from the quay cranes to the yard block is subject to uncertainty.

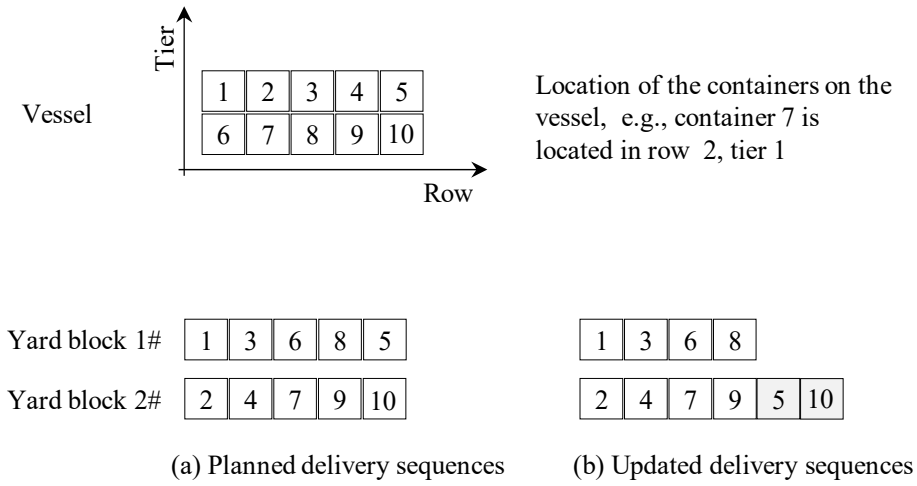


Figure 2. Schematic diagram of the planned delivery sequences and their updates.

The yard operations are subject to uncertain delivery sequences and may cause disturbance to the planned scheme of the crossover yard crane. The planned scheme may become suboptimal or even infeasible under uncertainty [6]. To eliminate as much as possible the impact of uncertainty on production operations, terminals need to constantly adjust the scheme of cranes. As a result, a new set of tasks (i.e., instant request) emerge to adapt to uncertainty. When instant requests occur, the terminals should update the scheme quickly. This paper proposes an iterative solution framework in which the instant request is re-optimized whenever the delivery sequence change is revealed. The dynamic scheduling of the instant request for cranes may be like solving static problems with the latest available data. Therefore, efficient re-optimization is the key to a quick response to the instant request.

Moreover, the re-optimized scheme takes effect as soon as possible to ensure continuous operation. In the literature, heuristic algorithms are widely used for solving dynamic scheduling problems, e.g., [6–10]. Among them, the batch-based task assignment and the rolling-horizon are usually developed to cope with uncertainty and large-scale operations at terminals. Inspired by these approaches, we divide the task to be handled within a planning period into several batches. The scheme of the crossover yard crane is generated within each batch task one by one. When instant requests emerge, we add these requests to an appropriate batch task. Then, the instant requests perform a re-optimization along with the tasks that are already in that batch. Therefore, within the re-optimization framework, the re-optimized scheme is generated quickly to ensure continuous operations. In addition, the batch-based task assignment approach can avoid extensive scheme adjustments and reduce the difficulty and cost of task adjustment in static mode.

In the scheduling of crossover yard cranes, the coupling of operational processes and the interdependence of loading and unloading decisions and uncertainties constitute the complexity [11]. To simplify, Vis and Carlo [12] formulated the crossover yard-crane scheduling problem. [13] abstracted the crane scheduling problem as a multiple asymmetric generalized traveling salesman problem (TSP) with precedence constraints. Chen et al. [14] proposed a multi-commodity network flow model to formulate the integrated scheduling of cranes and automated guided vehicles. The approach of the above studies inspires us to formulate the scheduling problem of the crossover yard crane as a vehicle routing problem with interference constraints. A workload balanced model inspired by [12] is developed to derive a lower model. In addition, interference will occur if two cranes are operating simultaneously at the same bay. To avoid interference, [15] dealt with the interference between two cranes by analyzing the minimum time interval between any two tasks and representing the interference relationship with a matrix. They developed a greedy-based insertion algorithm to re-optimize infeasible solutions. Pre-defining the obstacle graph is another approach to avoiding interference. For example, Briskorn and Angeloudis [16]

pre-defined the rules for crane avoidance when interference occurs and separated the scheduling problem using a decomposition strategy. This paper analyzes two forms of interference and proposes an avoidance strategy and its exact calculation methods.

The scheduling problem of cranes is NP-hard. Most studies attempt to develop a variety of algorithms to solve large-scale problems. We summarize the algorithm in three aspects: (1) Model-based mathematical heuristics, e.g., Branch-and-Bound [14] and Branch-and-Cut [2]; (2) neighborhood search-based heuristics, e.g., greedy [15]; and (3) evolutionary algorithms based on swarm optimization, e.g., genetic algorithm [17] and Particle swarm optimization [18]. Inspired by the methods, a Branch-and-Bound and a genetic algorithm are proposed for solving the planned scheme with small-scale and medium-scale instances, respectively. The instant requests are re-optimized by a greedy insertion algorithm. The proposed algorithms run within the iterative re-optimization framework.

The uncertain delivery sequence may make the planned scheme of crossover yard crane suboptimal or even infeasible. We propose an iterative re-optimization framework for the dynamic scheduling of cranes. We propose mixed-integer linear programming to solve the solution (i.e., the sequence of the cranes). A genetic algorithm is developed to solve the planned scheme, and a greedy-based insertion algorithm is designed to re-optimize the batch tasks that include instant requests. Moreover, a relaxation model is proposed to derive a lower bound for demonstrating the performance of the proposed algorithms. The major contributions of this paper are: (1) proposing a dynamic optimization approach for complex field operations and demonstrating that the greedy algorithm can be used effectively in a dynamic environment; (2) modeling and solving the scheduling problem batch by batch, avoiding extensive adjustments to the planned scheme; and (3) providing insights into how to choose an appropriate duration of the scheme under various workloads at terminals.

The rest of the article is organized as follows. In Section 2, we present a literature review of related studies. In Section 3, we develop a model for scheduling the crossover yard crane and derive a relaxation model to provide a lower bound. Section 4 develops an iterative re-optimization framework that contains a genetic algorithm and a greedy insertion algorithm. Numerical experiments are presented in Section 5 to show that the framework can cope with the dynamic problem. Section 6 concludes this paper.

2. Related Studies

2.1. Crane Scheduling

Operations research on ACTs has gained widespread attention in the field of Port Logistics. An essential research topic is scheduling optimization for operating devices. The aim is to improve production efficiency and thereby reduce the berthing time of the vessel [15]. In the container yard, a block is usually configured with one or two cranes for handling the tasks. There are two kinds of cranes, namely, rail-mounted gantry cranes (RMGs) and rubber-tired gantry cranes (RTGs). The RMG is automatically controlled, and the ASCs (including crossover yard cranes) fall into this category. For a review of research on RTGs, see [19]. The ASCs have the advantage of high-level operational efficiency and low labor costs and are currently used in terminals such as the Hamburg Port and Shanghai Port.

Developing a scheduling plan (i.e., scheme) for cranes is complex. Generally, a mathematical program is developed subject to several assumptions. Even so, the scale of tasks that the model can solve is very limited (no more than 20 tasks). To solve large-scale instances and to improve the applicability of the model, many studies have developed a decomposition-based approach. This approach decomposes the original problem into a master problem and a series of sub-problems, e.g., [2,3,14]. To simplify the solving of sub-problems, multi-stage or multi-layer iterative algorithms are proposed to reach a continuous interaction between the main problem and several sub-problems, e.g., the column generation algorithm continuously generates new columns using an optimality test. Finally, the master problem continues to be improved and solved. In addition, swarm optimization-

based algorithms have advantages in solving scheduling problems. The genetic algorithms (GAs), have adaptive and robust features which improve search diversity while avoiding premature convergence of the algorithm. For example, [15,17,20] used GAs to solve the crane scheduling problem including multiple operating devices. Table 1 lists the pioneering research on yard-crane scheduling. The development of heuristic or intelligent algorithms has been of interest to most studies.

Table 1. Research on yard-crane scheduling.

Literature	The Problem	Objective Function (Minimize)	Model	Algorithm
[11]	Twin-crane	Interference-free	-	-
[16]	Twin-crane	Makespan	MILP	Heuristic
[21]	Crane + YT	Makespan	MILP	BD
[14]	AGV + ASC	Makespan	MCF	B and B
[13]	Twin-crane	Makespan	MILP	ALNS
[17]	Twin-ASC	Makespan	MILP	GA
[22]	Multi-crane	Delay time + Energy consumption	MILP	GA + PSO
[18]	Crane + YT	Makespan	Analytical	GA + PSO
[15]	Twin-crane	Makespan	MILP	GA + Greedy
[3]	SM + Crane	Crane's movements cost + Penalty cost	MIP	B and B + CG
[23]	Crane + YT	Makespan	MILP	DP
[4]	QC + AGV + Crane	Berthing time	CP	Heuristic
[24]	Twin-crane	Makespan	MILP	PSO
[25]	ASC + AGV	Makespan	MILP	GA
[26]	Twin-crane	Makespan	MILP	Heuristic + DP
[2]	Dual-ASC	Makespan	MILP	NS
[27]	Twin-crane	ASC and truck waiting time	MILP	Look-Ahead
[20]	Twin-ASC + AGV	Waiting time + Handling time	MILP	GA
[5]	Crane	Makespan	MILP	Tabu Search
[28]	ASC + AGV	Makespan	MILP	ALNS

ALNS—Adaptive large neighborhood search; B and B—Branch-and-Bound; BD—Benders' decomposition; CG—Column generation; CP—Constraint programming; DP—Dynamic programming; GA—Genetic algorithm; MCF—Multi-commodity network flow; MILP—Mixed-integer linear programming; NS—Neighborhood search; PSO—Particle swarm optimization; QC—Quay crane; SA—Simulated annealing; SM—Storage management; YT—Yard truck.

2.2. Dynamic Optimization

In a real-world system, the constituent elements are interrelated and interlocked, constituting the complexity of the system. Among them, the dynamic nature is a major expression of complexity, and it is always throughout the system's operational process. For example, in ACTs, multi-level operating devices are highly coordinated and orderly when completing the operations. However, the devices and their operational activities affect each other. In particular, if there is a change in an operation, then its related operational activities involve taking appropriate action to respond to the impact of that change. Therefore, the dynamic optimization of operational activities at all levels in response to changing demands is the direction of research into terminal operations.

This section summarizes the studies on three aspects of dynamic optimization, namely, distribution vehicle routing optimization, job shop scheduling, and crew (or crew) staff scheduling for aircraft (or trains). These studies are listed in Table 2. In terms of modeling, Markov decision processes and non-linear programming methods are commonly used to describe dynamic processes. To simplify the solving, mathematically based heuristic algorithms such as column generation and Lagrangian relaxation are used to deal with complex constraints in the model. On the other hand, the dynamic process is formulated as a neural network and solved using machine learning approaches.

Table 2. Research on dynamic optimization.

Literature	The Problem	Model	Algorithm
[29]	Dynamic routing	-	Cluster reschedule
[30]	Railway scheduling with dynamic passenger demand	MILP	ALNS
[31]	Dynamic large-scale urban transportation	Network	Simulation
[32]	Dynamic job shop scheduling	MIP	GA
[7]	Dynamic routing with congestion	Set Covering	Batch assignment
[33]	Dynamic workload management for cranes	-	DSTP
[34]	Railway crew scheduling	ILP	CG + LR
[35]	Airline crew scheduling	QP	CG
[36]	A review of dynamic vehicle routing problems	-	-
[37]	Metro train scheduling dynamic passenger demand	MILP	LR
[6]	Crane scheduling	DP	Heuristic + Exact
[8]	Dynamic routing	-	B and P + CG
[9]	Dynamic job shop scheduling	MILP	PSO
[38]	Dynamic bus scheduling	MIQP	RO
[39]	Dynamic vehicle allocation in ridesharing	MIQP	ADP + LR
[40]	Dynamic job shop scheduling	MDP	DQN
[41]	Dynamic routing	MILP	DDD + LR
[42]	Dynamic routing	MDP	-
[43]	Dynamic job shop scheduling	DRL	PPO
[44]	Dynamic job shop scheduling	MDP	DQN
[10]	Dynamic routing	MILP	Heuristic

ADP—Approximate dynamic programming; B and B—Branch-and-Price; CG—Column generation; GA—Genetic algorithm; MDP: Markov decision process; MICQP—Mixed integer convex quadratic programming; MILP—Mixed-integer linear programming; LP—Linear relaxations algorithm; LR—Lagrangian relaxation; NLP—Non-linear programming; DDD- Dynamic discretization discovery; DP—Dynamic programming; DQN—Deep Q network; DRL—Deep reinforcement learning; DSTP—Dynamic space and time partitioning; PPO—Proximal policy optimization; PSO—Particle swarm optimization; QP—Quadratic programming; RO—Robust optimization.

2.3. A Summary

Crane scheduling problems are complex, and researchers work to come up with new solutions to serve the production practices of terminals. For example, 13 Gharehgozli et al. [13] formulated the crane scheduling problem as a multiple asymmetric generalized traveling salesman problem (TSP) with precedence constraints. Nossack et al. [2] proposed a decomposition approach to solve the scheduling problem of the crossover yard crane. The approach decomposes the problem into two sub-problems and connects them via logic-based Benders' constraints to prevent crane interference. The related literature has provided rich and solid contributions that point the way to our work. Inspired by the literature, we formulate the scheduling problem as a vehicle routing problem with interference constraints. Similarly, the main component is to determine the task assignment of each crane and the operational order (i.e., sequence). The difference is that two cranes should avoid conflict at the same time on the same bay. We provide a rigorous conflict analysis process, which is an innovation of this paper.

In addition, the planned scheme of the cranes is influenced by the loading-first strategy, and the re-optimization of the affected scheme is necessary to ensure the optimality of the operations. In this paper, we study the dynamic scheduling of cranes while considering uncertain delivery sequences. To reduce the impact of the dynamic scheduling on the planned scheme, a batch-based task assignment inspired by Gans and Van Ryzin (1999) and Ozbaygin and Savelsbergh (2019) [7,8] is used to divide the task within the scheduling period into several batch tasks, and the formulations and solving are independent within each batch task. Another innovation is that we propose a solution framework with a genetic algorithm and a greedy insertion algorithm to cope with uncertainty.

3. Formulations

3.1. Problem Definition

This paper investigates a scheduling problem of the crossover yard crane with an uncertain delivery sequence. The tasks to be handled consist of a series of import containers. The import containers enter the yard block at the seaside. Each task has two locations: the starting position (i.e., origin) and the target position (i.e., destination). Origins are located on the seaside, and destinations are located in a certain position in the yard block. The crossover yard-crane configuration is two cranes that can pass each other while handling tasks. Each task is dispatched to exactly one crane. The crane lifts the container from the origin and drops it at the destination. If the cranes are ready to pass each other, the spreader of the larger crane needs to be located at the far-end side of the crane to guarantee effective operations.

However, interferences may occur if the larger crane (Crane 1) works (i.e., lift or drop a container) in a certain bay and the smaller crane (Crane 2) wants to pass or work in the same bay as well, as illustrated in Figure 3. In Figure 3, there are two crossover points for two crane operations, i.e., point A and point B. The two cranes conflict at point A, while point B is conflict-free. In addition, the two cranes are not allowed to work (i.e., lift and drop a container) simultaneously at the seaside or at a certain bay in the yard block.

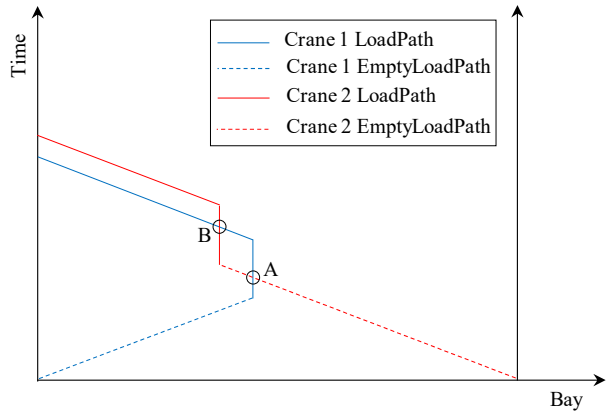


Figure 3. Schematic drawing of interference.

Unlike the universal scheduling problem, the objective is not only to determine the crane scheduling and crane routing but also to prevent crane interferences. Therefore, three major decision problems emerge from the scheduling of crossover yard cranes, namely: (1) crane scheduling: the task is dispatched to the cranes; (2) crane routing: the handling sequence of each crane; and (3) preventing crane interferences: conflict-free scheduling.

The parameters are explained as follows. We address the case of two cranes $K = \{1, 2\}$, where 1 represents the larger crane and 2 represents the smaller crane. The bays in the yard block are denoted by $B = \{0, 1, \ldots, 40\}$, where 0 represents the bay at the seaside. For each task $i \in J$, the origin O_i and destination D_i are known, as explained below. As a result, for each task i, its service time T_i (i.e., the travel time of the crane from the origin to the destination) is known. We assume that the travel time between any two bays is measured in time units. One time unit T_m corresponds to the time required by the crane to move the distance of one container (i.e., one bay)—a default of 4 s. We define T_{ij} as the empty travel time between any two tasks i, j. In addition, T_s denotes the time of lifting and dropping a container)—a default of 30 s, i.e., 7.5 T_m.

Parameters

J: Set of tasks. $J = \{1, 2, \ldots, N\}$, indexed by i, j.
J^-, J^+: $J^- = J \cup \{O\}$ and $J^+ = J \cup \{\overline{O}\}$, where O and $\overline{O}$ indicate the start and finish dummy tasks, respectively.
K: Set of cranes, $K = \{1, 2\}$, indexed by k.
B: Set of bays, $B = \{0, 1, \cdots, 40\}$, indexed by b.
O_i, D_i: The origin/destination of task i, $D_i \in B \backslash \{0\}$, $O_i = 0$.
T_{ij}: Empty travel time between two adjacent tasks i, j. $T_{ij} = \left|O_j - D_i\right| \cdot T_m$.
T_m: Time unit, $u = T_m$.
T_i: Service time of task i.
T_s: The time of lifting or dropping a container by crane, $T_s = 7.5 \cdot u$.

Therefore, crane scheduling and crane routing need to be addressed to find the efficient assignment and sequence of the two cranes and to find efficient combinations of tasks such that the total travels are minimized. The assignment and the sequence constitute the scheme of the crossover yard crane. Meanwhile, the scheme must not contain a conflicted handling path, i.e., conflict-free scheduling.

3.2. The Model

The proposed problem aims to optimize the task assignment and the sequences of cranes by minimizing the makespan. For each task i, three primary decisions need to be clarified, namely, (1) which crane is going to handle task i; (2) at what time is crane k ready to lift task i at its origin; and (3) the order (i.e., sequence) in which crane k should handle the tasks that it is assigned. In addition, the handling path contained in the two sequences should not overlap, i.e., performing interference avoidance if there is interference in the scheme. The decision variables in the model are as follows.

Variables

x_{kij}: $x_{kij} \in \{0, 1\}$. 1 if tasks i and j are handled by crane k sequentially; otherwise, 0.
y_{ki}: $y_{ki} \in \{0, 1\}$. 1 if task i is handled by crane k; otherwise, 0.
a_{ij}: $a_{ij} \in \{0, 1\}$. 1 if task i starts before task j; otherwise, 0.
b_{ij}: $b_{ij} \in \{0, 1\}$. 1 if task i ends before task j; otherwise, 0.
s_{ki}, c_{ki}: The start/completion time of the task i handled by crane k.
s_i, c_i: The start/completion time of the task i.

A mixed-integer linear programming model [M1] is formulated to determine a scheme for both cranes, indicating the order in which all tasks should be handled so that the maximum completion time is minimized (i.e., the makespan). The objective function f_1 is shown in Constraint (1).

[M1]

$$\min\{f_1 | (1); (2) - (18)\}$$

$$f_1 = \min\left\{\max_{k \in K, i \in J}\{c_{ki}\}\right\} \quad (1)$$

(1) Baseline model

The baseline model formulates the scheduling of the crossover yard crane without considering interference. The modeling mechanism of the crane scheduling problem is the same as that of the vehicle routing problem, i.e., both with assignment constraints and order constraints. In the proposed problem, the assignment constraints are the requirement for handling the given tasks, and the order constraints are the requirement for the sequence of operating tasks. The difference is that the demand in the crane scheduling problem is homogeneous and there are no capacity constraints on the cranes (i.e., one-load operations). The segment constraints of the baseline model are shown in Constraints (2)–(12).

Constraint (2) ensures that one and only one task follows the start dummy task and one and only one task leads the finish dummy task. Constraint (3) guarantees that all tasks are handled by one crane. In Constraint (4), the assignment variable (y_{ki}) is presented by the sequencing variable (x_{kij}), namely, task i is handled by crane k if task i is assigned to

crane k. Constraint (5) ensures that the numbers of in-arcs and out-arcs of each task are either one or zero. Constraint (6) determines the start time and completion time of each task. In Constraint (7), the start time of a task is restricted by the completion time of the precedent task handled by the same crane (x_{kij}). Constraints (8) and (9) are upper operating time constraints. Constraint (10) is a minimum value of M_a. Constraints (11) and (12) constrain the ranges of the variables.

$$\sum_{j\in J} x_{k,O,j} = \sum_{i\in J} x_{k,i,\overline{O}} = 1,\ \forall k \in K \tag{2}$$

$$\sum_{k\in K} y_{ki} = 1,\ \forall i \in J \tag{3}$$

$$\sum_{j\in J^+} x_{kij} = y_{ki},\ \forall k \in K, \forall i \in J^- \tag{4}$$

$$\sum_{i\in J^-} x_{kij} = \sum_{i\in J^+} x_{kji},\ \forall k \in K, \forall j \in J \tag{5}$$

$$c_{ki} \geq s_{ki} + T_i + 2{\cdot}T_s + (y_{ki} - 1){\cdot}M_a,\ \forall i \in J, \forall k \in K \tag{6}$$

$$s_{kj} \geq c_{ki} + T_{ij} + \left(x_{kij} - 1\right){\cdot}M_a,\ \forall i \in J^-, j \in J^+, \forall k \in K \tag{7}$$

$$s_{ki} \leq y_{ki}{\cdot}M_a,\ \forall i \in J^-, \forall k \in K \tag{8}$$

$$c_{ki} \leq y_{ki}{\cdot}M_a,\ \forall i \in J^-, \forall k \in K \tag{9}$$

$$M_a = \sum_{ij}\left(T_i + T_{ij}\right) + 2{\cdot}N{\cdot}T_s \tag{10}$$

$$s_{ki}, c_{ki} \geq 0, \forall i \in J, \forall k \in K \tag{11}$$

$$x_{kij}, y_{ki} \in \{0, 1\},\ \forall i, j \in J, \forall k \in K \tag{12}$$

(2) Conflict-free scheduling

The loading and unloading decisions are interdependent due to interference. The solution by Constraints (2)–(12) may be infeasible, i.e., there is conflict in the two sequences. Hence, generating conflict-free sequences in the scheme constrains the time relationship of operations at the same bay. Constraints (13) and (14) represent the start time and completion time of operating task i. Constraints (15) and (16) guarantee that either the start (or completion) time of task i is later or the start (or completion) time of task j is later. Constraint (17) makes sure that the start time of task i is strictly earlier than task j at the seaside. Constraint (18) makes sure that the completion time of task i is strictly earlier than task j on a certain bay in the block. Constraints (19) and (20) constrain the ranges of the variables.

$$s_i = \sum_{k\in K} s_{ki},\ \forall i \in J^- \tag{13}$$

$$c_i = \sum_{k\in K} c_{ki}, \forall i \in J^- \tag{14}$$

$$a_{ij} + a_{ji} = 1, \forall i, j \in J,\ O_i = O_j, i \neq j \tag{15}$$

$$b_{ij} + b_{ji} = 1, \forall i, j \in J,\ D_i = D_j, i \neq j \tag{16}$$

$$s_j \geq s_i + T_s + (a_{ij} - 1) \cdot M_a, \ \forall i, j \in J, O_i = O_j, i \neq j \tag{17}$$

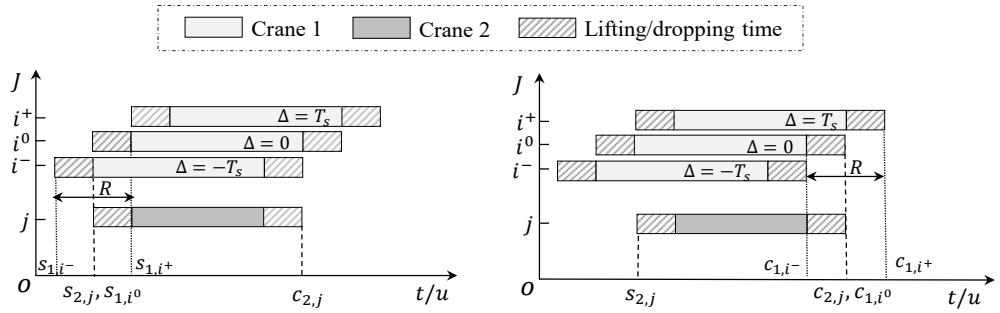
$$c_j \geq c_i + T_s + (b_{ij} - 1) \cdot M_a, \ \forall i, j \in J, D_i = D_j, i \neq j \tag{18}$$

$$s_i, c_i \geq 0, \forall i \in J \tag{19}$$

$$a_{ij}, \ b_{ij} \in \{0, 1\}, \ \forall i, j \in J \tag{20}$$

3.3. Analysis of Variables and Model

In this section, we analyze the validity of Constraints (15) and (16) based on the characteristics of the problem. The critical conditions for conflicts are shown in Figure 4. We use Figure 4b as an example to illustrate the interference relationship between two cranes. Within a certain time horizon R, two cranes preparing to store task i and task j, respectively, in the same bay (i.e., $D_i = D_j$) may conflict. We introduce the task i^-, i^0, i^+ to denote the left-, median-, and right-critical cases of conflict, respectively. Introducing Δ measures the overlap time of conflict, i.e., Δ can be negative, positive, or zero, where $\Delta = c_{1,i^*} - c_{2,j}$, $i^* \in \{i^-, i^0, i^+\}$. Thus, the overlap times Δ between task j and i^-, i^0, i^+ are $-T_s$, 0, T_s, i.e., the length of the diagram between the "two-way arrows" is the time horizon R in which the conflict may occur, where $R = [-T_s, T_s]$, $\Delta \in R$.



(a) Tasks i, j are ready to be lifted at the seaside (b) Tasks i, j are ready to store in the yard block

Figure 4. Diagram of conflict-avoidance constraints and their critical conditions.

The two types of conflict and their avoidance rules are shown in Table 3. Namely, the two tasks are ready to be lifted by the crane at the seaside or are ready to store in a certain bay. When the two cranes are ready to drop containers at the same bay (see Figure 4b), this currently delays one crane's completion time $c_{1,i}$ as it has to wait until the other crane completes its operations. When $\Delta \geq 0$, the waiting time is $(T_s - \Delta)$; otherwise, the waiting time is $T_s + |\Delta|$, i.e., $T_s - \Delta$. Therefore, the completion time of storage operations by Crane 1 is set to $(c_{1,i} + T_s - \Delta)$. Similarly, the avoidance rule in Figure 4a follows the above analysis.

Table 3. Conflict and their avoidance rules.

No.	Type	Overlap Time Δ	Rules	
			$\Delta \geq 0$	$\Delta < 0$
1	Ready to lift	$s_{1,i} - s_{2,j}$	$s_{1,i} \leftarrow s_{1,i} + (T_s - \Delta)$	$s_{1,i} \leftarrow s_{1,i} + T_s + \lvert\Delta\rvert$
2	Ready to store	$c_{1,i} - c_{2,j}$	$c_{1,i} \leftarrow c_{1,i} + (T_s - \Delta)$	$c_{1,i} \leftarrow c_{1,i} + T_s + \lvert\Delta\rvert$

Note: $|\Delta|$ indicates the absolute value when Δ is less than zero.

3.4. A Lower Bound

In the scheduling problem under study, we deal with two cranes that can handle all tasks. In this section, we derive a model to determine a lower bound for the makespan of the two cranes. The key of the model is to solve the sequence problem for a single crane. Vis and Roodbergen [45] proposed an optimal lower-bound derivation method based on dynamic programming, whose objective is to find the shortest path of crane operations that contains all tasks. Vis and Carlo [12] demonstrated that the makespan of the two cranes is approximately equal to half the travel time of a single crane for all tasks. Inspired by the above two approaches, we propose a model for deriving a lower bound considering the workload balanced. In this model, the makespan of the two cranes is half of the operating time for all tasks. We introduce an integer decision variable $u_{ij} \in \{0, 1\}$ to denote that task j is handled by the crane immediately after task i. A relaxed model [M-LB] is developed to derive a lower bound f_{LB}.

The objective function (21) is to minimize the makespan under the workload balanced for both cranes. Constraint (22) ensures that one and only one task follows the start dummy task and one and only one task leads the finish dummy task. The degree constraint (23) imposes that exactly two edges are incident into each vertex associated with a task. In Constraint (24), the operation sequence constraint of ASCs is converted into the well-known generalized sub-tour elimination constraints. However, Constraint (24) has a cardinality growing exponentially with N. Therefore, we relax the constraints containing binary variables $u_{ij} \in \{0, 1\}$ to variables $0 \le \overline{u}_{ij} \le 1$, namely, Constraint (26). The model can be solved by CPLEX. As a result, we find the lower bound for the makespan as a criterion for evaluating the performance of the proposed algorithms.

[M-LB]

$$\min z = \frac{1}{2} \cdot \left(2 \cdot N \cdot T_S + \sum_{i,j \in J} u_{ij} \cdot T_{ij} + \sum_{i \in J} T_i \right) \tag{21}$$

Subject to

$$\sum_{j \in J} u_{O,j} = \sum_{i \in J} u_{i,\overline{O}} = 1 \tag{22}$$

$$\sum_{i<j} u_{ij} + \sum_{j<i} u_{ji} = 2, \forall i \in J \tag{23}$$

$$\sum_{i \in S, j \in S} u_{ij} \le |S| - 1, \forall S \subset J, 2 \le |S| < N - 1 \tag{24}$$

$$u_{ij} \in \{0,\ 1\}, \forall i, j \in J \tag{25}$$

$$0 \le u_{ij} \le 1, \forall i, j \in J \tag{26}$$

4. The Framework and Solution Algorithms

In this paper, we propose a dynamic scheduling problem for a crossover yard crane while considering an uncertain delivery sequence. To solve the problem, we develop an iterative solution to cope with the delivery sequence changes. Initially, the planned scheme is activated and cranes handle the tasks according to the planned scheme. The planned scheme may become suboptimal or even infeasible when instant requests emerge. Hence, the instant requests are re-optimized together with the remaining tasks within the planned scheme to generate a new scheme (i.e., an updated scheme). Then, when the scheme is updated, the cranes handle the updated scheme until the delivery sequence changes again. If a new instant request occurs, we solve another re-optimization problem. In brief, the iterative solution generates an updated batch of schemes whenever the delivery sequence changes. We propose a dynamic optimization process as follows.

Step 1 The given tasks are divided into multiple-batch subtask set $\varphi_i, i \in \{1, 2, \ldots, m\}$. T_i^- and T_i^+ denote the start time and completion time of the scheme φ_i, respectively.

Step 2 At moment 0, three schemes of subtasks are generated, i.e., scheme P_1, P_2, P_3. Set $i \leftarrow 1$.

Step 3 At the current moment t ($t \in [T_i^-, T_i^+]$), one scheme P_{i+3} is generated backwards.

Step 4 **If** an instant request r emerges at moment $I_r \in \left[T_j^-, T_j^+\right]$, $j \in \{i+3, i+4, \ldots, m\}$, then request r is added into the subtask φ_j and this subtask is re-optimized. The subsequent schemes $P_{i+3}, P_{i+4}, \ldots, P_m$ should also be refreshed—return to Step 3. **Otherwise**, Step 5 is performed.

Step 5 Scheme P_i is generated and the current moment t^* is recorded, where t^* is the moment when all tasks in the scheme P_i are completed.

Step 6 The current moment and the progress of the task handling are updated. $t \leftarrow t^*$, $i \leftarrow i+1$. Return to Step 3.

All the tasks in the scheduling period are divided into a multi-batch subtask $\varphi_i, i \in \{1, 2, \ldots, m\}$. During the scheduling period, the scheme of two cranes is generated for all subsets successively. Thus, within each scheme P_i, there is a start time T_i^- and a completion time T_i^+. When an instant request emerges at time t, the request is added to an appropriate subtask φ_i, i.e., $t \in [T_i^-, T_i^+]$). Subsequently, the instant requests are re-scheduled together with the subtask φ_i. The segmented optimization approach effectively reduces the computing time and is helpful for the updated scheme taking effect as soon as possible. In addition, several subtasks after that subtask with instance requests are also refreshed to ensure optimality.

The core of dynamic optimization is "generating schemes and handling tasks simultaneously". Namely, at the same moment, the crane handles tasks within the scheme while generating the scheme for the remaining tasks. Dynamic optimization based on iterative updates is the premise for the design of the solution framework.

4.1. The Iterative Re-Optimization Framework

In ACTs, dynamics and real-time are always present in every process of field operations. Therefore, an updated scheme should be put into effect as soon as possible to maintain the smooth operations of the system. We develop an iterative re-optimization framework combining GA and greedy strategy to solve the proposed dynamic problem, and the structure is shown in Figure 5. The GA is used to generate the planned scheme of each subtask, while the greedy insertion algorithm is used to re-optimize the scheme that includes instant requests and to update the scheme that copes with changes in delivery sequences. It should be noted that the MILP solver (e.g., CPLEX) is used to find the exact solution (i.e., scheme) for small-scale instances. For large-scale instances, the genetic algorithm is applied for solving. The re-optimization problem is solved during the handling process of two cranes, so it is very important to quickly generate the updated scheme. The greedy insertion algorithm has the features of lower complexity and fast search and can insert the instant request into the best position in the sequence based on the initial scheme. As a result, the greedy algorithm can re-optimize and generate an updated scheme with little computing time.

In this framework, the constitutive logic of the algorithm and its closed-loop optimization strategy has three advantages. First, the genetic algorithm, or CPLEX, solves a certain batch task rather than the entire set of tasks. Thus, the solution space is limited to a small space, and the possibility of finding the global optimal solution becomes greater. Second, the greedy algorithm can quickly generate an optimal scheme in its finite neighborhood to complete the re-optimization of the corresponding batch of tasks. Finally, the re-optimized scheme forms a smoother transition with the planned scheme, avoiding large disturbances to crane operations due to extensive adjustments.

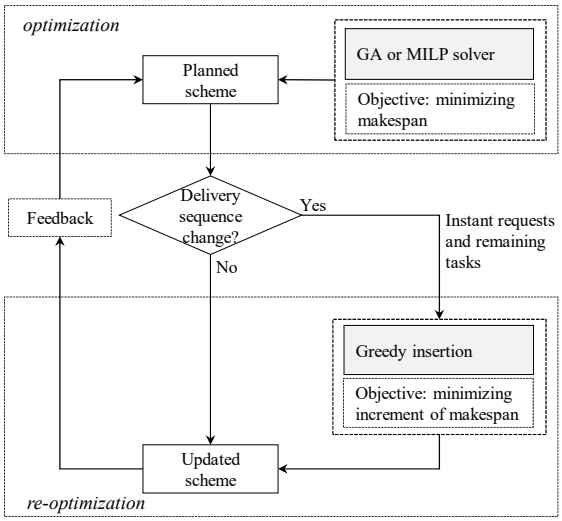


Figure 5. The iterative re-optimization framework and its logic.

4.2. A GA Based on Greedy Insertion

4.2.1. Encoding and Initial Population

In this section, we use randomly arranged encoding vectors to represent the handling sequences of cranes. Figure 6 illustrates the encoding vector (i.e., an individual) with nine points, where each point corresponds to a task and all points are distinct. We cannot know the task assigned to each crane until it is decoded; that is, Crane 1 handles the task set $\{6,3,1,8,4\}$ and Crane 2 handles the task set $\{2,9,7,5\}$, which are determined according to a decoding algorithm (as in Section 4.2.2).

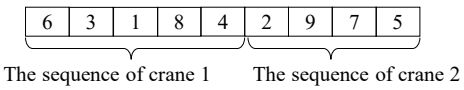


Figure 6. An illustration of encoding vector.

We first randomly generate and evaluate 100 individuals. Then, the first 30 individuals with a smaller makespan were selected as the initial population. In field operations, the population size can be adjusted according to the limitations of the computing time.

4.2.2. Decoding Algorithm

The decoding strategy of the GA is to transform the vector into a sequence of the two cranes. The encoding vector is converted into a feasible sequence of crossover yard cranes subject to sequential constraints and conflict-free constraints. The decoding strategy and its corresponding algorithm (Algorithm 1) are illustrated using a batch task as an example.

Before decoding, the number of tasks assigned to each crane is not determined. A flexible strategy is developed for assigning the optimal task set to each crane. We traverse the encoding vector from the head to the end, successively splitting the points of the individual into two segments (i.e., Step 2 in Algorithm 1). In each traverse, the makespan of sequences is evaluated. Finally, the sequence with a minimum makespan is the optimal solution.

Algorithm 1: Decoding strategy based on random arrangement

Input:	(1) The set and parameter: $[J, K, O_i, D_i]$; (2) random arranged sequence: S_0.
Output:	(1) S_k: the optimal scheme of cranes, $\forall k \in K$; (2) $[s_{ki}, c_{ki}]$: the start and completion time of task i, $\forall i \in J, \forall k \in K$; (3) f: the makespan.
Variables:	F : the set of the makespan.
Steps	
Step 1	Initialization: Let f be a sufficient number. Set $0 \leftarrow s_{ki}, c_{ki}$.
Step 2	**For** w in $W, W = \lvert S_0 \rvert$
Step 2.1	Divide the sequence S_0, and the preceding and succeeding segments are assigned in turn to the two cranes, i.e., $S_1^{**} \leftarrow S_0[:, w]$; $S_2^{**} \leftarrow S_0[w, :]$.
Step 2.2	Calculate the makespan f^{**} of sequence $S_k^{**}, k \in K$ and append f^{**} into F, i.e., $F = F \cup \{f^{**}\}$.
Step 2.3	**End for**
Step 3	Find the minimum f^* and its corresponding sequences S_k^*, i.e., $f^* = argmin(F)$. Calculate the s_{ki}^*, c_{ki}^* in sequence $S_k^*, k \in K$.
Step 4	Detect and avoid conflict.
Step 4.1	**For** i, j in S_1^*, S_2^*
Step 4.1.1	**If** $O_i = O_j$ and $\Delta = \left\lvert s_{1,i}^* - s_{2,j}^* \right\rvert \leq T_s$, then set $s_{1,i}^* \leftarrow s_{1,i}^* + (T_s - \Delta)$;
Step 4.1.2	**Elif** $D_i = D_j$ and $\Delta = \left\lvert c_{1,i}^* - c_{2,j}^* \right\rvert \leq T_s$, then set $c_{1,i}^* \leftarrow c_{1,i}^* + (T_s - \Delta)$.
Step 4.1.3	**End for**
Step 4.2	Calculate the makespan f^{1*}, f^{2*} for the two sequences.
Step 4.3	Set $f^* = \max(f^{1*}, f^{2*})$.
Step 5	If $f^* < f$, then $f \leftarrow f^*$, $S_k \leftarrow S_k^*, s_{ki} \leftarrow s_{ki}^*, c_{ki} \leftarrow c_{ki}^*$.
Step 6	Output $f, S_k, s_{ki}, c_{ki}, i \in J, k \in K$.

Proposition 1. *The computational time complexity of Algorithm 1 is* $O(n + nm - m^2)$, *where n denotes the number of tasks and m denotes the number of tasks assigned to anyone crane* $(m < n)$.

Proof. The decoding algorithm consists of two main steps: task assignment and conflict avoidance. In the worst case, the randomly arranged sequence is split n times, and the two parts of the sequence at the front and back the split obtained are assigned to two cranes, respectively. Thus, the complexity of task assignment is $O(n)$. In conflict avoidance, the sequences assigned to the two cranes are calculated in turn to identify whether they conflict. Suppose the number of tasks assigned to one crane is m, then the other crane is $(n - m)$. Thus, the complexity of task assignment is $O(m(n - m))$. Finally, the computational time complexity of the entire algorithm is $O(n + nm - m^2)$. □

4.2.3. Crossover and Mutation Operations

In GAs, we design a multi-point crossover operator and a 2-opt mutation operator to perform the neighborhood search. The operators are visualized in Figure 7. In brief, one or two individuals are randomly assigned as parents, and the neighborhood generations are executed to generate offspring. The operations of the operators are defined as follows.

In the decoding algorithm, the individual is split into two parts—front and back—as two sequences of crane operations. In the multi-point crossover operator, one point in each of the front and back parts of the individual is selected (the same operation is performed by the other individual), and then two points in the two individuals are exchanged to generate two new individuals. Therefore, the advantage of the multi-point crossover operator is that it can increase the diversity of search. The 2-opt mutation operator focuses on performing operations on the front or back half of an individual's point position. This operation helps to prevent the algorithm from falling into a local optimum solution.

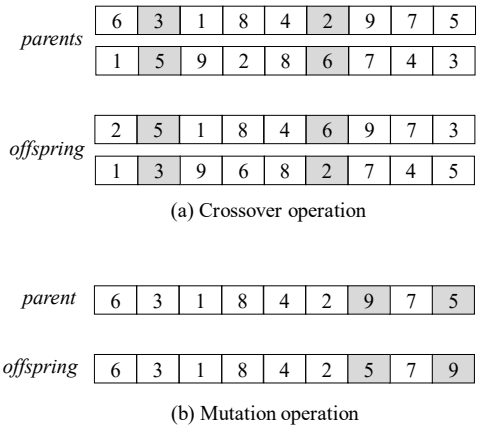


Figure 7. Schematic diagram of crossover and mutation operators.

In addition, while executing the multi-point crossover operator, the point in the individual should be detected to avoid repetition. For example, the duplicate tasks 5# and 6# in the first individual need to be transformed to 3# and 2#. The second individual executes a similar operation.

4.3. Greedy Insertion

A greedy insertion algorithm (Algorithm 2) is developed to re-optimize the scheme. The instant request is added to a batch task close to its emergence time. The instant request (new task) is inserted at the corresponding position in the existing sequence in the scheme based on the principle of minimal incremental makespan, as shown in Figure 8. Multiple new tasks will be sorted by ascending service time T_i, and greedy insertion operations are performed in turn, as in Algorithm 2.

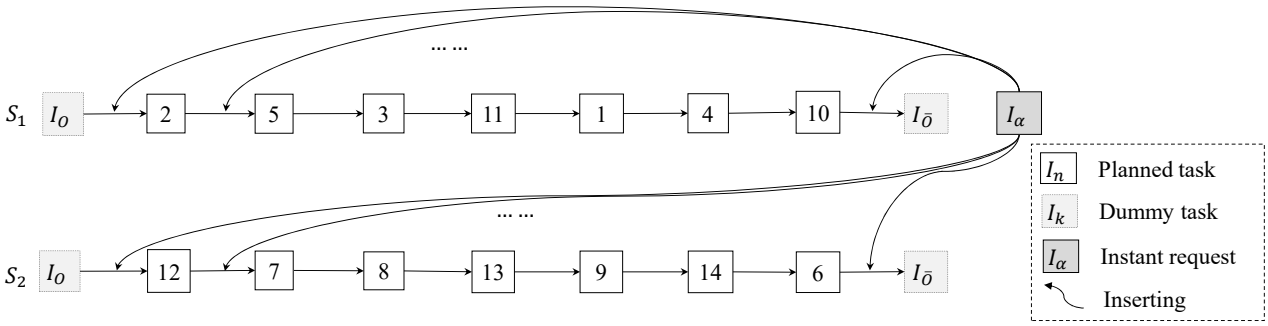


Figure 8. Schematic diagram of re-optimization for instant requests.

Proposition 2. *The computational time complexity of Algorithm 2 is* $O\big((qr + r^2 + 1)\log_2(r)\big)$, *where q denotes the number of p-th scheme, r denotes the number of instant requests, and* $\log_2()$ *denotes the logarithm with base 2.*

Proof. Algorithm 2 consists of two processes, i.e., a sorting operation and a greedy insertion operation. Firstly, sorting the r instant requests by the length of the service has a computational time complexity of $O(r\log_2(r))$. In the greedy insertion operation, suppose the p-th scheme has q tasks, and the q tasks and r instant requests constitute $(q + r)$ tasks. Thus, in the worst case, there are total of $(q + r + 1)$ insertion positions, i.e., $(q + r + 1)$ insertion possibilities. Therefore, the computational time complexity of the entire algorithm is $O\big((qr + r^2 + 1)\log_2(r)\big)$.□

Algorithm 2: The greedy insertion algorithm within the iterative re-optimization framework

Input:	(1) Set of cranes: K; (2) The sequence in planned scheme p: $S_{kp}, \forall k \in K, p \in P$; (3) Set of instant requests: R
Output:	(1) P_p: the p-th updated scheme; (2) $[s_{ki}, c_{ki}]$: the start and completion time of each task i in p-th scheme, $\forall i \in J, \forall k \in K$.
Variable:	(1) T_p^-, T_p^+: the start/completion time of p-th scheme; (2) δ: the minimum lead time for the planned to be re-optimize; (3) τ: the current moment.
Steps	
Step 1	At moment 0, generate a planned scheme for three batch tasks, i.e., the initial scheme $P_p, \forall p \in \{1,2,3\}$. Set $m \leftarrow 1$.
Step 2	The two cranes handle the scheme P_m.
Step 3	Within $\tau \in (T_m^-, T_m^+)$, generate a scheme for one batch task backwards P_{m+3}.
Step 4	If the emergence time of the instant request is close to the start handling time T_p^- of the p-th planned scheme, i.e., $\tau < T_p^- - \delta$, $p = \{m+1, m+2\}$, then re-optimize the p-th planned scheme; otherwise, go to Step 7.
Step 4.1	The instant requests $j \in R$ are arranged in ascending order according to the service time T_j and perform the insert operations in turn. Namely, take $l = argmin_h c_{ki}$, inserting task l to S_{kp}.
Step 4.2	Update the sequence in the p-th scheme, $S_{kp} \leftarrow S_{kp} \cup \{l\}, S_{\bar{k}p} \leftarrow S_{\bar{k}p} \backslash R$.
Step 5	Avoid conflict as Step 4 in Algorithm 1. Update $[s_{ki}, c_{ki}] \leftarrow [s_{ki}^*, c_{ki}^*]$.
Step 6	Generate a re-optimization scheme P_p, $p = \{m+1, m+2\}$.
Step 7	When $\tau \geq T_{m+1}^-$, the scheme P_{m+1} is handled by the two cranes; Set $m \leftarrow m+1$ and return to Step 3.

5. Computational Experiments

5.1. Instances

We generated several instances to demonstrate the effectiveness of the model [M1] and relaxation model [M_{LB}], the performance of Algorithm 1, and the re-optimization performance of Algorithm 2. The computer configuration was Intel (R) Core (TM) i7-10510 CPU @1.8 GHz, 16 G RAM. We used CPLEX as a solver for solving the proposed models.

We provide a generator to generate the datasets (https://github.com/MaritimeYZWang/craneSchedulingModel.git) accessed on 8 March 2023. We generated multiple sets of small-scale instances (i.e., $I_S = \{5, 6, \ldots, 20\}$) and large-scale instances (i.e., $I_L = 30, 40, 50, 75, 100, 125, 150, 175$), where the figures in the instances indicate the number of tasks. The dimensions of the block were 40 bays × 6 stacks × 5 tiers. The movement speed of the crane was set to 3 m/s, i.e., the crane can pass one bay (the length is 12 m) in 4 s (set as a unit). In addition, the lifting or dropping time for a container by crane was set to 30 s (i.e., 7.5 units). The origin O_i and the destination D_i were the two main parameters of task i, and an example is shown in Table 4.

Table 4. An instance ($N = 10$).

No.	O_i	D_i	Bay	No.	O_i	D_i	Bay
1	0	16	16	6	0	16	16
2	0	28	28	7	0	38	38
3	0	32	32	8	0	24	24
4	0	19	19	9	0	16	16
5	0	23	23	10	0	31	31

5.2. Settings

The experimental purpose and settings are shown in Table 5.

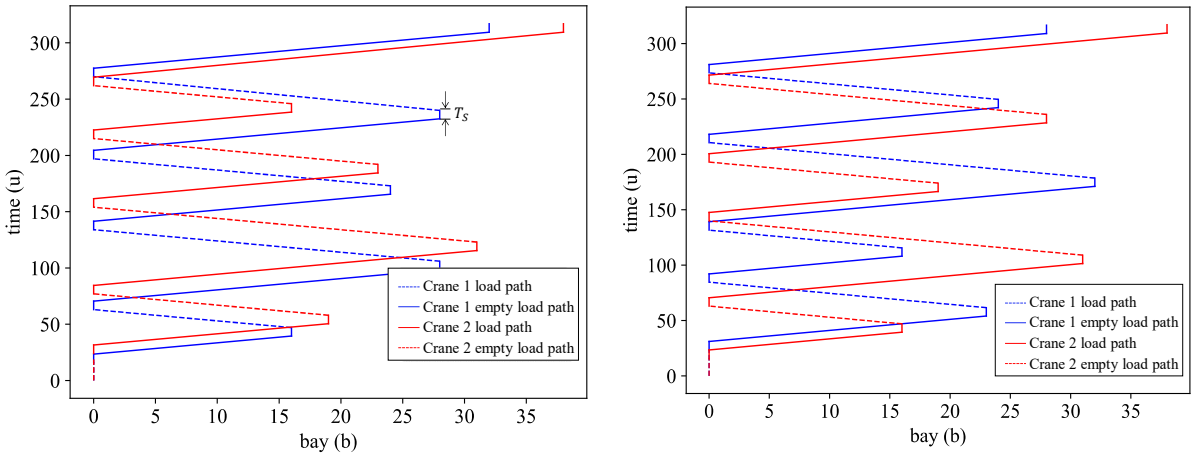
Table 5. Experimental settings.

No.	Purpose	Settings	Results
Exp. 1	Verify the correctness of the models and the validity of the algorithms	a. Use small-scale instances $I_S = \{5, 6, \ldots, 20\}$; b. Solve by CPLEX solver and Algorithm 1.	Table 6 Figures 9 and 10
Exp. 2	Analyze the performance of the algorithms	a. Use large-scale instance $I_L = \{30, 50\}$; b. The crossover probability C is set to $\{0.1, 0.2, \ldots, 0.9\}$; c. The mutation probability M is set to $\{0.1, 0.2, \ldots, 0.9\}$. d. Let Algorithm 1 iterate 100 times in each combination of C and M.	Tables 7 and 8 Figures 11 and 12
Exp. 3	Analyze the impact of the duration of the scheme on the planning results	a. Use large-scale instance $I_L = \{30, 40, 50, 75, 100, 125, 150, 175\}$; b. The duration of the scheme is set to $T = \{300, 600, 900\}$.	Table 9 Figure 13
Exp. 4	Analyze the impact of the level of uncertainty (the volume of instant request) on the planning results	a. Use large-scale instance $I_L = \{30, 50, 100\}$; b. The ratio of the volume of instant requests to the number of tasks is set to $\beta = \{0.1, 0.2, 0.3\}$; c. The duration of the scheme is set to $T = \{300, 600, 900\}$.	Table 10 Figure 14

Table 6. Comparison of the model [M1] and Algorithm 1.

N	*fLB* (*u*)	[M1] (CPLEX Solver)			Algorithm 1 (GA)				Dev 2 (%)
		f_1 (*u*)	CPU (s)	Dev 1 (%)	Min (*u*)	Max (*u*)	Ave (*u*)	CPU (s)	
5	147.00	154.00	0.11	6.82	154.00	157.00	154.03	1.00	0
6	181.00	188.50	0.25	5.57	188.50	194.00	188.64	1.54	0.07
7	209.50	220.00	4.30	5.68	220.00	232.00	220.22	1.53	0.10
8	233.00	241.50	7.58	2.07	241.50	256.00	241.83	1.63	0.14
9	269.50	280.00	135.58	3.75	281.00	295.00	281.31	2.06	0.47
10	307.00	317.00	1278.39	5.05	317.00	328.00	317.27	2.87	0.09
11	334.50	342.50	3600.00	3.21	342.50	357.00	342.88	3.93	0.11
12	371.00	385.50	3600.00	4.41	385.50	401.00	385.87	4.83	0.10
13	402.50	420.00	3600.00	4.52	423.00	438.00	423.20	4.93	0.76
14	437.00	456.50	3600.00	4.27	456.00	474.00	457.00	6.13	0.11
15	473.50	478.00	3600.00	2.51	479.00	496.00	479.33	6.71	0.28
16	495.50	504.50	3600.00	2.38	505.50	523.00	505.81	7.77	0.26
17	522.00	-	3600.00	-	542.00	558.00	542.34	9.14	3.63
18	556.50	-	3600.00	-	583.00	597.00	583.18	10.87	4.11
19	600.00	-	3600.00	-	623.00	640.00	623.46	11.33	4.71
20	639.50	-	3600.00	-	651.00	665.00	652.70	12.04	3.66

Note: Min, Max, and Ave represent the minimum, maximum and average of the makespan; CPU is the computing time; Dev 1 = $(f_1 - f_{LB})/f_{LB} \times 100\%$; Dev 2 = $(\text{Ave} - f_1)/f_1 \times 100\%$.



(**a**) Solved by model [M1] ($N = 10$) (**b**) Solved by GA ($N = 10$)

Figure 9. The working paths of the crossover yard crane.

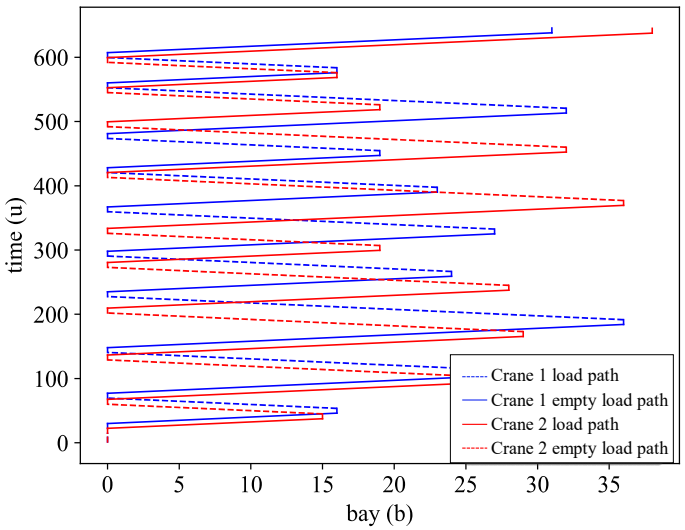


Figure 10. The operation paths of Dual-ASCs solved by Algorithm 1 (N = 20).

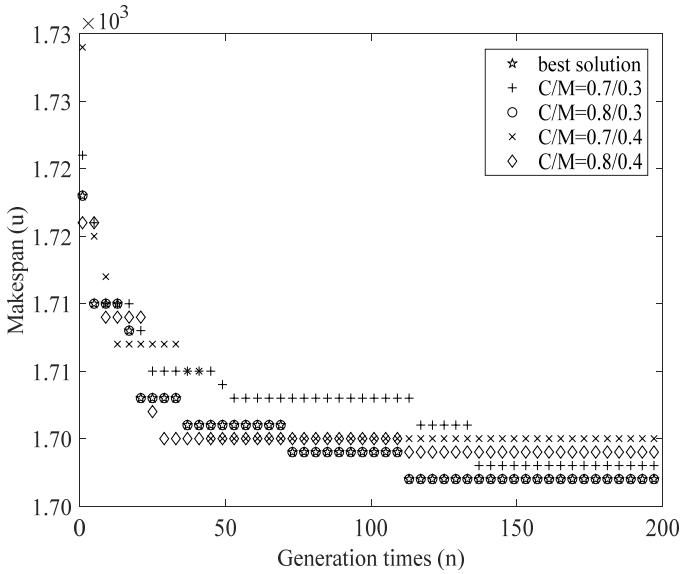


Figure 11. Convergence procedure under multi-parameter combinations.

Table 7. Comparison of the simulated annealing and the proposed GA.

N	Simulated Annealing (SA)		Algorithm 1 (GA)		Gap 1 (%)	Gap 2 (%)
	f_a (u)	CPU_a (s)	f_b (u)	CPU_b (s)		
20	699.50	7.34	651.00	12.04	7.45	−39.04
30	1076.00	27.01	1030.60	16.81	4.41	60.68
40	1391.50	56.03	1306.00	40.58	6.55	38.07
50	1775.50	124.54	1689.50	79.28	5.09	57.09
75	2832.50	415.36	2699.00	256.99	4.95	61.62
100	3881.00	891.11	3772.50	672.93	2.89	32.42
125	4845.00	1755.37	4696.00	1162.30	3.18	51.03
150	6009.00	2782.72	5915.00	2223.17	1.59	25.17
175	7171.50	4957.47	7092.50	3537.56	1.11	40.14

Note: Gap 1 = $(f_a - f_b)/f_b \times 100\%$; Gap 2 = $(CPU_a - CPU_b)/CPU_b \times 100\%$.

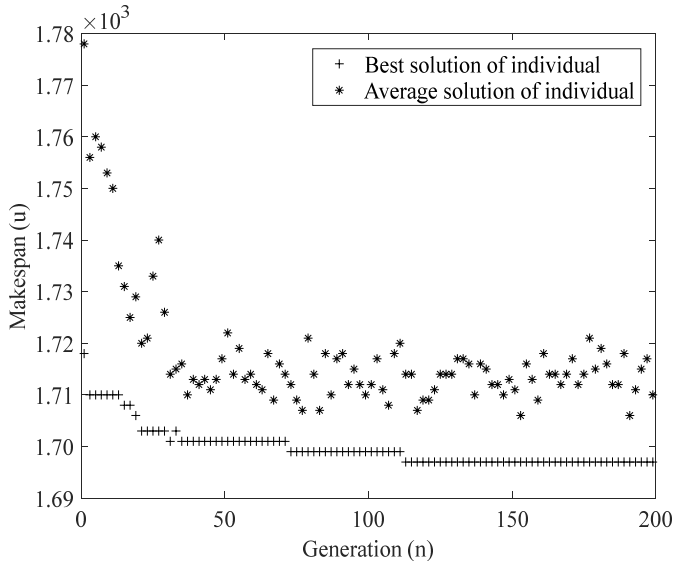


Figure 12. Convergence diagram of genetic algorithm in solving 30 instances.

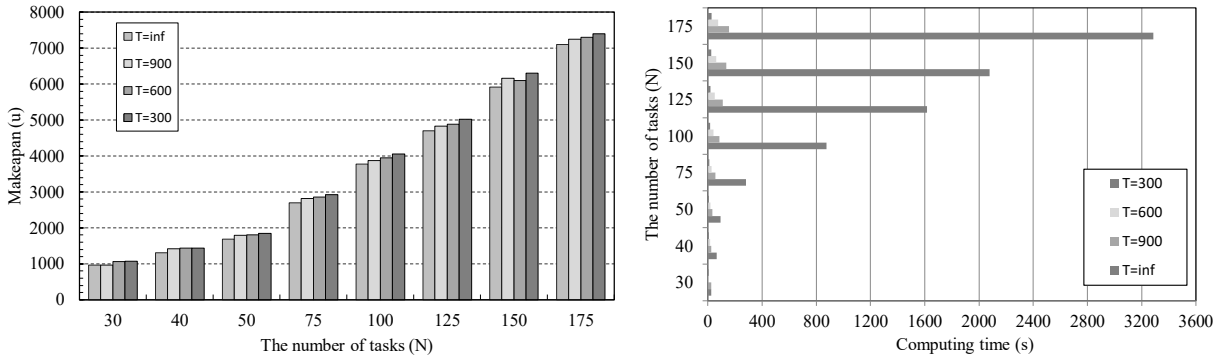


Figure 13. The makespan and computing time under various durations of the scheme.

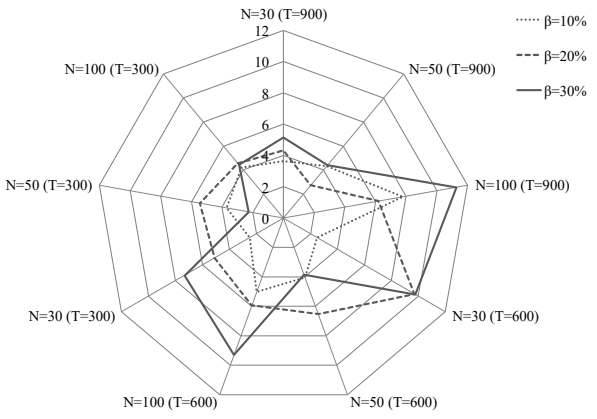


Figure 14. The effect of the uncertain degree and the duration of the scheme on the results.

5.3. Experimental Results

5.3.1. The Demonstration of Model and Algorithm

The working path of two cranes is solved using the MILP solver and Algorithm 1, respectively, as shown in Figure 9. The solid line represents the path of the crane in the loaded state, and the dashed line represents the path of the empty load operation. The

results show that (1) there is no overlap between the operating times of the two cranes at the same bay, and (2) there is no crossover between the load paths (shown by the solid line) and the empty load paths (shown by the dashed line) of the two cranes. The above two points indicate that the conflict-avoidance rules and their constraints are effective. In addition, Algorithm 1 obtains conflict-free paths within 12 s. Hence, initially, it is shown that Algorithm 1 has a good performance. The working path for the instance with task 20 is shown in Figure 10.

The results of [M1] and Algorithm 1 as shown in Table 6. f_{LB} represents a lower bound derived by $[\mathrm{M_{LB}}]$. The results show that the average gap between f_{LB} and the optimal makespan f_1 is 4.19%. Hence, the relaxation model can provide a good lower bound and can be used as a benchmark for evaluating the optimized performance of the model and the proposed algorithms. The average gap between Algorithm 1 and the optimal makespan found by CPLEX is only 1.16% on average (the maximum gap is no more than 5%). The CPLEX is unable to solve the problem within an acceptable time when the number of tasks exceeds 10. However, Algorithm 1 is still able to find the approximate optimal makespan in a shorter time. Therefore, the proposed algorithm has a good performance in small-scale instances. In addition, the performance in large-scale instances was tested in Exp. 2.

5.3.2. Parameter Tuning and Performance

The GA has two main parameters (crossover/mutation probability) whose values affect the convergence ability of the algorithm. We selected several combinations of parameters to find the best convergence ability. Figure 11 represents the solving process in various parameter combinations. When crossover probability (C) and mutation probability (M) are set to 0.8 and 0.3 respectively, the algorithm finds the optimal makespan within 120 generations. The convergence trend of the other parameter combinations is approximately the same as the optimal parameters. Therefore, the proposed algorithm has a stable ability, which is beneficial to extend to the complex field operation at terminals. In addition, Figure 12 shows the solving process for the algorithm taking the optimal parameters. Before 120 generations, the average makespan is approximately uniformly distributed around the current optimal makespan. Therefore, the proposed algorithm has good performance in terms of optimization results, computing time, and stability. Hence, the algorithm can provide a reference in the algorithm design for terminals.

We use large-scale instances (i.e., $N \geq 30$) to further test the performance of the proposed algorithms. Without loss of generality, the parameters near the optimal parameter combinations were taken to reduce the effect of fixed crossover and mutation probabilities on the solving. The results are shown in Tables 8 and 9. The average gap between the proposed Algorithm 1 and the lower bound is no more than 5%. Therefore, the algorithm still has an advantage in solving large-scale instances.

In addition, to further validate the performance of the algorithm, we developed simulated annealing for comparison with the proposed GA. The same decoding strategy is used by both algorithms. The results show that the proposed GA outperforms simulated annealing in terms of solution quality and computing time, as shown in Table 7.

Table 8. Results for each of the 30 tasks of Exp. 2.

No.	f_{LB} (u)	Algorithm 1 (N=30)				Diff Ave (%)
		Min (u)	Max (u)	Ave (u)	CPU (s)	
1	997.50	1050.00	1070.50	1061.57	37.16	6.42
2	1018.00	1078.50	1090.50	1089.62	33.30	7.04
3	1065.70	1113.00	1127.50	1124.34	32.44	5.50
4	959.20	984.00	996.50	998.25	39.03	4.07
5	916.50	946.00	975.00	956.43	32.35	4.36
6	1009.00	1043.00	1060.00	1053.85	38.78	4.44
7	987.50	1017.00	1042.00	1027.80	37.05	4.08
8	1017.50	1046.00	1061.50	1056.45	57.02	3.83
9	960.50	999.00	1025.50	1011.66	40.38	5.33
10	1028.00	1075.50	1090.50	1088.35	41.04	5.87
11	989.00	1028.00	1059.00	1039.99	36.23	5.16
12	1022.70	1034.50	1054.00	1048.58	34.59	2.53
13	1007.00	1035.50	1045.00	1044.06	33.69	3.68
14	1039.50	1075.00	1087.50	1085.87	38.09	4.46
15	993.50	1024.50	1038.00	1036.43	34.11	4.32
16	978.00	1032.00	1045.50	1044.48	36.53	6.80
17	978.00	1000.50	1026.00	1009.52	35.58	3.22
18	936.70	960.00	980.00	971.50	32.41	3.72
19	959.20	1016.50	1041.00	1028.18	34.77	7.19
20	898.00	924.50	947.00	935.43	33.28	4.17

Note: Diff Ave $= (Ave - f_{LB})/f_{LB} \times 100\%$.

5.3.3. Validity Analysis of the Framework

In the iterative re-optimization framework, the given tasks are divided into multiple batch subtasks, so that Algorithm 1 solves the scheme within each batch task one by one. Algorithm 2 quickly assigns the instant requests to a corresponding batch task and re-optimizes the scheme for this batch task. To verify the validity of this framework, we used various durations of the scheme to compare the planning results.

The effect of the duration of the scheme on planning results.

The duration of the scheme T is set to inf, 900, 600, and 300, where "inf" denotes the duration is unlimited. The results are shown in Table 10 and Figure 13. The smaller the duration (i.e., the more batch tasks divided), the greater the makespan, but the computing time is significantly shorter. In the framework for dynamic scheduling, there are only a few tasks in the scheme, which not only reduces the computing time but also reserves time and space for the dynamic scheduling of the instant requests generated by the change of delivery sequences. Therefore, selecting an appropriate duration will help to improve the efficiency of the developing scheme.

Analyze the effect of the volume of instant requests on results.

In this section, we define the degree of uncertainty β, which represents the ratio of the volume of instant requests to the number of tasks in the planned scheme, i.e., $\beta = N_I/N_O$. For example, the volume of instant requests N_I for the instance $N_O = 30$ is 3 when $\beta = 10\%$. A planned scheme is generated using Algorithm 1, and Algorithm 2 is invoked to perform a re-optimization of the tasks including instant requests. The results are shown in Table 11, and their visual representation is shown in Figure 14. The results show that the larger the degree β, the larger the incremental Y of the makespan, and this effect gradually weakens as the duration of the scheme T decreases. There are two possible reasons: (1) as the uncertainty in the delivery sequence increases, re-optimizing increases the disturbance to the planned scheme, and (2) shortening the duration of the scheme helps to assign a more suitable batch task for instant requests and allows the updated scheme to take effect as soon as possible and reduce the disturbance to the planned scheme.

Table 9. Results for each of the 50 tasks of Exp. 2.

No.	f_{LB} (u)	Algorithm 1 (N=50)				Diff Ave (%)
		Min (u)	Max (u)	Ave (u)	CPU (s)	
1	1694.25	1753.00	1778.50	1769.22	172.50	4.42
2	1664.00	1758.00	1779.50	1771.42	153.84	6.46
3	1652.75	1726.50	1768.50	1745.41	168.01	5.61
4	1749.00	1787.50	1813.00	1803.08	155.29	3.09
5	1608.75	1674.00	1686.00	1690.00	153.44	5.05
6	1663.50	1736.00	1758.50	1757.21	168.27	5.63
7	1779.50	1807.00	1827.50	1817.67	166.99	2.14
8	1671.00	1722.50	1740.50	1737.00	168.80	3.95
9	1561.50	1632.50	1663.50	1649.07	155.25	5.61
10	1666.75	1708.50	1739.00	1725.68	154.47	3.54
11	1726.75	1792.50	1816.00	1810.56	155.56	4.85
12	1676.25	1745.00	1756.00	1760.24	168.89	5.01
13	1628.50	1676.00	1703.00	1688.98	154.28	3.71
14	1741.00	1790.00	1814.00	1807.93	167.84	3.84
15	1663.75	1729.00	1761.00	1747.41	165.59	5.03
16	1676.50	1717.00	1749.50	1736.90	167.50	3.60
17	1747.00	1807.00	1824.50	1821.77	155.30	4.28
18	1681.50	1751.00	1779.50	1765.90	154.07	5.02
19	1637.50	1695.00	1718.00	1710.97	166.76	4.49
20	1793.50	1850.00	1874.50	1862.59	169.77	3.85

Note: Diff Ave $= (Ave - f_{LB})/f_{LB} \times 100\%$.

Table 10. Comparison of results under various durations of the scheme.

N	T=inf (∞)	T=900 (u)		T=600 (u)		T=300 (u)	
	Ave (u)	Ave_1 (u)	Diff (%)	Ave_2 (u)	Diff (%)	Ave_3 (u)	Diff (%)
30	961.50	961.50	0	1058.44	10.08	1073.42	11.64
40	1306.00	1421.47	8.84	1435.38	9.91	1440.13	10.27
50	1689.50	1796.51	6.33	1809.29	7.09	1844.60	9.18
75	2699.00	2819.39	4.46	2857.16	5.86	2919.51	8.17
100	3772.50	3874.73	2.71	3950.56	4.72	4054.50	7.48
125	4696.00	4827.55	2.80	4883.37	3.99	5025.19	7.01
150	5915.00	6158.73	4.12	6094.82	3.04	6298.88	6.49
175	7092.50	7242.93	2.12	7304.57	2.99	7394.64	4.26

Note: Diff $= (|\text{Ave}_\alpha - \text{Ave}|)/\text{Ave} \times 100\%$, where $\alpha = \{1, 2, 3\}$.

Table 11. The effect of the volume of the instant requests on the results.

No.	$T(u)$	N_O	β=10%		β=20%		β=30%	
			Y_1 (u)	Diff (%)	Y_2 (u)	Diff (%)	Y_3 (u)	Diff (%)
1	900	30	35.00	3.64	41.50	4.32	49.50	5.15
2	900	50	73.00	4.32	46.00	2.72	74.50	4.41
3	900	100	293.50	7.78	234.00	6.20	425.00	11.27
4	600	30	24.00	2.50	93.50	9.72	94.00	9.78
5	600	50	68.50	4.05	110.50	6.54	65.00	3.85
6	600	100	189.50	5.02	224.00	5.94	350.50	9.29
7	300	30	24.00	2.50	49.00	5.10	70.50	7.33
8	300	50	62.50	3.70	92.00	5.45	38.00	2.25
9	300	100	158.50	4.20	172.00	4.56	167.00	4.43

Note: Y_i indicates the incremental Y of makespan, i.e., the increased value of the updated scheme relative to the planned scheme, $i = \{1, 2, 3\}$; $Diff = Y_i/f_{N_O}$, where f_{N_O} represents the makespan without instant requests.

5.4. Discussion

Here, we present a discussion of the managerial implications for terminals' operations based on the experimental results and our analysis of the experiments.

(1) In the field operations of automated terminals, generating schemes for devices quickly is essential for real-time operations. The algorithm in this paper finds an approximate optimal makespan for 20 tasks within 12 s, and the gap between the makespan and the optimal makespan does not exceed 5%. In addition, experimental results show that the proposed algorithm can still guarantee solution quality with a small range of parameter drift. Therefore, the proposed genetic algorithm based on conflict-avoidance strategy is able to generate the scheme of cranes in a fast and stable manner.

(2) The batch-based task assignment and optimization can overcome the effects of uncertainty in the delivery sequence on the operating system. With uncertainty, generating schemes for a large number of tasks at once does not guarantee that operations will proceed as planned. The approach of batch-based task assignment divides the given task into several batch tasks, and the schemes are generated within each batch task one by one. The experimental results show that when duration of the scheme is smaller (i.e., the more batch tasks divided), the computing time is significantly smaller, and the less affected it is by uncertainty, but the makespan also increases. Therefore, the batching method makes scheduling and operations flexible, but the number of batch tasks should be reasonably divided based on the data of historical operations.

(3) The iterative re-optimization framework and its solution methods provide some theoretical references for dynamic scheduling. Dynamic scheduling is an essential requirement for scheduling in complex and changing environments. If all devices in the operating system perform the planned scheme, the static approach is perfectly capable. The scheme may be suboptimal or even infeasible due to the changes in the operating environment. As a result, re-optimization is necessary. However, the computing resources at terminals will be in short supply if completely re-optimized for all suboptimal schemes. Hence, the iterative re-optimization framework in this paper combines the two patterns (i.e., static and dynamic). Static scheduling is performed if there are no changes in the delivery sequence (i.e., no instant request). Dynamic scheduling is performed as soon as instance requests emerge. The effectiveness of optimization and re-optimization is verified by Exp. 4.

6. Conclusions

This paper proposes a dynamic scheduling problem for crossover yard cranes. For unloading operations, the crane scheduling scheme is usually based on the delivery sequence from the quayside to the yard. In automated container terminals, the loading-first rule is established to ensure the shipping date, which may lead to uncertainty in the delivery sequence for unloading operations. As a result, the planned scheme of cranes may change due to uncertainty, making the scheme become suboptimal or even infeasible. Therefore, we proposed an iterative re-optimization framework to re-optimize the scheme.

In this paper, conflict-avoidance constraints are derived to ensure the orderly operations of the two cranes. Meanwhile, we analyze the effectiveness of the formulated constraints. A mixed-integer program is proposed to solve the problem to minimize the makespan. We divide the given tasks into several batch tasks, and the scheme for each batch task is generated in turn. An iterative re-optimization framework is proposed based on batch-based task assignments and dynamic scheduling processes. Within this framework, a genetic algorithm is designed to generate the planned scheme for the cranes, and a greedy insertion algorithm is proposed to perform the re-optimization. Whenever the instant requests emerged by the delivery sequence change, the framework instantly adds them to a corresponding batch task and re-optimizes the batch task to generate an updated scheme. In addition, a relaxation model is proposed to derive a lower bound for comparison with the makespan by the proposed algorithms.

The experiments show that the average gap between the proposed algorithm and the optimal makespan obtained by the CPLEX solver is only 1.16% in small-scale instances. In large-scale instances, the average gap between the algorithm and the lower bound does not exceed 5%. The greater the degree of uncertainty in the delivery sequence, the greater the incremental increase of makespan compared to the planned scheme. However, this effect gradually weakens as the duration of the scheme decreases. Therefore, the iterative re-optimization framework can meet the requirements of dynamic scheduling, reserving enough time and space for the dynamic optimization of the instant requests. In addition, terminals should collect historical operational data and develop personalized solutions for yard operations to determine an appropriate duration of the scheme based on the workload.

The simultaneous consideration of multiple-device scheduling can optimize the empty load time between devices. In future research directions, the integrated scheduling problem, such as the integrated scheduling of AGVs and ASCs, can be considered to study the representation of dynamic scheduling on the integration problem and its modeling and algorithm design.

Author Contributions: Z.-H.H.: Conceptualization, Methodology. Y.-Z.W.: Formal analysis, Software, Writing- Reviewing and Editing. All authors have read and agreed to the published version of the manuscript.

Funding: National Natural Science Foundation of China: 71871136; the Natural Science Foundation of Shanghai: 23ZR1426500.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study. Written informed consent has been obtained from the patient(s) to publish this paper.

Data Availability Statement: The data that support the findings of this study are available in GitHub at https://github.com/MaritimeYZWang/craneSchedulingModel.git (accessed on 8 March 2023).

Acknowledgments: The authors thank the editors and the anonymous referees for the opportunity of considering this study. This study is partially supported by the Natural Science Foundation of Shanghai (23ZR1426500).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Carlo, H.J.; Vis, I.F.; Roodbergen, K.J. Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. *Eur. J. Oper. Res.* **2014**, *236*, 1–13. [CrossRef]
2. Nossack, J.; Briskorn, D.; Pesch, E. Container Dispatching and Conflict-Free Yard Crane Routing in an Automated Container Terminal. *Transp. Sci.* **2018**, *52*, 1059–1076. [CrossRef]
3. Jiang, X.J.; Jin, J.G. A branch-and-price method for integrated yard crane deployment and container allocation in transshipment yards. *Transp. Res. Part B Methodol.* **2017**, *98*, 62–75. [CrossRef]
4. Kizilay, D.; Van Hentenryck, P.; Eliiyi, D.T. Constraint programming models for integrated container terminal operations. *Eur. J. Oper. Res.* **2020**, *286*, 945–962. [CrossRef]
5. Zhou, C.; Lee, B.K.; Li, H. Integrated optimization on yard crane scheduling and vehicle positioning at container yards. *Transp. Res. Part E Logist. Transp. Rev.* **2020**, *138*, 101966. [CrossRef]
6. Otto, A.; Li, X.; Pesch, E. Two-way bounded dynamic programming approach for operations planning in transshipment yards. *Transp. Sci.* **2017**, *51*, 325–342. [CrossRef]
7. Gans, N.; van Ryzin, G. Dynamic vehicle dispatching: Optimal heavy traffic performance and practical insights. *Oper. Res.* **1999**, *47*, 675–692. [CrossRef]
8. Ozbaygin, G.; Savelsbergh, M. An iterative re-optimization framework for the dynamic vehicle routing problem with roaming delivery locations. *Transp. Res. Part B Methodol.* **2019**, *128*, 207–235. [CrossRef]
9. Wang, Z.; Zhang, J.; Yang, S. An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals. *Swarm Evol. Comput.* **2019**, *51*, 100594. [CrossRef]
10. Xu, B.; Jie, D.; Li, J.; Zhou, Y.; Wang, H.; Fan, H. A Hybrid Dynamic Method for Conflict-Free Integrated Schedule Optimization in U-Shaped Automated Container Terminals. *J. Mar. Sci. Eng.* **2022**, *10*, 1187. [CrossRef]
11. Boysen, N.; Briskorn, D.; Meisel, F. A generalized classification scheme for crane scheduling with interference. *Eur. J. Oper. Res.* **2017**, *258*, 343–357. [CrossRef]

12. Vis, I.F.A.; Carlo, H.J. Sequencing Two Cooperating Automated Stacking Cranes in a Container Terminal. *Transp. Sci.* **2010**, *44*, 169–182. [CrossRef]
13. Gharehgozli, A.H.; Laporte, G.; Yu, Y.; de Koster, R. Scheduling Twin Yard Cranes in a Container Block. *Transp. Sci.* **2015**, *49*, 686–705. [CrossRef]
14. Chen, X.; He, S.; Zhang, Y.; Tong, L.; Shang, P.; Zhou, X. Yard crane and AGV scheduling in automated container terminal: A multi-robot task allocation framework. *Transp. Res. Part C Emerg. Technol.* **2020**, *114*, 241–271. [CrossRef]
15. Hu, Z.-H.; Sheu, J.-B.; Luo, J.X. Sequencing twin automated stacking cranes in a block at automated container terminal. *Transp. Res. Part C Emerg. Technol.* **2016**, *69*, 208–227. [CrossRef]
16. Briskorn, D.; Angeloudis, P. Scheduling co-operating stacking cranes with predetermined container sequences. *Discret. Appl. Math.* **2016**, *201*, 70–85. [CrossRef]
17. Han, X.; Wang, Q.; Huang, J. Scheduling cooperative twin automated stacking cranes in automated container terminals. *Comput. Ind. Eng.* **2019**, *128*, 553–558. [CrossRef]
18. Hsu, H.-P.; Tai, H.-H.; Wang, C.-N.; Chou, C.-C. Scheduling of collaborative operations of yard cranes and yard trucks for export containers using hybrid approaches. *Adv. Eng. Inform.* **2021**, *48*, 101292. [CrossRef]
19. Vis, I.F.; De Koster, R. Transshipment of containers at a container terminal: An overview. *Eur. J. Oper. Res.* **2003**, *147*, 1–16. [CrossRef]
20. Zhang, Q.; Hu, W.; Duan, J.; Qin, J. Cooperative Scheduling of AGV and ASC in Automation Container Terminal Relay Operation Mode. *Math. Probl. Eng.* **2021**, *2021*, 5764012. [CrossRef]
21. Cao, J.X.; Lee, D.H.; Chen, J.H.; Shi, Q. The integrated yard truck and yard crane scheduling problem: Benders' decomposition-based methods. *Transp. Res. Part E Logist. Transp. Rev.* **2010**, *46*, 344–353. [CrossRef]
22. He, J.; Huang, Y.; Yan, W. Yard crane scheduling in a container terminal for the trade-off between efficiency and energy consumption. *Adv. Eng. Inform.* **2015**, *29*, 59–75. [CrossRef]
23. Kim, K.H.; Kim, K.Y. An optimal routing algorithm for a transfer crane in port container terminals. *Transp. Sci.* **1999**, *33*, 17–33. [CrossRef]
24. Li, J.; Yang, J.; Xu, B.; Yin, W.; Yang, Y.; Wu, J.; Zhou, Y.; Shen, Y. A Flexible Scheduling for Twin Yard Cranes at Container Terminals Considering Dynamic Cut-Off Time. *J. Mar. Sci. Eng.* **2022**, *10*, 675. [CrossRef]
25. Liang, C.; Hu, X.; Shi, L.; Fu, H.; Xu, D. Joint dispatch of shipment equipment considering underground container logistics. *Comput. Ind. Eng.* **2022**, *165*, 107874. [CrossRef]
26. Ng, W.C. Crane scheduling in container yards with inter-crane interference. *Eur. J. Oper. Res.* **2005**, *164*, 64–78. [CrossRef]
27. Park, T.; Choe, R.; Ok, S.M.; Ryu, K.R. Real-time scheduling for twin RMGs in an automated container yard. *OR Spectr.* **2010**, *32*, 593–615. [CrossRef]
28. Zhuang, Z.; Zhang, Z.; Teng, H.; Qin, W.; Fang, H. Optimization for integrated scheduling of intelligent handling equipment with bidirectional flows and limited buffers at automated container terminals. *Comput. Oper. Res.* **2022**, *145*, 105863. [CrossRef]
29. Huisman, D.; Freling, R.; Wagelmans, A.P.M. A robust solution approach to the dynamic vehicle scheduling problem. *Transp. Sci.* **2004**, *38*, 447–458. [CrossRef]
30. Barrena, E.; Canca, D.; Coelho, L.C.; Laporte, G. Single-line rail rapid transit timetabling under dynamic passenger demand. *Transp. Res. Part B Methodol.* **2014**, *70*, 134–150. [CrossRef]
31. Chong, L.; Osorio, C. A simulation-based optimization algorithm for dynamic large-scale urban transportation problems. *Transp. Sci.* **2018**, *52*, 637–656. [CrossRef]
32. Fan, H.; Xiong, H.; Goh, M. Genetic programming-based hyper-heuristic approach for solving dynamic job shop scheduling problem with extended technical precedence constraints. *Comput. Oper. Res.* **2021**, *134*, 105401. [CrossRef]
33. Guo, X.; Huang, S.Y. Dynamic space and time partitioning for yard crane workload management in container terminals. *Transp. Sci.* **2012**, *46*, 134–148. [CrossRef]
34. Potthoff, D.; Huisman, D.; Desaulniers, G. Column generation with dynamic duty selection for railway crew rescheduling. *Transp. Sci.* **2010**, *44*, 493–505. [CrossRef]
35. Saddoune, M.; Desaulniers, G.; Elhallaoui, I.; Soumis, F. Integrated airline crew pairing and crew assignment by dynamic constraint aggregation. *Transp. Sci.* **2012**, *46*, 39–55. [CrossRef]
36. Pillac, V.; Gendreau, M.; Guéret, C.; Medaglia, A.L. A review of dynamic vehicle routing problems. *Eur. J. Oper. Res.* **2013**, *225*, 1–11. [CrossRef]
37. Yin, J.; Yang, L.; Tang, T.; Gao, Z.; Ran, B. Dynamic passenger demand oriented metro train scheduling with energy-efficiency and waiting time minimization: Mixed-integer linear programming approaches. *Transp. Res. Part B Methodol.* **2017**, *97*, 182–213. [CrossRef]
38. Li, S.; Liu, R.; Yang, L.; Gao, Z. Robust dynamic bus controls considering delay disturbances and passenger demand uncertainty. *Transp. Res. Part B Methodol.* **2019**, *123*, 88–109. [CrossRef]
39. Lei, C.; Jiang, Z.; Ouyang, Y. Path-based dynamic pricing for vehicle allocation in ridesharing systems with fully compliant drivers. *Transp. Res. Part B Methodol.* **2019**, *132*, 60–75. [CrossRef]
40. Luo, S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Appl. Soft Comput.* **2020**, *91*, 106208. [CrossRef]

41. Scherr, Y.O.; Hewitt, M.; Saavedra, B.A.N.; Mattfeld, D.C. Dynamic discretization discovery for the service network design problem with mixed autonomous fleets. *Transp. Res. Part B Methodol.* **2020**, *141*, 164–195. [CrossRef]
42. Ulmer, M.W.; Goodson, J.C.; Mattfeld, D.C.; Thomas, B.W. On modeling stochastic dynamic vehicle routing problems. *EURO J. Transp. Logist.* **2020**, *9*, 100008. [CrossRef]
43. Wang, L.; Hu, X.; Wang, Y.; Xu, S.; Ma, S.; Yang, K.; Liu, Z.; Wang, W. Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning. *Comput. Netw.* **2021**, *190*, 107969. [CrossRef]
44. Li, Y.; Gu, W.; Yuan, M.; Tang, Y. Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network. *Robot. Comput. Manuf.* **2022**, *74*, 102283. [CrossRef]
45. Vis, I.F.A.; Roodbergen, K.J. Scheduling of Container Storage and Retrieval. *Oper. Res.* **2009**, *57*, 456–467. [CrossRef]