applied sciences

MDPI

Article

Real-Time Batch Optimization for the Stochastic Container Relocation Problem

Sifang Zhou [1,2,*] and Qingnian Zhang [1]

[1] School of Transportation and Logistics Engineering, Wuhan University of Technology, Wuhan 430070, China; zqnwhut@163.com
[2] School of Information Technology, Shangqiu Normal University, Shangqiu 476000, China
* Correspondence: sf.zhou@whut.edu.cn

Abstract: The container relocation problem (CRP) is an important factor affecting the operation efficiency of container terminal yards, and it has attracted much attention for decades. The CRP during the pickup operations of import containers is still an intractable problem for two reasons: the first is that the solution efficiency of the algorithms developed in the existing literature cannot meet the real-time operation requirements; the second is that the pre-optimized operation plan cannot cope with the changes in the real-time operation scenarios caused by the uncertainty of the arrival time of external trucks. This paper proposes an optimization method for the real-time operation scenario which aims to solve the most reasonable operation plan quickly according to the arrivals of external trucks, in which a dynamic upper bound of the optimal solution is derived based on the dynamic programming model of the import containers' CRP, and an approximate optimal solution can be obtained by minimizing this dynamic upper bound. A heuristic algorithm based on three relocation rules is developed to implement this method, considering the adjustment of the pickup sequence of the target containers. Numerical experiments show that (1) when the number of a batch of target containers is less than 10 (excluding target containers that can be directly picked up), the method proposed in this paper can solve the problem quickly to meet the demand of optimizing real-time pickup operations; (2) compared with other outstanding algorithms, the quality of the solutions obtained by this method is also improved; and (3) this method can be applied to the most container terminals for optimizing real-time pickup operations.

Keywords: containers terminals; import containers; container relocation problem; real-time optimization; heuristics

Citation: Zhou, S.; Zhang, Q. Real-Time Batch Optimization for the Stochastic Container Relocation Problem. *Appl. Sci.* **2024**, *14*, 2624. https://doi.org/10.3390/app14062624

Academic Editors: Dusica Marijan and Paolo Renna

Received: 6 January 2024
Revised: 17 March 2024
Accepted: 18 March 2024
Published: 21 March 2024

1. Introduction

Generally, in the yard of the container terminal containers are stacked on top of each other in the ground slots, forming stacks. At the same time, a row comprises several stacks placed abreast, constituting a bay, and each yard block is composed of dozens of bays. The containers in a stack adhere to the last-in-first-out (LIFO) principle. If a target container that needs to be retrieved is not the topmost in its stack and is covered by other containers, the blocking containers must be moved to another stack in the same bay first. As a result, the yard crane is required to perform one or more relocation movements. Moving a blocking container, referred to as relocation (rehandling or reshuffling), is an unproductive operation. A high proportion of unproductive operations resulting from container rehandling decreases yard operational efficiency, delays the turnaround time of external trucks, and increases the detention time of vessels. Regarding relocations, three topics have been identified: the re-marshaling problem, the pre-marshaling problem, and the relocation problem. The relocation problem is referred to as the container relocation problem (CRP) or the block relocation problem (BRP) in the literature and is defined as needing to retrieve all the containers (export or import) from a bay in the prescribed order

while minimizing the number of rehandles. Despite the fact that CRP has attracted much attention for decades, the CRP during the pickup of import containers is still an intractable problem, which is the focus of this paper.

The difficulty in dealing with the CRP of import containers is not only due to the fact that it is an NP-hard problem [1] but also to the uncertainty of the pickup sequence and the decentralization of the retrieval operations. In response to the first point, the NP-hard nature, it is necessary to find more efficient methods for solving this problem. Various existing models or algorithms are based on the idea of reducing the problem scale to improve the solution efficiency, whether it is to use a solver to solve an integer programming model, to solve based on a tree search algorithm, or to use heuristic methods. However, the CRP aims to find the optimal operation plan for retrieving all the containers in a bay while minimizing the number of relocations during the pickup operations. The computational effort of the solution process increases rapidly as the instance size increases. Many methods do not perform well in solving the CRP, e.g., restricted CRP is still an NP-hard problem, and the B&B algorithm and other variants of the tree search algorithm are not easy to solve medium-sized and large-sized instances quickly. To improve the efficiency of solving the CRP, we need to consider some ways to reduce the problem size further. To be precise, it is difficult to accurately master the retrieval order of containers in advance since the arrival time of trucks is random [2]. But, the truck appointment system (TAS), which is implemented in most container terminals, can provide container terminals with more credible retrieval information and help terminals optimize yard operations [3,4]. On this basis, the stochastic container relocation problem (SCRP) [5], also called a CRP with time windows [6], has been the focus of research related to the import container relocation problem. In the SCRP, the containers are divided into different batches (groups) according to their appointment departure time window, i.e., a batch of containers indicates a group of containers stacked in the same bay and with the same appointment retrieval time window. Containers are ordered so that all containers in a batch must be retrieved before any containers from a later batch. The pickup order of the containers within the same batch is uncertain. This suggests that their pickup sequence needs to be determined based on the arrival of corresponding external trucks (ETs).

Due to the decentralization of the retrieval operation of import containers, the operation process for retrieving all the import containers in the bay requires multiple independent operation rounds (also referred to as "operation stages" in other studies). The pickup order and operation plan are determined within each operation round based on the ETs' arrival information. Between two neighboring operation rounds, the bay layout at the end of the previous operation round will be the basis for the next round. Therefore, in this paper, we propose a method that gradually optimizes the pickup operations during each operation round instead of optimizing the overall operation process for retrieving all the containers in a bay. In this method, the problem size of the SCRP is significantly reduced.

This method is referred to as *Real-time Batch Optimization*, which focuses on the real-time operation round (or the current operation round). The "real-time" means that the current operation round is subject to the instant generation of an operation plan based on the arrival of ETs rather than execution based on a predetermined operation plan. The real-time operation scenario has three characteristics: first, a batch of ETs has arrived at the yard; second, a batch of target containers is known, and the number of target containers depends on the arrived ETs; third, the pickup sequence of ETs is adjustable. In the new approach, we first categorize the ETs into batches (groups) based on their reservation information from the TAS and the arrival information from terminal gates. Of course, the retrieval sequence of the ETs belonging to different batches is precise, and the retrieval sequence of the ETs within the same batch is still being determined. Then, based on this grouping, all the arrived ETs within the real-time operation round are defined as a specific batch (group) whose retrieval priories are set to zero, representing the highest pickup priority. Correspondingly, the target containers within the real-time operation round are a batch with a retrieval priority of 0, which may only be part of the containers booked in the

current time window. Moreover, their pickup service order of all arrived ETs is adjustable (no longer following the first-come-first-served principle), and the optimal pickup order is determined based on the bay layout to minimize the number of relocations.

The contributions of this paper are as follows:

- Based on the dynamic programming (DP) model of the SCRP, we derive a dynamic upper bound of its optimal solution. Thus, we transform the optimization objective into minimizing this dynamic upper bound, significantly reducing the size of the solution process.
- We propose a new heuristic relocation strategy, namely Least-relocations-and-Lowest-precedence-gap (LL), which learns from the expected relocation index (ERI) heuristic presented by [6] and the expected min-max (EM) heuristic proposed by [5].
- We introduce two novel relocation rules: moving ahead for a sequential-placed slot (MSS) and freeing up for a sequential-placed slot (FSS). The former rule assigns, as far as possible, a sequential placement for the potential blocking containers in the future in advance. The latter rule tries to find a stack that can free its topmost container to satisfy a sequential placement for the current blocking container.
- A novel heuristic algorithm, called the sequential-placement first heuristic ($SPFH$), is devised. It takes advantage of the properties of the two relocation rules mentioned above.

The rest of the paper is structured as follows: In Section 2, we quickly survey the recent literature on the CRP. Section 3 gives a detailed description of the real-time CRP and the formulation of the improved DP model. Section 4 describes the proposed $SPFH$ algorithm in detail. We report the results of our computational experiments with the methods discussed in this paper in Section 5. Finally, Section 6 concludes the study and suggests future research directions.

2. Literature Review

Two fields have been extensively investigated in the existing literature to reduce relocations during the retrieval process of import containers. In the first field, the objective is to allocate stacking locations for new unloaded containers to reduce relocation moves in the future. Such as, the formulations for evaluating the expected number of rehandles were proposed in [7,8] based on the segregation storage strategies. Other similar studies on generating container storage strategies can be found in [9–11]. The second field, the container relocation problem, focuses on the relocation movements during the retrieval process and aims to retrieve all containers from a bay based on the prescribed order while minimizing relocations.

The CRP has received much attention from scholars for decades, and extensive studies from multiple perspectives have been conducted [12,13]. In order to obtain the optimal solution of CRP, many papers have formulated integer programming (IP) to find the optimal solution [4,14–18]. Because the number of variables and constraints in IP exponentially increases as the size of the problem grows, the IP model cannot be solved for large instances. Another method to obtain the optimal solution of CRP is to utilize the exact algorithms. Many exact algorithms are developed based on tree search procedures, such as tree search algorithms [19,20], (iterative deepening) A* algorithms [21,22], B&B algorithms [18,23–28], and abstract approaches [6].

However, the CRP is an NP-hard problem [29]. For larger-scale instances, the exact algorithms or IP models are time-consuming and impractical for real-time operation demands. Therefore, researchers have tackled this challenge from two viewpoints. First, Caserta et al. [29] proposed an assumption that a container can only be relocated if it is blocking the target container. This assumption reduces the number of movable containers during pickup operations, thereby reducing the size of the CRP problem. The CRP under this assumption is referred to as restricted CRP. The restricted CRP has become an important branch of the CRP research field [17,18,20,27,29]. Compared to unrestricted CRP, restricted CRP does save a small amount of computational time, but at the cost of losing the

quality of the solution. It is worth noting that the restricted CRP is still an NP-hard problem. Another approach is to utilize the heuristics to solve the CRP model. Researchers have been increasingly exploring heuristic algorithms to overcome the computational complexity of CRP. To date, many outstanding heuristic algorithms have been developed, such as beam search algorithms [25,26,30,31], greedy heuristics [14,32–34], and other heuristics based on relocation rules [5,6,35,36].

Due to the difficulty of obtaining the retrieval time information in advance, many models of CRP were constructed based on assuming that the retrieval processes of import containers conform to some probability distribution [37,38]. Moreover, Zehendner et al. [35] proposed the online CRP models where the retrieval orders of the containers are revealed progressively with the arrival of ETs. Similar studies can be found in [6,39]. Truck appointment systems (TAS) can help terminals address this issue. The TAS has been deployed in most container terminals, which allows terminal operators to obtain the containers' retrieval time window. Based on the appointment information, Galle et al. [5], Ku and Arthanari [6] introduced a batch model in which the containers within the same appointment time window are considered a batch. Based on the batch model, Galle et al. [5] proposed the stochastic container relocation problem (SCRP) where the retrieval order between different batches is clear but uncertain within the same batch. It means the containers within the same batch have the same retrieval probability. Indeed, the retrieval sequence of the same batch containers is indistinguishable until associated ETs arrive.

For the CRP, relocating operations occur when the container pickup orders do not match their stacking sequence. As a manager, the terminal should not passively deal with the pickup sequence according to the ETs' arrival order but should proactively coordinate the ETs' pickup sequence to conform to the containers' stacking sequence to minimize the number of rehandles [4]. Zhao and Goodchild [39] first proposed an embryo of the Batch Optimization method, in which a batch of ETs has arrived, and their pickup service orders are adjustable. Nevertheless, they only apply this method to the first batch to avoid prolonging the other ETs' waiting time. Zeng et al. [3] proposed a rehandling model that considered adjustments to the container pickup order within each group to minimize the number of rehandles. They grouped the ETs according to the appointment time slots in the TAS. Feng et al. [2] proposed a flexible service policy for the SCRP, which assumes all the ETs corresponding to the containers of each batch arrived in advance and dictates the retrieval orders of the current batch's import containers to minimize the number of relocations.

The novelty of this work. A review of the literature indicates that few studies have focused only on optimizing the real-time operation scenario during import container pickup operations rather than the whole retrieval process. Considering the importance of pickup operations' efficiency to the yard's efficiency, it is necessary to find a practical and effective way to solve this problem. This paper attempts to fill this gap by focusing only on the optimization of the real-time operation round for a batch of target containers. Zeng et al. [3] and Feng et al. [2] are closer to the optimization objective of this paper. They assume the simultaneous arrival of ETs within the same reservation time windows and allow the ETs' retrieval sequence to be adjustable, aiming to optimize the pickup operation process for all target containers. However, their approach does not really reduce the problem size, and it is still difficult to solve the larger instances. In this paper, we only optimize one operation round, a part of the whole retrieval process of import containers, i.e., the real-time operation round, which effectively reduces the scale of the solution process, which allows for the adjustment of the pickup sequence and iterates all the available pickup sequences and effectively improves the solution quality.

3. Problem Description and Formulation

Considering the CRP from a practical viewpoint, instead of passively formulating an optimization plan based on the reservation information from TAS or the assumed probability distribution of ETs' arrivals, we should strengthen the terminal's management of the real-time operation process. In this paper, we optimize the overall pickup operations

of the target containers according to the actual arrivals of the ETs (affected by various uncertainties, the ETs may not all arrive according to their booking time), and the pickup orders of ETs is adjustable (which is true in practical operations). Therefore, we iterate over all feasible pickup sequences to find the operation plan with the minimal number of relocations. This section explains the modeling framework of the proposed DSS and the mathematical formulation of the proposed bi-objective optimization model and introduces a comprehensive numerical example.

3.1. Problem Description

Due to the limited number of yard cranes, each must move back and forth between different bays rather than being dedicated to a specific unit. So, the state change of a bay and the division of operation rounds during the retrieval process of import containers are shown in Figure 1. As shown in Figure 1a, we define an *operation round* as the bay is in *busy* state, whose period that begins when the yard crane moves into the destination bay and ends when the yard crane moves out the destination bay. After an operation round is completed, the bay state turns into *idle*. Figure 1b tells us that the whole retrieval process of a bay can be divided into three parts—completed operation rounds, the real-time operation round and future operation rounds. The completed operation rounds represent that the retrieval and relocation operations have been finished, which is irreversible. The operations within the real-time operation round are unrelated to the completed operation round but will impact future operations.

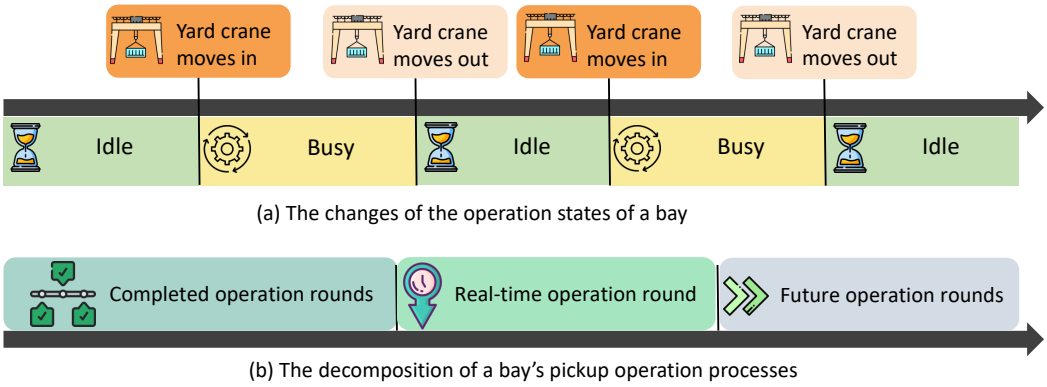


Figure 1. The retrieval operation process analysis of a bay.

Consider a bay with N containers piled on W stacks, and the height of each stack cannot exceed H tiers. In the initial state, all the N containers in the bay have been appointed their departure time windows according to the corresponding ETs' reservation arrival period. In turn, they can be classified into G groups based on their respective booking departure times and numbered $1, \ldots, G$, which implied their retrieval priorities (the smaller group index, the higher retrieval priority).

Suppose this instance requires K ($K \geq 1$) operation rounds to finish retrieving all the N containers. Let S^k denote the bay layout after the completion of kth operation round, especially indicating the initial layout when $k = 0$. Suppose we have completed the first $k-1$ operational rounds, then we need to complete the kth operation round. We aim to find an optimized solution a^k for the kth operation round that minimizes the number of relocations during the kth operation round. Once the kth operation round finishes, the bay layout will be transformed from S^{k-1} to S^k, which becomes the basis for $(k+1)$th operation round. The change in the bay layout between two adjacent operation rounds is described by the following Equation (1).

$$S^{k-1} \xrightarrow{a^k} S^k, \ 1 \leq k \leq K \tag{1}$$

The a^k is a set of operation instructions. If a^k includes some relocation operation instructions, some containers will be moved from their stacks into other new stacks. For the restricted CRP, the relocated containers are only the blocking containers above the targets. The relocated containers may contain containers other than the blocking containers above the targets for the unrestricted CRP. However, these movements may result in the addition of the number of blocking containers for the moved-in stacks and impact the $(k+1)$th and subsequent operation rounds. Our objective is to minimize the addition of the number of blocking containers in the real-time operation round.

Moreover, during real-time operations, the retrieval sequence of target containers and the movement orders of blocking containers determine the changes in the bay layout, and different bay layouts will produce different increments in the number of blocking containers and other impacts on the subsequent operation process. In the real-time operation round, when multiple target containers need to be picked up, it makes sense to choose a reasonable target container pickup order that can produce fewer relocation operations, even though it may prolong the waiting time of some ETs. Therefore, the method proposed in this paper allows for adjusting the pickup order according to the bay layout, integrating the optimization of the pickup sequence and the reallocations of blocking containers to minimize the increment of the expected number of blocking containers (ENBC). No doubt, the pickup order in practice is likely to be inconsistent with the ETs' arrival sequences, which will result in loading operations being advanced for some ETs and delayed for others. In practice, the number of waiting ETs for retrievals is usually small, and as the number of relocations decreases, the whole operation time decreases, and the delays incurred by individual ETs are acceptable.

3.2. Formulation for the Real-Time Scenario of CRP

Learning from [6], we first present a simplified DP model of real-time CRP. Let $R(S^k)$ denote the minimum number of relocations required to remove all the remaining containers based on the layout S^k, and let $q(a^k|S^{k-1})$ denote the number of realized operations that occur during the kth operation round. The DP model can be formulated as a recursive function as in Equation (2).

$$\begin{cases} R(S^{k-1}) = \min\limits_{a^k} \left\{ q(a^k|S^{k-1}) + R(S^k) \right\}, \ 1 \le k \le K \\ R(S^K) = 0 \end{cases} \tag{2}$$

Another equivalent form of Equation (2) is shown in Equation (3)

$$R(S^{k-1}) = \min_{a^k} \left\{ q(a^k|S^{k-1}) + \min_{a^{k+1}} \left\{ q(a^{k+1}|S^k) + \ldots + \min_{a^K} \left\{ q(a^K|S^{K-1}) + \right\} \ldots \right\} \right\} \tag{3}$$

From Equation (3), to obtain the optimal operation plan a^k for the kth operation round, the solution process requires simultaneous evaluation of the optimal operation plans from the $(k+1)$th operation round to the last operation round. So, solving the total optimization objective $R(S^0)$, which is the minimum number of relocations for picking up all containers during the whole retrieval process, is difficult because the complete solving process will generate a large number of redundant computation efforts. Consider it another way, and we can try to find an approximate optimal solution for it.

According to Equation (2), if a^* is the theoretically optimal operation plan for the kth round, then we have $R(S^{k-1}) = q(a^*|S^{k-1}) + R(S^k)$. Since solving for the theoretical optimum is difficult, assuming that we obtain an available solution a^k for the kth operation round, we can derive an inequality Equation (4).

$$R(S^{k-1}) \le q(a^k|S^{k-1}) + R(S^k) \tag{4}$$

Without loss of generality, if $a^1, a^2, \ldots, a^K$ are available solutions for each round, respectively, then we must have inequality Equation (5).

$$R(S^0) \leq \sum_{k=1}^{K} q(a^k|S^{k-1}) \tag{5}$$

In this way, we convert the solving process of Equation (3) to summing the number of practical relocations during each operation round. We can ensure an approximate optimal solution if we minimize the number of relocations within each round and the impact on the subsequent operation rounds.

According to the definition, $q(a^k|S^{k-1})$ includes two aspects: forced relocations and relocations to free up storage placements. The forced relocations result from blocking the target containers during the real-time operation round. The relocations to free up storage placements result from the containers having nothing to do with the target containers and aim to free up good placements for the blocking containers. In addition, some blocking containers may be relocated twice or more due to all candidate storage positions containing the target containers, but this case is included in the forced relocations. Therefore, we can derive the following inequation as Equation (6).

$$R(S^0) \leq \sum_{i=1}^{K} \left(\left| \bigcup_{c \in P_k} B_c^k \right| + |U_k| \right) \tag{6}$$

B_c^k denotes the blocking containers of the target container c in kth operation round, and U_k means the set of relocated containers unrelated to the target containers during the kth operation round.

Let $LB(S^k)$ denote the number of blocking containers in the layout S^k, which will require rehandling operations in the remaining rounds. P_k indicates the set of target containers in kth operation round. During the kth operation round, each container $b \in B_c^k$ must be moved into other stacks and may become new blocking containers in the move-in stack. Let $f(b, s)$ denote the expected additional number of the blocking containers when the container b is moved into the stack s. Its formula will be given later. Thus, after completing the kth operation round, the number of blocking containers in the layout S^k is as Equation (7).

$$LB(S^k) = LB(S^{k-1}) - \left| \bigcup_{c \in P_k} B_c^k \right| + \sum_{s=1}^{W} \sum_{b \in \mathcal{B}} f(b, s),\ \mathcal{B} = \left\{ \bigcup_{c \in P_k} B_c^k \right\} \cup U_k \tag{7}$$

When we have removed all the containers, the last bay layout S^K is empty and $LB(S^K) = 0$. The iterative summation of Equation (7) for k from 1 to K yields Equation (8).

$$\sum_{k=1}^{K} \left| \bigcup_{c \in P_k} B_c^k \right| = LB(S^0) + \sum_{k=1}^{K} \sum_{s=1}^{W} \sum_{b \in \mathcal{B}} f(b, s),\ \mathcal{B} = \left\{ \bigcup_{c \in P_k} B_c^k \right\} \cup U_k \tag{8}$$

The total optimizing objective of this study is min $R(S^0)$. It satisfies inequality Equation (9).

$$R(S^0) \leq LB(S^0) + \sum_{k=1}^{K} \sum_{s=1}^{W} \sum_{b \in \mathcal{B}} f(b, s) + \sum_{k=1}^{K} |U_k|,\ \mathcal{B} = \left\{ \bigcup_{c \in P_k} B_c^k \right\} \cup U_k \tag{9}$$

It is well known that $LB(S^0)$ is the minimal lower bound of $R(S^0)$. According to Equation (9), we derive an upper bound of $R(S^0)$. An approximate optimal solution to the SCRP can be obtained by minimizing the upper bound. Therefore, we concentrate on optimizing the real-time operation process of each round and decrease the computational efforts for assessing the influence of relocating obstructive containers within each operation

round on subsequent operation rounds. This method will significantly enhance the solution efficiency of the SCRP and make it suitable for real-time scenarios.

4. Methodology

According to Equation (9), the real-time scenario of the SCRP focuses on two aspects: (1) minimizing the instant relocation number during the real-time operation round; (2) minimizing the incremental number of the blocking containers after each operation round.

4.1. Preliminaries

Before describing the implementation of the algorithms in detail, we first define the evaluation method of ENBC and then introduce three heuristic relocation rules.

4.1.1. The Evaluation Method of ENBC

Let $p(c)$ represent the retrieval priority value of container c and $p_{min}(s)$ denote the minimum retrieval priority value of the container set s (if s is a stack number, it represent all the containers in stack s), the smaller value is the the higher priority. Specifically, $p_{min}(s) = N + 1$ if s is an empty set or stack. Let $under(c)$ denote the containers set within the same stack with container c and below it.

We define three placement state types of containers as follows. The placement state of the container c is to be known as: (1) *sequential-placement* if $p(c) < p_{min}(under(c))$ is satisfied; (2) *inverted-placement* if $p(c) > p_{min}(under(c))$ is satisfied; (3) *level-placement* if $p(c) = p_{min}(under(c))$ is *True*. A sequential-placement container means that all other containers placed below have lower retrieval priorities than it. Similarly, an inverted-placement container means that all other containers placed below have higher retrieval priorities than it, i.e., it has a bigger retrieval priority value than the other containers below it. A level-placement container has the same retrieval priority as the highest retrieval priority of the other containers placed below.

Based on their predetermined precedence (e.g., appointment time window), the containers with higher precedence will be retrieved earlier than those with lower precedence during retrievals. Therefore, a sequential-placement container will be picked up earlier than other containers below it without relocation. Conversely, an inverted-placement container must be relocated at least once, as some containers below it will be picked up earlier. We assume the containers with the same retrieval priority have the same retrieval probability. So, a level-placement container causes a relocation when the same retrieval priority container below it is picked up earlier. The blocking containers in a bay layout include the inverted-placement and level-placement containers, and the ENBC is equal to the sum of the number of inverted-placement containers and the expected relocation number of the level-placement containers. The calculation method for the ENBC can be described as Equation (10), which has been investigated in [5]. In Equation (10), $I(x)$ is an indicator function that equals one if x is *True* and zero otherwise, and $h(r)$ indicates the container number within stack r.

$$lb(r) = h(r) - \sum_{i=1}^{h(r)} \left(\frac{I\left(p(c_i) = \min\limits_{1 \le j \le i} \{p(c_j)\} \right)}{\sum\limits_{j=1}^{i} I(p(c_i) = p(c_j))} \right) \tag{10}$$

There is no increase in the number of blocking containers in a candidate stack when a blocking container becomes a sequential-placement container after being moved into the candidate stack during retrieval. We refer to the candidate stacks for the blocking container that satisfy the above condition as *sequential-stack* (*SS* for short). Contrarily, the ENBC of the candidate stack will increase when a blocking container becomes the inverted-placement or level-placement container after being moved into a candidate stack. The candidate stack

that satisfies the condition is referred to as *inverted-stack* (*IS* for short) or *level-stack* (*LS* for short), respectively. The notation $f(c,r)$ has been defined previously, which denotes the increment of ENBC of the stack r after relocating the container c into stack r, and the formula is shown Equation (11).

$$f(c,r) = \begin{cases} 0, & r \in SSs, \\ 1, & r \in ISs, \\ \frac{m-1}{m}, & r \in LSs, \end{cases} \tag{11}$$

where $m = \sum_{j=1}^{h(r)} I(p(c) = p(c_j))$ denotes the number of containers with the same priority as container c in the stack r.

4.1.2. Heuristic Relocation Rules

We introduce three relocation rules in this subsection. The first relocation rule, namely the Least-relocations-and-Lowest-precedence-gap (LL) rule, draws inspiration from relocation rules such as ERI [6], Minmax [29], and EM [5]. The other two relocation rules are used to compensate for the deficiencies of the LL rule, including the prospective re-allocations for some non-instant-relocated blocking containers to produce a better layout.

LL Rule

Two conditions construct the LL rule. They are first, selecting the set of stacks that produces the smallest increment of the ENBC after reshuffles, and, second, choosing the move-in stack from the set of stacks obtained in the "first step" with the closest priority value for the blocking container. Equation (12) describes the mathematical formula of the LL rule.

$$s^* = \arg\min_{s} \{|p_{\min}(s) - p(c)|\},\ s \in \{r|\min f(c,r)\},\ s \neq s(c) \tag{12}$$

$s(c)$ denotes the function to obtain the stack index where the container c is stacked. If $f(c,s^*) = 0$, it indicates that the topmost slot of stack s^* is a sequential stack for blocking container c. Therefore, moving container c to stack s^* does not increase the ENBC. If $f(c,s^*) \neq 0$, it indicates that there is no sequential stack for blocking container c to move in. The LL rule provides a method for selecting the move-in stack for container c, which suggests that the stack with the closest precedence to container c is the best choice. This rule has the advantage of keeping the stack with lower priority to provide sequential stacks for subsequent blocking containers as much as possible. Figure 2 illustrates the processing of heuristic rule *LL*. In Figure 2, we set the priority of the target container to 0 and presented the decision process to relocate the blocking containers by LL. The container priority value marked by the red circle is the current blocking container. Numbers under the layout represent the minimum priority of each assignable stack and the increment of ENBC if the current blocking container is moved into this stack. The blue number marks the optimal selection. The tag "$(1,3)$" represents the container placed in the first stack and 3rd tier. The tag "$\langle(1,3),1,4\rangle$" denotes an instruction, meaning the container "(1,3)" will be moved from the first stack into the fourth one. (The symbols illustrated in Figure 2 have the same meaning in the subsequent figures.)

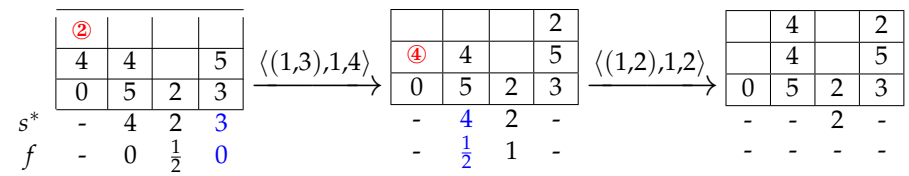


Figure 2. Processing of heuristic rule *LL*.

The *LL* rule has two potential areas for improvement. Firstly, it is a greedy approach that always assigns a sequential placement for the relocating container whenever possible, which will likely trap into the local optimum. In other words, reallocating a blocking container with higher retrieval priority to a sequential placement with lower priority may result in losing sequential-placed slots for subsequent relocations. Secondly, when only inverted-placement slots are available for the blocking container, the subsequent blocking containers likely have no more sequential placements to select from, which will directly make additional relocations or result in more potential relocations in the following retrieval processes. The next two relocation rules remedy the deficiencies of the *LL* rule in the previously mentioned aspects.

Moving Ahead for Sequential Stack (MSS)

Given the blocking container c that is next to be relocated, its optimal reallocation stack s^* is determined according to the *LL* rule. If stack s^* is sequential-stack for blocking container c, we try to find another inverted-placement container, which is the topmost item in its stack s ($s \neq s^*$) and denotes as $top(s)$. If stack s^* is sequential-stack for container $top(s)$ and moving it into stack s^* before container c does not change the sequential-placement state of container c. We define this method as a relocating policy, known as the Moving-ahead for Sequential Stack (MSS) rule. The MSS rule aims to move a inverted-placement container into a sequential-stack stack in advance and reduce the likelihood of the container $top(s)$ becoming a blocking container again in the subsequent operation process. This rule needs to comply with the following conditions: (1) there are at least two empty slots in stack s^*; (2) the topmost container $top(s)$ in stack s must be in inverted-placement state, i.e., its retrieval priority satisfies the condition $p(top(s)) > p_{min}(s)$; and (3) the retrieval priority of container $top(s)$ must satisfy the condition $p(c) < p(top(s)) < p_{min}(s^*)$, i.e., the stack s^* is sequential-stack for containers c and $top(s)$ and the retrieval priority of container $top(s)$ is lower than container c. Figure 3 shows us the processing of the rule *MSS*.

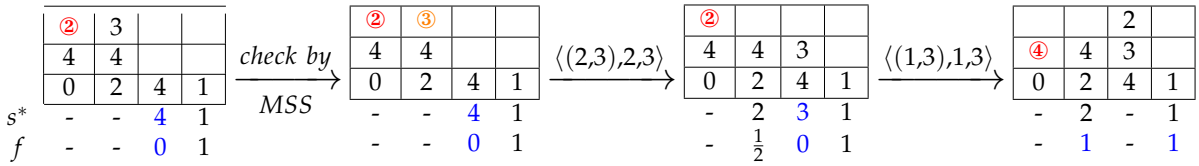


Figure 3. Processing of heuristic rule *MSS*.

Freeing up for Sequential Stack (FSS)

For a blocking container c, if only inverted-stack stacks are available, directly moving container c into one of the stacks will add a new blocking container and, in turn, add a relocation during the subsequent retrieval processes. In this case, selecting a suitable stack s and moving its topmost container $top(s)$ to another stack s' will free up an sequential-placement for container c. If stack s' is a sequential-stack for container $top(s)$, the total operation cost of the "free up" operations is the same as that of direct moving. As well as it may also increase the possibility of selecting sequential-placement for the subsequent relocations. We refer to the above method as a new relocation policy: freeing up for the sequential stack (FSS) rule. The FSS rule needs to comply with the following conditions: (1) the available reallocation positions for the blocking container c are inverted-placements; (2) if stack s is a candidate sequential-placement for the blocking container c by freeing up the topmost item $c^{'}$, while the retrieval priority value satisfies $p(c^{'}) < p(c) < p(under(c^{'}))$; and (3) the container $c^{'}$ will be moved into stack $s^{'}$, which must be the sequential-stack for container $c^{'}$.

We now continue to analyze the instance depicted in Figure 3. In the final configuration, container $(1,2)$ is the current forced relocating item, and two available stacks, the second and fourth stack, are both inverted stacks. Let us move container $(4,1)$ from the fourth

to the second stack and obtain a sequential stack by consuming one practical relocation movement. Furthermore, this reshuffle is cost-effective since it may provide an additional sequential-placement in subsequent processes. Figure 4 illustrates the detailed processing of the FSS rule.

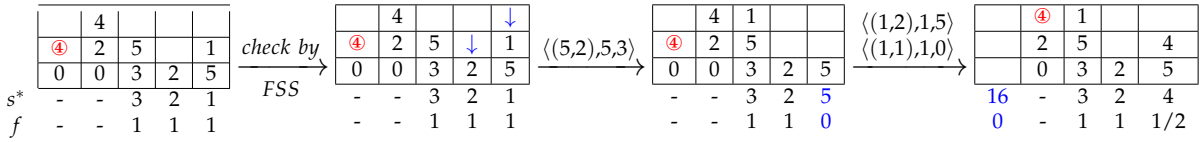


Figure 4. Processing of heuristic rule *FSS*.

4.2. Sequential-Placement First Heuristic (SPFH)

This section presents the *SPFH* algorithm for the real-time operation round of SCRP. The algorithm structure of *SPFH* includes two levels. The outer level adopts a policy iteration mechanism, i.e., the set of policies comprising all possible pickup sequences, from which the best operation solution has the least cost; the inner level is the specific process of simulating the operation solution and evaluating the cost for each pickup sequence.

The outer level of *SPFH* takes two inputs: S_0 denotes the initial configuration of the current operation round of the bay, and P denotes the target container set of the current operation round. In the outer level of *SPFH*, the procedure, namely AutoRetrieval, is a conventional prioritization mechanism that checks and automatically picks up the target containers if retrievable before the start of each operation round or after each relocation. This mechanism makes the configuration irreducible while freeing up optional slots for other blocking containers. The procedure, namely GenerateSequence, generates all possible pickup orders for the target containers of the current operation round. Then, iterate these orders to obtain the best operation plan for the yard cranes with minimal operation costs. The procedure, namely Simulation-Evaluation, is the inner level of the *SPFH* algorithm. The pseudo-code of the outer level of *SPFH* is shown in Algorithm 1, and the pseudo-codes of the procedure AutoRetrieval and the procedure GenerateSequence are shown in Algorithm 2.

Algorithm 1 Outer level: the main procedure of *SPFH* algorithm

```
1:  procedure SPFH(S_0,P)
2:      Initialize S_cur ← S_0, sol_cur ← ∅, sol_best ← ∅, F_best = ∞
3:      AUTORETRIEVAL(S_cur,sol_cur,P)
4:      if P is empty then F_best ← 0, sol_best ← sol_cur
5:      else
6:          Q ← GENERATESEQUENCE(S_cur,P)
7:          while Q is not empty do
8:              q ← select first item from the pickup sequence set Q
9:              F_cur ←SIMULATION-EVALUATION(q,S_cur,sol_cur)
10:             if F_cur < F_best then F_best ← F_cur, sol_best ← sol_cur
11:             else if F_cur = F_best then sol_best ← sol_best ∪ sol_cur
12:             Q ← Q\{q}
13:     return F_best, sol_best
```

The objective of the inner level of SPFH is to simulate the relocation operation processes and estimate the associated operation costs concerning a given pickup sequence. This procedure involves removing the target containers of the current operation round and moving their blocking containers into the best placements with the least cost (the sum of the number of instant relocation moves and the addition of ENBC).

Algorithm 2 The procedures: AutoRetrieval and GenerateSequence

procedure AUTORETRIEVAL(S,sol,T)
  for all $t \in T$ **do**
    if t is topmost item in stack $w(t)$ **then**
      $S \leftarrow S \oplus \langle t, w(t), 0\rangle$, $sol \leftarrow sol + \langle t, w(t), 0\rangle$, $T \leftarrow T \backslash \{t\}$
procedure GENERATESEQUENCE(P)
  $W \leftarrow$Get the set of stacks containing the target containers in P.
  $top_1[W] \leftarrow$Get the top target container in each stack of W and remove them from corresponding stack.
  $Q_1 \leftarrow$Generate the set of full permutations from each target container in top_1.
  $Q_2 \leftarrow \varnothing$
  $top_2[W] \leftarrow$Get the top target container in each stack of W and remove them from corresponding stack.
  while An item in the $top_2[W]$ is not empty **do**
    for all $q_1 \in Q_1$ **do**
      for all $t_1 \in top_1$ **do**
        $l_1 \leftarrow$Finding the position index of t_1 in q_1
        for all $t_2 \in top_2$ **do**
          for all $i \in [l_1, len(q_1))$ **do**
            $q_2 \leftarrow$Insert t_2 sequentially into q_1 at ith item, and generate a new sequence.
            $Q_2 \leftarrow Q_2 \cup \{q_2\}$
    $Q_1 \leftarrow \varnothing$, $Q_1 \leftarrow Q_2$, $Q_2 \leftarrow \varnothing$, $top_1 \leftarrow top_2$
    $top_2[W] \leftarrow$Get the top target container in each stack of W and remove them from corresponding stack.
  return Q_1

The detailed simulation and evaluation procedure consists of the following steps. Firstly, we apply the LL rule to identify the optimal stack s^*, which will minimize the addition of ENBC among all assignable slots to which we can move the blocking container directly. However, the earlier reallocations of the blocking containers occupy reasonable placements and may lead subsequent blocking containers to lose the well-stacking slots. Hence, we will consider the impact of current reallocation decisions on subsequent blocking containers and determine whether there are better solutions than the current selection. Suppose the selected stack s^* is a sequential-stack for the current blocking container. In this case, we will check for some inverted-placement containers that satisfy the MSS rule to increase the chances of choosing a well-stacking position for other blocking containers. Similarly, suppose the selected stack s^* is a inverted-stack, we will try to find a stack s' satisfying the FSS rule that the stack s' is a sequential-stack for the current blocking container after moving the topmost container in stack s' to another stack. The optimal result is that the non-forced relocating container obtains a new well-stacking placement. Otherwise, relocating the non-forced container will increase the final operational cost in the future. Moreover, when stack s^* provides a level-placement for the current relocating container, neither the MSS rule nor the FSS rule is applicable because they may exacerbate the problem. Algorithm 3 presents the pseudo-code for the inner level of $SPFH$.

Algorithm 3 Inner level: Policy iteration procedure

procedure SIMULATION-EVALUATION(q, S, sol)
  $P \leftarrow q$,$S_{cur} \leftarrow S$;
  while P is not empty **do**
    $p \leftarrow$ obtains the first item of P
    $B_p \leftarrow$ obtain the blocking containers of the target container p
    for all $b \in B_p$ **do**
      $opr \leftarrow \varnothing$, $s^* \leftarrow LL(b, S_{cur})$ ▷ Get the stack for container b according to LL rule
      if $f(b, w(b), s^*) = 0$ **then**
        $s' \leftarrow MSS(s^*, b, S_{cur})$ ▷ Find a stack in satisfying MSS rule
        if s' is not **NULL then**
          $opr \leftarrow opr + \langle top(s'), s', s^* \rangle$, $opr \leftarrow opr + \langle b, w(b), s^* \rangle$
      else if $f(b, w(b), s^*) > 0$ **then**
        $s' \leftarrow FSS(s^*, b, S_{cur})$ ▷ Find a stack in satisfying FSS rule
        if s' is not **NULL then**
          $s'' \leftarrow LL(top(s'), S_{cur})$, $opr \leftarrow opr + \langle top(s'), s', s'' \rangle$, $opr \leftarrow opr + \langle b, w(b), s' \rangle$
        else
          $opr \leftarrow opr + \langle b, w(b), s^* \rangle$
      else
        $opr \leftarrow opr + \langle b, w(b), s^* \rangle$
      $S_{cur} \leftarrow S_{cur} \oplus opr$, $sol \leftarrow sol + opr$ ▷ $\oplus$ is an operator that indicates that opr will be executed based on the bay layout S_{cur} and generate a new layout.
      AUTORETRIEVAL(S_{cur},sol,P)
  return $\|opr\|$

5. Computational Experiments

In this section, we present some computational experiments to validate the effectiveness of the LL heuristic and $SPFH$ algorithm for the real-time operations of SCRP.

5.1. Implementation Details

Our experiments are implemented by Matlab 2016b and conducted on a PC with an Intel Core I7-9700 K CPU at 3.6 GHz and 16 GB RAM. The program code, results and instances used in this section are available at https://github.com/zhou-sf/rt-scrp (accessed on 16 July 2023). We generate our experiment dataset based on the existing dataset from [5] and complete our experiments from four cases. Each case of the experiments is implemented with 30 instances.

The first two experiments validate the effectiveness and performance of the LL heuristic and $SPFH$ algorithm under small-batch and large-batch instances, respectively. In the case of small-batch instances, the configuration sizes of the instances vary from $S = 5, \ldots, 10$ stacks and $T = 3, \ldots, 6$ tiers and batch sizes of each operation round vary from $B = 1, \ldots, 4$ containers. The configuration sizes of the large-batch instances vary from $S = 10, 12$ stacks and $T = 8, 9, 10$ tiers, and batch sizes of each operation round vary from $B = 5, \ldots, 12$ containers. The last experiment illustrates our algorithms' improvement compared to the algorithms proposed in [5,6].

5.2. Experiment Results

In this section, we will illustrate the numerical results to verify the validity and feasibility of this paper's methods under small-batch instances and large-batch instances, and also compare it with similar methods from other literature under small-batch instances.

5.2.1. Effectiveness Validation under Small-Batch Instances

A total of 48 configurations, each with 30 instances, are solved using the proposed heuristic LL and algorithm $SPFH$. Each configuration has $S \times T \times FillRate$ containers, where S is the number of stacks of the bay, T is the maximum number of tiers in a stack, and the FillRate is the occupancy rate of the slots in the bay. In this experiment, the value of S varies between 5 and 10, and the T varies between 3 and 6. The FillRate is divided into two cases: 50% and 67%. The batch size of each operation round in each instance is random and varies from $B = 1, \ldots, 4$.

(1) Comparison of complete time.

Tables 1 and 2 summarize the experiment's results as follows. The rows labeled as "C" indicate the container number of each configuration, and the rows labeled as "Solved" indicates that the number of instances solved optimally is provided in the form $x/30$. We can find that all 30 instances for each configuration are solved by $SPFH$ and LL in very short time, respectively. The "MaxTime (s)" and "AvgTime (s)" in the two tables refer to the runtime of the most time-consuming instance of each configuration and the average runtime of 30 instances of each configuration, respectively. As the tiers and the fill rates grow, given instances need longer solution time. Most important, the worst case in Table 1 is that an instance with configuration (five stacks and three layers) and with 67% fill rate has four target containers, and it takes about 0.134 s to solve. For these problem sizes, the solution time of most instances is less than 0.02 s. Since many ports today have a maximum tier requirement of four and need fast solutions, $SPFH$ and LL could be used in practices in small-batch cases.

Table 1. Runtime and completed instances of the heuristic LL with small batch instances.

	FillRate	**50 Percent**				**67 Percent**			
S	**T**	**3**	**4**	**5**	**6**	**3**	**4**	**5**	**6**
5	C	8	10	13	15	10	14	17	20
	Solved	30/30	30/30	30/30	30/30	30/30	30/30	30/30	30/30
	MaxTime (s)	0.043	0.005	0.008	0.011	0.134	0.008	0.011	0.011
	AvgTime (s)	0.005	0.002	0.003	0.004	0.002	0.003	0.004	0.005
6	C	9	12	15	18	12	16	20	24
	Solved	30/30	30/30	30/30	30/30	30/30	30/30	30/30	30/30
	MaxTime (s)	0.009	0.006	0.012	0.013	0.011	0.010	0.013	0.015
	AvgTime (s)	0.002	0.003	0.003	0.004	0.002	0.004	0.005	0.006
7	C	12	14	18	21	14	19	24	28
	Solved	30/30	30/30	30/30	30/30	30/30	30/30	30/30	30/30
	MaxTime (s)	0.006	0.011	0.011	0.013	0.011	0.014	0.013	0.021
	AvgTime (s)	0.003	0.003	0.004	0.006	0.003	0.004	0.005	0.007
8	C	12	16	20	24	16	21	27	32
	Solved	30/30	30/30	30/30	30/30	30/30	30/30	30/30	30/30
	MaxTime (s)	0.008	0.010	0.012	0.019	0.007	0.011	0.014	0.027
	AvgTime (s)	0.003	0.004	0.005	0.006	0.003	0.005	0.007	0.008
9	C	14	18	23	27	18	24	30	36
	Solved	30/30	30/30	30/30	30/30	30/30	30/30	30/30	30/30
	MaxTime (s)	0.008	0.009	0.013	0.014	0.011	0.013	0.020	0.026
	AvgTime (s)	0.003	0.004	0.006	0.007	0.004	0.005	0.008	0.009
10	C	15	20	25	30	20	27	34	40
	Solved	30/30	30/30	30/30	30/30	30/30	30/30	30/30	30/30
	MaxTime (s)	0.007	0.012	0.015	0.014	0.012	0.013	0.019	0.024
	AvgTime (s)	0.003	0.005	0.006	0.008	0.005	0.006	0.008	0.010

Table 2. Runtime and completed instances of the algorithm *SPFH* with small-batch instances.

	FillRate	**50 Percent**				**67 Percent**			
S	**T**	**3**	**4**	**5**	**6**	**3**	**4**	**5**	**6**
5	C	8	10	13	15	10	14	17	20
	Solved	30/30	30/30	30/30	30/30	30/30	30/30	30/30	30/30
	MaxTime (s)	0.022	0.021	0.014	0.021	0.038	0.016	0.027	0.023
	AvgTime (s)	0.003	0.003	0.004	0.005	0.002	0.003	0.005	0.007
6	C	9	12	15	18	12	16	20	24
	Solved	30/30	30/30	30/30	30/30	30/30	30/30	30/30	30/30
	MaxTime (s)	0.013	0.012	0.022	0.016	0.014	0.020	0.021	0.030
	AvgTime (s)	0.003	0.003	0.004	0.005	0.003	0.004	0.006	0.008
7	C	12	14	18	21	14	19	24	28
	Solved	30/30	30/30	30/30	30/30	30/30	30/30	30/30	30/30
	MaxTime (s)	0.016	0.016	0.011	0.015	0.020	0.026	0.021	0.027
	AvgTime (s)	0.003	0.004	0.005	0.007	0.003	0.005	0.006	0.009
8	C	12	16	20	24	16	21	27	32
	Solved	30/30	30/30	30/30	30/30	30/30	30/30	30/30	30/30
	MaxTime (s)	0.013	0.014	0.013	0.024	0.014	0.018	0.021	0.029
	AvgTime (s)	0.003	0.004	0.006	0.008	0.003	0.006	0.008	0.011
9	C	14	18	23	27	18	24	30	36
	Solved	30/30	30/30	30/30	30/30	30/30	30/30	30/30	30/30
	MaxTime (s)	0.012	0.014	0.018	0.015	0.012	0.021	0.033	0.038
	AvgTime (s)	0.004	0.005	0.008	0.008	0.004	0.006	0.010	0.011
10	C	15	20	25	30	20	27	34	40
	Solved	30/30	30/30	30/30	30/30	30/30	30/30	30/30	30/30
	MaxTime (s)	0.008	0.021	0.019	0.023	0.013	0.021	0.028	0.037
	AvgTime (s)	0.004	0.006	0.007	0.009	0.005	0.007	0.010	0.014

(2) Analysis of the results

Tables 3 and 4 show us the statistical information of the experimental results of *LL* and *SPFH* under 50% and 67% fill rate, respectively. The *IB* refers to the initial number of blocking containers for each instance. Its value equals the sum of the number of forced relocations and potential relocations of a bay's initial layout, determined by the number of the inverted-placement and level-placement containers. The I_{eb} indicates the average of the increment in the expected number of new additional blocking containers and the number of movements other than the forced relocations during each operation round of a bay.

The *IB* shows the average value of the initial blocking container number, which is nearly half the sum of the maximum and minimum values of the initial blocking container number for each configuration's instances. Furthermore, the standard deviation of *IB*, *Stdev*, increases with the instance configuration's size, which suggests that the number of potential relocations grows as the instance size grows. A similar pattern can be found in the standard deviations of I_{eb} of *LL* and *SPFH*. However, the I_{eb} of *LL* and *SPFH* exhibit greater randomness, and it is since each instance layout varies greatly and the pickup sequence of target containers is adjustable, the incremental number of potential relocations generated during the practical operations is indeterminate. In addition, the value of I_{eb} in *LL* differs significantly from that in *SPFH*, mainly because *SPFH* introduces two relocation rules *MSS* and *FSS* on top of *LL*, and the two rules avoid many potential relocations caused by container movements. The *Act* refers to the average value of each instance configuration's actual number of relocations. Its value is statistically derived from the optimized operation plan.

Table 3. The results of the *LL* heuristic and the *SPFH* algorithm under 50% fill rate.

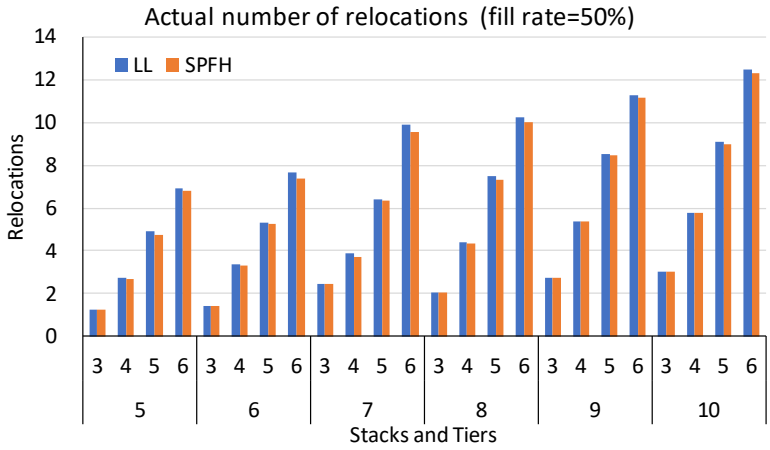
Fill Rate = 50%						**LL**					**SPFH**				
S	**T**	**IB**	**Max**	**Min**	**Stdev**	**Ieb**	**Max**	**Min**	**Stdev**	**Act**	**Ieb**	**Max**	**Min**	**Stdev**	**Act**
5	3	1.61	3.00	0.50	0.70	0.00	0.00	0.00	0.00	1.27	0.03	1.00	0.00	0.18	1.27
	4	2.83	5.00	0.50	1.22	0.08	1.00	0.00	0.26	2.77	0.05	1.00	0.00	0.20	2.70
	5	4.48	5.50	2.50	0.93	0.33	1.50	0.00	0.47	4.90	0.57	2.50	0.00	0.64	4.73
	6	6.11	10.00	3.00	1.68	0.83	4.00	0.00	0.92	6.90	1.28	5.00	0.00	1.17	6.80
6	3	1.63	3.50	0.00	0.94	0.02	0.50	0.00	0.09	1.43	0.08	1.00	0.00	0.26	1.43
	4	3.49	6.00	1.00	1.29	0.03	1.00	0.00	0.18	3.37	0.23	2.00	0.00	0.50	3.33
	5	5.31	8.00	2.00	1.32	0.22	1.50	0.00	0.46	5.30	0.48	2.00	0.00	0.58	5.27
	6	7.05	10.50	4.00	1.55	0.70	4.00	0.00	0.86	7.63	1.05	3.00	0.00	1.00	7.37
7	3	2.75	4.00	0.50	0.96	0.00	0.00	0.00	0.00	2.43	0.03	1.00	0.00	0.18	2.43
	4	3.93	5.00	2.00	0.76	0.10	1.00	0.00	0.30	3.87	0.20	1.00	0.00	0.40	3.73
	5	6.42	9.50	3.00	1.39	0.22	1.00	0.00	0.33	6.40	0.60	3.00	0.00	0.81	6.37
	6	8.77	12.00	5.50	1.57	0.88	3.00	0.00	1.06	9.90	1.27	5.00	0.00	1.27	9.53
8	3	2.22	4.00	0.00	1.17	0.00	0.00	0.00	0.00	2.03	0.10	1.00	0.00	0.30	2.03
	4	4.68	8.00	1.50	1.62	0.06	1.00	0.00	0.21	4.40	0.26	1.00	0.00	0.43	4.37
	5	7.18	10.50	3.50	1.71	0.32	2.00	0.00	0.63	7.47	0.93	3.50	0.00	0.96	7.30
	6	9.45	13.00	4.50	1.85	0.87	4.00	0.00	1.19	10.23	1.32	5.00	0.00	1.27	10.00
9	3	2.93	5.00	1.00	1.05	0.00	0.00	0.00	0.00	2.77	0.10	1.00	0.00	0.30	2.77
	4	5.58	7.50	3.00	1.07	0.00	0.00	0.00	0.00	5.40	0.23	2.00	0.00	0.50	5.40
	5	8.55	13.00	5.00	2.13	0.20	2.00	0.00	0.46	8.50	0.67	2.00	0.00	0.65	8.43
	6	10.40	14.00	6.00	2.00	0.67	3.00	0.00	0.97	11.27	1.69	4.00	0.00	1.12	11.13
10	3	3.18	6.00	1.00	1.35	0.00	0.00	0.00	0.00	3.03	0.20	2.00	0.00	0.48	3.03
	4	6.15	9.00	3.00	1.25	0.03	0.50	0.00	0.12	5.77	0.35	2.00	0.00	0.59	5.80
	5	9.07	13.00	5.50	1.54	0.13	1.00	0.00	0.31	9.07	1.07	3.00	0.00	0.94	8.97
	6	11.90	16.50	5.50	2.40	0.57	3.50	0.00	0.88	12.47	1.43	3.00	0.00	0.85	12.27

Table 4. The results of the *LL* heuristic and the *SPFH* algorithm under 67% fill rate.

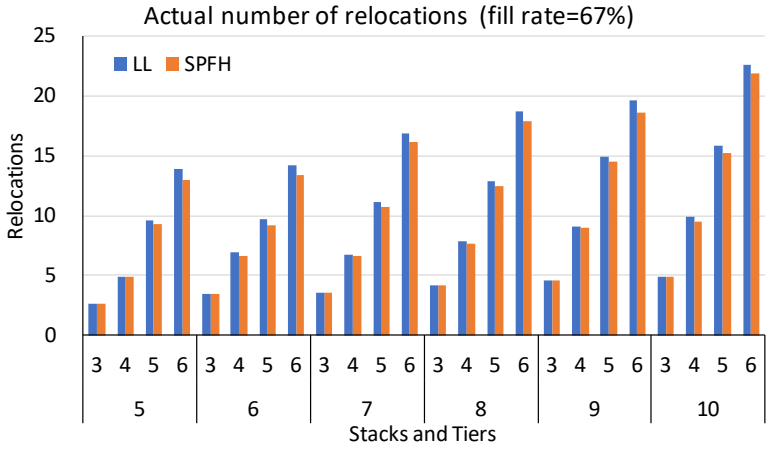
Fill Rate = 67%						**LL**					**SPFH**				
S	**T**	**IB**	**Max**	**Min**	**Stdev**	**Ieb**	**Max**	**Min**	**Stdev**	**Act**	**Ieb**	**Max**	**Min**	**Stdev**	**Act**
5	3	2.70	4.50	1.00	0.86	0.02	0.50	0.00	0.09	2.63	0.18	1.00	0.00	0.38	2.63
	4	4.59	7.00	2.00	1.20	0.47	4.00	0.00	0.85	4.93	0.67	3.50	0.00	0.89	4.87
	5	7.50	11.00	4.00	1.60	1.21	4.00	0.00	0.98	9.57	1.71	5.00	0.00	0.98	9.33
	6	9.62	13.50	6.17	1.68	3.17	6.50	0.00	1.76	13.83	3.43	7.50	0.50	1.62	12.93
6	3	3.55	6.00	0.50	1.18	0.08	1.00	0.00	0.26	3.47	0.28	1.00	0.00	0.44	3.50
	4	6.13	9.00	3.00	1.48	0.52	2.00	0.00	0.65	6.90	0.90	3.00	0.00	0.82	6.63
	5	8.22	11.50	5.00	1.56	1.25	5.00	0.00	1.29	9.70	1.63	7.00	0.00	1.41	9.20
	6	11.57	15.00	7.50	1.78	2.42	6.00	0.00	1.61	14.17	2.92	8.00	0.00	2.00	13.33
7	3	3.83	7.00	1.00	1.24	0.07	1.00	0.00	0.21	3.60	0.17	1.00	0.00	0.35	3.60
	4	6.20	9.00	2.00	1.70	0.57	2.00	0.00	0.70	6.77	0.80	4.00	0.00	1.00	6.60
	5	9.68	12.50	6.00	1.58	1.31	4.00	0.00	1.11	11.17	1.81	4.00	0.00	1.11	10.73
	6	13.42	17.00	8.50	1.92	2.35	5.50	0.00	1.36	16.87	3.00	6.50	0.00	1.40	16.10
8	3	4.41	7.00	1.00	1.53	0.07	1.00	0.00	0.25	4.20	0.33	2.00	0.00	0.54	4.17
	4	7.60	11.50	4.00	1.85	0.37	4.00	0.00	0.81	7.83	0.75	3.00	0.00	0.87	7.67
	5	11.54	15.00	6.50	1.67	1.13	4.00	0.00	1.10	12.90	2.23	5.00	0.00	1.26	12.47
	6	15.57	20.50	11.50	2.49	2.38	7.50	0.00	1.83	18.70	3.28	9.50	0.00	2.05	17.83
9	3	4.79	8.00	1.50	1.43	0.08	1.00	0.00	0.26	4.60	0.35	1.00	0.00	0.47	4.60
	4	8.98	13.00	6.00	1.72	0.35	2.00	0.00	0.55	9.13	1.10	4.00	0.00	1.03	9.00
	5	13.11	17.00	8.67	1.79	1.35	4.00	0.00	1.30	14.90	2.20	5.00	0.00	1.35	14.47
	6	16.67	20.50	12.17	1.98	2.83	7.50	0.00	2.27	19.63	3.80	8.50	0.00	1.91	18.60
10	3	5.21	8.17	2.00	1.31	0.08	1.00	0.00	0.26	4.90	0.32	2.00	0.00	0.52	4.87
	4	9.18	13.00	5.50	2.06	0.67	4.67	0.00	1.02	9.90	1.05	3.00	0.00	0.93	9.53
	5	14.44	19.50	8.50	2.43	1.28	6.00	0.00	1.37	15.87	1.95	6.00	0.00	1.41	15.17
	6	19.52	23.50	14.00	2.39	2.48	7.00	0.00	1.90	22.60	3.73	9.00	0.00	2.07	21.80

According to Equation (9), the value of Act theoretically does not exceed the value of $(IB + I_{eb})$. However, in Tables 3 and 4, there are cases where the value of Act is greater than the value of $(IB + I_{eb})$. This case can happen because some blocking containers caused by level-placement containers produce some realistic relocations. Meanwhile, this case happens more in LL and less in $SPFH$. The reason for this case is because the LL heuristic, when selecting a position for the blocking container, will prioritize selecting a level stack in the absence of sequential candidate stacks (selecting level stack produces the increment of the expected number of potential blocking container less than selecting inverted stack). Some potential blocking containers may produce realized relocations, so there are more realized relocations than the theoretical optimal operation solution. However, this situation rarely occurs in $SPFH$ because the relocation rule FSS in $SPFH$ always tries to free up a sequential stack for the current blocking container.

As shown in Figure 5, the values of Act in $SPFH$ are better than those of LL, especially at the fill rate of 67%. Intuitively, the $SPFH$ algorithm outperforms the LL heuristic. Still, the relocation rules MSS and FSS, which allow moving other containers unrelated to the target container, may prolong the loading operation of a few ETs during some operation rounds.



(**a**) Comparison of the actual relocation number at 50% of fillrate.



(**b**) Comparison of the actual relocation number at 67% of fillrate.

Figure 5. Comparison between $SPFH$ and LL at fill rate 50% and 67% respectively.

5.2.2. Effectiveness Validation under Large-Batch Instances

In this experiment, the configuration sizes of the experimental instances vary from $S = 10,\ 12$ stacks and $T = 8,\ \ldots,\ 10$ tiers. The batch size varies from $B = 5,\ \ldots,\ 12$ and

the batch number varies from $W = 8, 10$. So, each instance configuration has a different file rate.

Table 5 summarizes the validation of the algorithm $SPFH$ and the heuristic LL for the large-size instances in the real-time operation scenarios. The column titled "MT" refers to the runtime of the most time-consuming instance out of 30 instances. The column "AT" refers to the average runtime of 30 instances. The meanings of other variables, namely I, IB, I_{eb}, and Act, are the same as those in the previous tables.

Table 5. Validation of the heuristic LL and algorithm $SPFH$ with large instances in the real-time operation scenario of SCRP.

							LL				**SPFH**			
S	**T**	**W**	**B**	**Fill Rate**	**I**	**IB**	**MT**	**AT**	I_{eb}	**Act**	**MT**	**AT**	I_{eb}	**Act**
10	8	8	5	0.500	30/30	18.23	0.35	0.03	3.11	26.99	0.18	0.04	4.50	27.27
			6	0.600	30/30	23.43	0.56	0.06	6.75	33.49	1.16	0.10	7.34	34.94
			7	0.700	30/30	29.67	4.98	0.32	12.71	46.41	8.26	0.50	12.97	46.21
			8	0.800	30/30	36.70	34.97	2.57	20.74	61.32	49.47	3.89	22.36	62.50
		10	4	0.500	30/30	17.73	0.05	0.01	3.68	25.27	0.10	0.02	4.75	27.74
			5	0.625	30/30	25.80	0.19	0.04	6.73	37.65	0.34	0.06	8.04	38.17
			6	0.750	30/30	32.47	0.56	0.09	14.57	51.65	0.96	0.13	14.99	52.72
			7	0.875	30/30	41.77	6.21	0.60	23.93	71.10	11.46	1.33	25.51	71.10
	9	8	6	0.533	30/30	22.83	0.74	0.07	6.48	34.24	1.26	0.11	7.71	35.85
			7	0.622	30/30	29.67	1.68	0.17	10.67	46.10	2.82	0.29	11.69	45.92
			8	0.711	30/30	36.30	7.09	0.61	19.35	59.31	12.09	0.98	19.19	59.18
			9	0.800	30/30	41.13	40.47	5.01	31.87	77.04	58.68	8.74	35.40	80.77
		10	5	0.556	30/30	25.53	0.11	0.03	7.13	36.94	0.17	0.04	8.63	37.83
			6	0.667	30/30	33.37	1.26	0.12	12.99	51.84	2.14	0.20	14.59	52.15
			7	0.778	30/30	42.40	5.80	0.54	24.09	71.75	10.07	0.89	25.03	72.97
12	10	8	8	0.533	30/30	33.97	1.44	0.27	9.45	46.70	2.55	0.43	10.19	49.95
			9	0.600	30/30	39.00	2358.47	84.57	19.60	62.58	94.83	9.97	23.41	66.14
			10	0.667	30/30	45.07	93.00	16.98	38.05	88.81	167.34	26.58	46.43	95.01
			11	0.733	12/30	50.94	7377.33	478.17	48.99	103.61	4010.35	373.49	59.00	113.84
			12	0.800	0/30	0.00	0.00	0.00	0.00	-	0.00	0.00	0.00	0.00
		10	8	0.667	30/30	49.61	68.87	3.79	24.44	78.34	111.87	6.33	27.21	81.81
			9	0.750	30/30	57.92	69.71	3.95	33.42	96.96	113.61	6.39	35.35	99.00
			10	0.833	30/30	67.15	4115.17	628.84	66.94	138.10	3925.53	567.58	60.74	133.42
			11	0.917	20/30	72.14	4818.37	841.89	99.55	176.27	3925.53	888.10	90.04	166.60

The results in Table 5 imply that both algorithms $SPFH$ and LL can solve most instances quickly. However, when the configuration of instances reaches 12 stacks and 10 tiers, if the number of target containers per round is more than 8 (the number of target containers here does not include those can be directly picked up), the solution time of some instances rises rapidly, even more than 2000 s, which means that the algorithms no longer meet the needs of real-time optimization operations. When the number of target containers per round is larger than 10, it is difficult to solve any one instance in a short time. Fortunately, in the actual operation scenario, there are rarely more than ten target containers in most container terminals during a real-time operation round.

The reason for the time-consuming solution process in the large-size batch instances is that the flexible pickup order mechanism requires iterating through all possible target pickup orders during the real-time operation round to explore the optimal operation plan. The number (r) of iterations of the solution process increases rapidly with the number (m) of stacks where the target containers are stacked and the target container number (n), and the value of r is between $m!$ and $n!$. In fact, if both n and m exceed 8, the method proposed in this paper will have difficulty solving in less than 60 s. Two perspectives can be considered to solve this problem: first, reducing the solution quality, using heuristics for selecting

target containers (e.g., always retrieving the target containers with the lowest number of blocking containers) instead of complete iteration to improve the solution efficiency, and, second, dividing the current larger batch into two smaller batches before processing, e.g., the current batch of 10 containers is divided into two small batches of 5 containers.

Under large-batch instances, it is more common for the value of Act to be greater than the theoretical upper bound, $(IB + I_{eb})$. The reason for this case has been given in the experiment mentioned above under small-batch instances. Moreover, it can be seen that the discrepancy between the Act and $(IB + I_{eb})$ decreases with increasing batch size.

5.2.3. Comparison with Other Algorithms under Small-Batch Instances

In this experiment, we chose [5,6] for comparison, mainly because our experimental dataset was derived from them. Moreover, they are the two most outstanding researchers of SCRP in the existing literature. They aim to find optimal operation plans for yard cranes for retrieving all the containers in a bay while minimizing the number of relocations. This optimization objective is the same as this paper, but the realization method differs. We directly compare the experimental results with those of [5,6] to illustrate the improvement of our methods in performance. In Tables 6 and 7, EM (heuristic) as well as $PBFS$ and $PBFSA$ (two variants of B&B) are proposed by [5] and the the heuristic ERI is proposed by [6].

Table 6. Performance of heuristic LL and algorithm $SPFH$ for the fill rate 50%.

		LL					**SPFH**				
S	**T**	**Act**	**EM**	**ERI**	**PBFS**	**PBFSA**	**Act**	**EM**	**ERI**	**PBFS**	**PBFSA**
5	3	1.27	25.49%	25.49%	25.49%	28.03%	1.27	25.49%	25.49%	25.49%	28.03%
	4	2.70	13.74%	14.01%	13.18%	18.92%	2.70	13.74%	14.01%	13.18%	18.92%
	5	4.73	12.02%	15.02%	11.03%	13.94%	4.73	12.02%	15.02%	11.03%	13.94%
	6	6.80	12.51%	15.53%	9.97%	12.51%	6.83	12.93%	15.95%	10.41%	12.93%
6	3	1.43	17.62%	17.62%	17.62%	17.15%	1.43	17.62%	17.62%	17.62%	17.15%
	4	3.33	9.42%	9.42%	9.42%	15.18%	3.33	9.42%	9.42%	9.42%	15.18%
	5	5.27	10.21%	11.26%	9.76%	12.71%	5.33	11.34%	12.37%	10.89%	13.80%
	6	7.37	9.93%	13.48%	9.28%	-	7.47	11.14%	14.64%	10.49%	-
7	3	2.43	15.51%	15.80%	15.51%	14.62%	2.43	15.51%	15.80%	15.51%	14.62%
	4	3.73	9.67%	10.32%	9.46%	11.16%	3.77	10.47%	11.11%	10.26%	11.95%
	5	6.37	9.05%	9.95%	8.66%	5.96%	6.37	9.05%	9.95%	8.66%	5.96%
	6	9.53	6.60%	10.16%	-	-	9.67	7.89%	11.40%	-	-
8	3	2.03	11.98%	11.98%	11.98%	11.98%	2.03	11.98%	11.98%	11.98%	11.98%
	4	4.37	8.90%	8.90%	8.71%	6.58%	4.40	9.59%	9.59%	9.41%	7.29%
	5	7.30	6.26%	7.78%	5.31%	-	7.43	7.94%	9.43%	7.01%	-
	6	10.00	7.49%	10.35%	-	-	10.17	9.01%	11.82%	-	-
9	3	2.77	8.08%	7.78%	7.78%	6.21%	2.77	8.08%	7.78%	7.78%	6.21%
	4	5.40	5.76%	5.76%	5.76%	5.10%	5.40	5.76%	5.76%	5.76%	5.10%
	5	8.43	6.87%	7.89%	-	-	8.40	6.50%	7.53%	-	-
	6	11.13	2.42%	3.91%	-	-	11.30	3.86%	5.33%	-	-
10	3	3.03	5.21%	5.21%	4.91%	3.70%	3.03	5.21%	5.21%	4.91%	3.70%
	4	5.80	8.17%	8.17%	8.17%	8.17%	5.77	7.64%	7.64%	7.64%	7.64%
	5	8.97	6.25%	7.50%	-	-	9.00	6.60%	7.85%	-	-
	6	12.27	4.28%	5.96%	-	-	12.37	5.06%	6.72%	-	-

Table 7. Performance of heuristic *LL* and algorithm *SPFH* for the fill rate 67%.

		LL					SPFH				
S	**T**	**Act**	**EM**	**ERI**	**PBFS**	**PBFSA**	**Act**	**EM**	**ERI**	**PBFS**	**PBFSA**
5	3	2.63	14.50%	15.33%	14.50%	14.22%	2.63	14.50%	15.33%	14.50%	14.22%
	4	4.87	13.12%	14.78%	12.19%	14.04%	4.90	13.71%	15.36%	12.78%	14.62%
	5	9.33	9.03%	13.65%	-	-	9.53	10.94%	15.46%	-	-
	6	12.93	7.17%	13.76%	-	-	13.60	11.72%	17.99%	-	-
6	3	3.50	11.11%	11.11%	10.88%	11.11%	3.47	10.26%	10.26%	10.03%	10.26%
	4	6.63	9.17%	11.08%	7.05%	5.63%	6.77	10.96%	12.83%	8.88%	7.48%
	5	9.20	8.16%	11.73%	-	-	9.53	11.37%	14.81%	-	-
	6	13.33	6.61%	12.23%	-	-	14.17	12.11%	17.39%	-	-
7	3	3.60	9.55%	10.45%	9.32%	13.04%	3.60	9.55%	10.45%	9.32%	13.04%
	4	6.60	9.42%	11.58%	8.55%	-	6.67	10.33%	12.47%	9.47%	-
	5	10.73	5.49%	10.03%	-	-	11.07	8.34%	12.74%	-	-
	6	16.10	4.71%	10.69%	-	-	16.80	8.68%	14.41%	-	-
8	3	4.17	10.59%	10.97%	10.59%	14.09%	4.17	10.59%	10.97%	10.59%	14.09%
	4	7.67	7.78%	8.66%	7.12%	-	7.70	8.18%	9.05%	7.52%	-
	5	12.47	4.97%	9.22%	-	-	12.80	7.45%	11.58%	-	-
	6	17.83	6.03%	12.07%	-	-	18.33	8.59%	14.47%	-	-
9	3	4.60	10.16%	10.51%	9.80%	10.51%	4.60	10.16%	10.51%	9.80%	10.51%
	4	9.00	5.65%	7.10%	5.36%	-	9.07	6.35%	7.79%	6.05%	-
	5	14.47	6.06%	11.44%	-	-	14.83	8.38%	13.63%	-	-
	6	18.60	4.25%	10.64%	-	-	19.53	8.82%	14.91%	-	-
10	3	4.87	7.20%	7.20%	7.02%	7.72%	4.90	7.83%	7.83%	7.65%	8.35%
	4	9.53	3.79%	4.53%	-	-	9.90	7.35%	8.07%	-	-
	5	15.17	4.06%	6.93%	-	-	15.53	6.32%	9.13%	-	-
	6	21.80	3.30%	8.29%	-	-	22.43	6.03%	10.87%	-	-

Tables 6 and 7 give us the summaries of the improvement of the solution quality between the *SPFH* and *LL* and the *EM*, *ERI*, *PBFS*, and *PBFSA* on the same data set with fill rate 50% and 67%, respectively. The experiment results show that the algorithms *PBFS* and *PBFSA* are not applicable to the real-time operation scenario of SCRP when the configuration size (S, T) exceeds $(7, 5)$ with the 50% fill rate or $(5, 5)$ with the 67% fill rate. All heuristics are able to complete all experimental instances, and the experimental results show that the solution quality of *LL* and *SPFH* improves to varying degrees for the heuristics *EM* and *ERI*. In terms of the trend of solution quality, the smaller the problem size, the more significant the improvement in solution quality. Meanwhile, the trend of improving the solution quality decreases as the number of stack tiers grows, indicating that the growth of the number of containers increases the proportion of uncertain information and the complexity of the solution process of the SCRP and decreases the quality of the solution.

The main reason why *LL* and *SPFH* outperform *EM*, *ERI*, *PBFS*, and *PBFSA* is that our method fully utilizes the pickup information of the arrived ETs and the adjustable strategy of the containers' pickup orders, and they increase the chances of obtaining a better solution. In fact, the terminal allows for yard operators to adjust the operation sequence of different ETs according to operational needs. This also shows that our methods are reasonable and practical.

6. Conclusions

This paper addresses issues related to optimizing the real-time pickup operations of import containers based on the appointment and arrival information of ETs to reduce the number of relocations. Our results include a method (namely *Real-time Batch Optimization*), which only focuses on the real-time pickup operation round of import containers, a dynamic upper bound of the theoretical optimal solution of the SCRP, and a heuristic

algorithm based on three relocation rules. The heuristic relocation rule, *LL*, can be used as a standalone algorithm for operation scenarios of restricted SCRP, and the algorithm *SPFH* is applied to operation scenarios of unrestricted SCRP. Numerical experiments show that the newly proposed method exhibits excellent performance on both small-batch and large-batch instance sets, and its solution efficiency and quality outperform other methods in the literature. The reason for the new method's performance compared with others in the literature is that it focuses only on real-time operations, i.e., a part of the overall retrieval process, which significantly reduces the solution process size. In addition, the adjustment of the target containers' pickup order helps us choose a better operation plan. The newly proposed method in this paper is suitable for large-throughput and busy-operation container terminals because more than one ET may arrive during most of the operation rounds, while it is difficult to achieve good optimization results for small-throughput and loose-operation container terminals because not more than one ET arrives at during most of the operation rounds. Since the SCRP is an NP-hard problem, reducing the problem size is the main idea to improve its solution efficiency. In this paper, focusing on only one stage of the overall pickup operation process of imported containers significantly reduces the problem size, which provides ideas for efficiently solving large-scale instances. Meanwhile, this idea can be used to solve other NP-hard problems related to container terminal yard operation optimization.

However, the method proposed in this paper has two potential limitations: retrieval priority of the containers and target containers' pickup sequence during each real-time operation round. First, the retrieval priority of containers is crucial for estimating the ENBC and deciding the new slot for the blocking containers. But, this paper relies on the appointment information from the TAS to determine containers' retrieval priority. Theoretically, the greater the proportion of containers within a bay booked in advance, the higher the accuracy of the solution. Suppose none of the containers within a bay have been booked in advance. In that case, the method proposed in this paper will degenerate into a general SCRP, except that the problem size is reduced, while the accuracy of the solution is unknown. The second limitation is caused by the procedure GenerateSequence in algorithm *SPFH*, and its solution approach has been mentioned in Section 5.2.2.

In view of the limitation and applicability, further research can be carried out in the following directions based on this paper. First, using machine learning algorithms to mine and analyze the operation logs of container terminals to predict the customers' retrieval time and infer the containers' pickup priority is a worthwhile area of research for improving the solution quality of SCRP due to insufficient appointment information. Alternatively, mining the operation logs of the container terminal to derive the probability distribution of customers' retrieval for estimating the ENBC is a valuable topic, too. Second, by extending the new method proposed in this paper to the CRP of export containers, the container pre-marshaling problem (PMP) or the storage allocation problem (SAP) of import containers will be our next major concern. Moreover, the integrated optimization of the container relocation problem, yard crane scheduling, and balancing the waiting time of ETs is a challenging topic for future studies.

Author Contributions: Conceptualization, Q.Z.; Methodology, S.Z.; Writing—original draft, S.Z.; Writing—review & editing, Q.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data presented in the study are openly available in GitHub at https://github.com/zhou-sf/rt-scrp (accessed on 5 January 2024).

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

References

1. Caserta, M.; Schwarze, S.; Voß, S. Container Rehandling at Maritime Container Terminals. In *Handbook of Terminal Planning*; Böse, J.W., Ed.; Springer: New York, NY, USA, 2011; pp. 247–269. [CrossRef]
2. Feng, Y.; Song, D.P.; Li, D.; Zeng, Q. The stochastic container relocation problem with flexible service policies. *Transp. Res. Part B Methodol.* **2020**, *141*, 116–163. [CrossRef]
3. Zeng, Q.; Feng, Y.; Yang, Z. Integrated optimization of pickup sequence and container rehandling based on partial truck arrival information. *Comput. Ind. Eng.* **2019**, *127*, 366–382. [CrossRef]
4. Azab, A.; Morita, H. Coordinating truck appointments with container relocations and retrievals in container terminals under partial appointments information. *Transp. Res. Part E Logist. Transp. Rev.* **2022**, *160*, 102673. [CrossRef]
5. Galle, V.; Manshadi, V.H.; Boroujeni, S.B.; Barnhart, C.; Jaillet, P. The Stochastic Container Relocation Problem. *Transp. Sci.* **2018**, *52*, 1035–1058. [CrossRef]
6. Ku, D.; Arthanari, T.S. Container relocation problem with time windows for container departure. *Eur. J. Oper. Res.* **2016**, *252*, 1031–1039. [CrossRef]
7. de Castillo, B.; Daganzo, C.F. Handling strategies for import containers at marine terminals. *Transp. Res. Part B* **1993**, *27*, 151–166. [CrossRef]
8. Kim, K.H. Evaluation of the number of rehandles in container yards. *Comput. Ind. Eng.* **1997**, *32*, 701–711. [CrossRef]
9. Saurí, S.; Martín, E. Space allocating strategies for improving import yard performance at marine terminals. *Transp. Res. Part E Logist. Transp. Rev.* **2011**, *47*, 1038–1057. [CrossRef]
10. Zhou, C.; Wang, W.; Li, H. Container reshuffling considered space allocation problem in container terminals. *Transp. Res. Part E Logist. Transp. Rev.* **2020**, *136*, 101869. [CrossRef]
11. Feng, Y.; Song, D.P.; Li, D. Smart stacking for import containers using customer information at automated container terminals. *Eur. J. Oper. Res.* **2022**, *301*, 502–522. [CrossRef]
12. Carlo, H.J.; Vis, I.F.; Roodbergen, K.J. Storage yard operations in container terminals: Literature overview, trends, and research directions. *Eur. J. Oper. Res.* **2014**, *235*, 412–430. [CrossRef]
13. Lersteau, C.; Shen, W. A survey of optimization methods for Block Relocation and PreMarshalling Problems. *Comput. Ind. Eng.* **2022**, *172*, 108529. [CrossRef]
14. Petering, M.E.; Hussein, M.I. A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem. *Eur. J. Oper. Res.* **2013**, *231*, 120–130. [CrossRef]
15. Zehendner, E.; Caserta, M.; Feillet, D.; Schwarze, S.; Voß, S. An improved mathematical formulation for the blocks relocation problem. *Eur. J. Oper. Res.* **2015**, *245*, 415–422. [CrossRef]
16. Forster, F.; Bortfeldt, A. A tree search procedure for the container relocation problem. *Comput. Oper. Res.* **2012**, *39*, 299–309. [CrossRef]
17. Galle, V.; Barnhart, C.; Jaillet, P. A new binary formulation of the restricted Container Relocation Problem based on a binary encoding of configurations. *Eur. J. Oper. Res.* **2018**, *267*, 467–477. [CrossRef]
18. Tanaka, S.; Voß, S. An exact approach to the restricted block relocation problem based on a new integer programming formulation. *Eur. J. Oper. Res.* **2022**, *296*, 485–503. [CrossRef]
19. Tricoire, F.; Scagnetti, J.; Beham, A. New insights on the block relocation problem. *Comput. Oper. Res.* **2018**, *89*, 127–139. [CrossRef]
20. Feillet, D.; Parragh, S.N.; Tricoire, F. A local-search based heuristic for the unrestricted block relocation problem. *Comput. Oper. Res.* **2019**, *108*, 44–56. [CrossRef]
21. Zhu, W.; Qin, H.; Lim, A.; Zhang, H. Iterative deepening A* algorithms for the container relocation problem. *IEEE Trans. Autom. Sci. Eng.* **2012**, *9*, 710–722. [CrossRef]
22. Quispe, K.E.; Lintzmayer, C.N.; Xavier, E.C. An exact algorithm for the Blocks Relocation Problem with new lower bounds. *Comput. Oper. Res.* **2018**, *99*, 206–217. [CrossRef]
23. Kim, K.H.; Hong, G.P. A heuristic rule for relocating blocks. *Comput. Oper. Res.* **2006**, *33*, 940–954. [CrossRef]
24. Tanaka, S.; Takii, K. A faster branch-and-bound algorithm for the block relocation problem. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 181–190. [CrossRef]
25. de Melo da Silva, M.; Erdoğan, G.; Battarra, M.; Strusevich, V. The Block Retrieval Problem. *Eur. J. Oper. Res.* **2018**, *265*, 931–950. [CrossRef]
26. Zhang, C.; Guan, H. A data-driven exact algorithm for the container relocation problem. In Proceedings of the IEEE 16th International Conference on Automation Science and Engineering (CASE), Hong Kong, China, 20–21 August 2020; pp. 1349–1354. [CrossRef]
27. Bacci, T.; Mattia, S.; Ventura, P. A branch-and-cut algorithm for the restricted Block Relocation Problem. *Eur. J. Oper. Res.* **2020**, *287*, 452–459. [CrossRef]
28. Jin, B.; Tanaka, S. An exact algorithm for the unrestricted container relocation problem with new lower bounds and dominance rules. *Eur. J. Oper. Res.* **2023**, *304*, 494–514. [CrossRef]
29. Caserta, M.; Schwarze, S.; Voß, S. A mathematical formulation and complexity considerations for the blocks relocation problem. *Eur. J. Oper. Res.* **2012**, *219*, 96–104. [CrossRef]
30. Ting, C.J.; Wu, K.C. Optimizing container relocation operations at container yards with beam search. *Transp. Res. Part E Logist. Transp. Rev.* **2017**, *103*, 17–31. [CrossRef]

31. Bacci, T.; Mattia, S.; Ventura, P. The bounded beam search algorithm for the block relocation problem. *Comput. Oper. Res.* **2019**, *103*, 252–264. [CrossRef]
32. Jovanovic, R.; Voß, S. A chain heuristic for the Blocks Relocation Problem. *Comput. Ind. Eng.* **2014**, *75*, 79–86. [CrossRef]
33. Jin, B.; Zhu, W.; Lim, A. Solving the container relocation problem by an improved greedy look-ahead heuristic. *Eur. J. Oper. Res.* **2014**, *240*, 837–847. [CrossRef]
34. Cifuentes, C.D.; Riff, M.C. G-CREM: A GRASP approach to solve the container relocation problem for multibays. *Appl. Soft Comput.* **2020**, *97*, 106721. [CrossRef]
35. Zehendner, E.; Feillet, D.; Jaillet, P. An algorithm with performance guarantee for the Online Container Relocation Problem. *Eur. J. Oper. Res.* **2017**, *259*, 48–62. [CrossRef]
36. Đurasević, M.; Đumić, M. Designing relocation rules with genetic programming for the container relocation problem with multiple bays and container groups. *Appl. Soft Comput.* **2024**, *150*, 111104. [CrossRef]
37. Lehnfeld, J.; Knust, S. Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *Eur. J. Oper. Res.* **2014**, *239*, 297–312. [CrossRef]
38. Zajac, M.; Rožić, T.; Naletina, D. Determining the Probability of Unproductive Manipulations in Inland Intermodal Terminal Operations. *Promet-Traffic Transp.* **2023**, *35*, 299–314. [CrossRef]
39. Zhao, W.; Goodchild, A.V. The impact of truck arrival information on container terminal rehandling. *Transp. Res. Part E Logist. Transp. Rev.* **2010**, *46*, 327–343. [CrossRef]