ISSN 1648-4142 / eISSN 1648-3480

VILNIUS TECH
Vilnius Gediminas Technical University

TRANSPORT

2024
Volume 39
Issue 2
Pages 146–160
https://doi.org/10.3846/transport.2024.21668

Original Article

A THREE-STAGE HEURISTIC FOR OPTIMIZING CONTAINER RELOCATIONS IN MARITIME CONTAINER TERMINALS

Qianwen ZHU[1], Bo JIN[2]✉

[1] *Dept of Management Sciences, City University of Hong Kong, Hong Kong, China*
[2] *College of Management, Shenzhen University, Shenzhen, China*

Highlights:

- the approach takes into consideration multiple containers simultaneously when optimizing relocations in maritime container terminals;
- we propose a well-designed three-stage heuristic (3SH) that examines and arranges containers in different stacks, overcoming the deficiencies of previous methods;
- extensive numerical experiments demonstrate that the proposed heuristic outperforms cutting-edge heuristics;
- the introduced heuristic performs particularly well on large-scale problem instances, which is crucial for practical applications.

Article History:

- submitted 8 May 2023;
- resubmitted 5 September 2023, 14 October 2023;
- accepted 19 November 2023

Abstract. The Container Relocation Problem (CRP) is one of the most important optimization problems in maritime container terminals. The objective is to minimize the number of relocation operations for retrieving containers in a sequence. If the container to be retrieved next is not at the top of a stack, unproductive relocations have to be carried out. Due to the large number of containers handled by busy terminals, a slight reduction in relocation rates can result in significant savings in operating costs. Most of the existing heuristics make relocation decisions for the blocking containers one by one, based on simple indicators. In this article, we propose a Three-Stage Heuristic (3SH) that extends the decision horizon to multiple containers to achieve a higher-quality solution. Computational experiments are conducted on 3 sets of benchmark instances, and the results show that the proposed heuristic outperforms the state-of-the-art heuristics documented in the research literature.

Keywords: logistics, optimization, maritime container terminal, container relocation problem, heuristic.

✉Corresponding author. E-mail: *jinbo@szu.edu.cn*

Notations

Variables:

S – number of stacks;
H – maximum height limit;
N – number of containers.

Abbreviations:

3SH – three-stage heuristic;
BRP – blocks relocation problem;
CRP – container relocation problem;
IP – integer programming;
LIFO – last-in, first-out;
NP – nondeterministic polynomial;
VRH – virtual relocation heuristic;
VRI – virtual relocation index.

1. Introduction

Economic globalization poses great challenges to the ocean shipping industry. The efficiency of container transport, which plays a key role in the maritime logistics system, is in need of advancement and streamlining. Importantly, maritime container terminals act as the hub nodes that connect sea and land container transport by providing temporary storage space for container transshipment. An efficient operation process at a terminal can reduce equipment consumption, alleviate traffic congestion, and reduce vessel turnaround time, thus increasing the competitiveness of the port.

A maritime container terminal is a complex system that generally consists of 3 major functional areas, as shown in Figure 1. The *quayside* is an area where vessels can berth, the *landside* is an area for trucks or trains to handover

containers, and a *yard* is the main area where containers are stored. In the yard, containers are piled up vertically to form a *stack*, and several stacks in a row form a *bay*. A *block* is a parallel group of consecutive bays, and the yard consists of a set of such blocks. The operations in the yard include *unloading/loading* containers from/to the yard and *pre-marshalling* containers within the yard (Lehnfeld, Knust 2014). In this article, we focus on the unloading operations, i.e., retrieving containers from the yard.

Let us consider a bay consisting of S stacks indexed from 1 to S, whose maximum height is limited to H. The tiers in the bay are indexed from 1 to H from bottom to top, and a specified location indexed by its stack and tier is called a *slot*. There are in total N containers initially stored in the bay, and each container is assigned a unique integer value from 1 to N to indicate its retrieval priority. Here, a smaller priority value indicates an earlier retrieval order. The objective of the problem is to minimize the total number of relocations needed to retrieve all containers in the bay according to their priority values, i.e., from 1 to N.

If the containers to be retrieved earlier are all placed on top of those to be retrieved later, then the entire retrieval process can be completed at ease. However, an optimal stacking order is not always the case in reality, because containers arrive at the terminal at random and most of the time, terminal operators do not have enough time to rearrange these containers after the exact retrieval sequence is available. During the retrieval process, the *target container* is defined as the container to be retrieved next, and the stack it is located in is called the *source stack*. If the target container is not at the top of the source stack, then all *blocking containers* placed above it have to be relocated to other stacks 1st. A decision has to be made for each blocking container to choose the best *destination stack* to be relocated to. Since relocation operations are valueless and time-consuming, the total number of relocations needs to be minimized to raise the efficiency of the retrieval process. This optimization problem is known as the CRP or the BRP in the literature, which has been proven to be NP-hard (Caserta *et al.* 2012).

Figure 2 presents an example depicting the 1st few steps in solving a CRP instance. 1st, container 1 is immediately removed from the bay since it is retrievable, i.e., it is at the top of stack 3. Then, in order to make container 2 retrievable, containers 3 and 13 are relocated from stack 2 to stacks 3 and 4, respectively. It is assumed that relocations only occur to the blocking containers atop the target container, which is a common practice in many terminals. The relocation problem with this restriction is also known as the restricted CRP in the literature.

In this article, we propose a method called the 3SH to address the CRP, which extends the decision horizon to deal with multiple blocking containers simultaneously. Experimental results show that it outperforms the state-of-the-art heuristics on 3 sets of benchmark instances. The remainder of this article is organized as follows. Section 1 – introduction. Section 2 reviews related works in the literature. Sections 3 describes the proposed method, and Section 4 presents the experimental results. Lastly, Section 5 concludes the article.

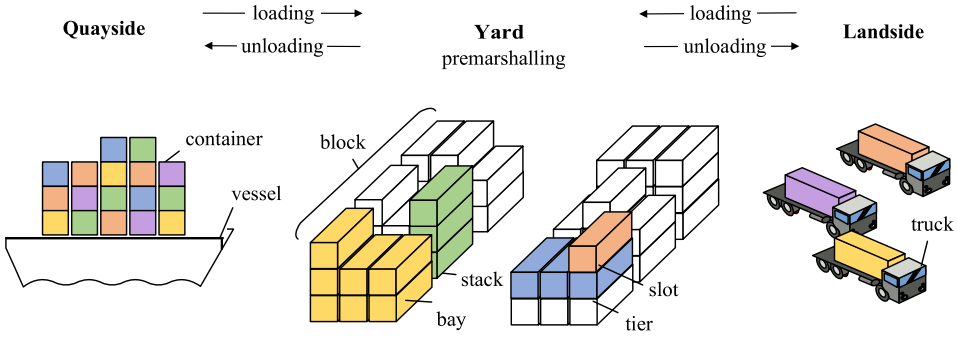

Figure 1. Structure of a maritime container terminal

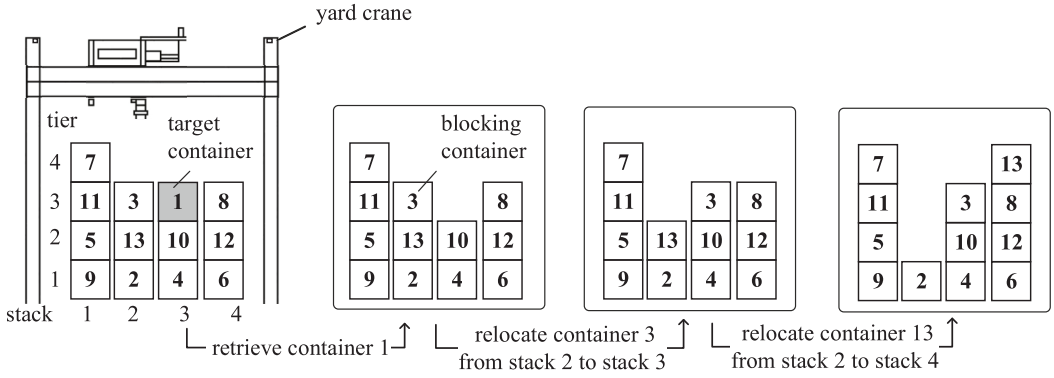

Figure 2. Example of retrieval and relocation operations

2. Literature review

As shown in Figure 1, there are 3 types of container handling processes in a container yard. There is the loading process in which incoming containers delivered by vessels/trucks need to be stored in a designated space, the pre-marshalling process in which the containers already stacked in the yard need to be rearranged to reduce the overall retrieval time in the upcoming unloading process, and the unloading process in which some containers in the yard need to be retrieved in a predetermined sequence for further transportation. Lehnfeld & Knust (2014) and Carlo *et al.* (2014) provided comprehensive surveys on existing yard optimization problems based on this classification scheme. The container stacking problem (e.g., Kim *et al.* 2000) occurs in the loading process to determine the locations for stacking incoming containers so as to minimize the total number of future relocations. For the same purpose, the container pre-marshalling problem (e.g., Lee, Hsu 2007) is raised in the pre-marshalling process. Finally, the CRP arises in the unloading process.

To the best of our knowledge, Kim & Hong (2006) was the 1st to study the CRP. They formally introduced the problem in the form of the restricted version, and they also proposed a rule-based heuristic and a branch-and-bound algorithm to solve it. Caserta *et al.* (2012) and Zhu *et al.* (2012) were the 1st to extend the CRP to the unrestricted version. A survey of optimization methods for the CRP can be found in Lersteau & Shen (2022). Caserta *et al.* (2012) presented integer linear programming formulations for both the restricted and unrestricted versions, and also designed a simple rule of thumb to solve the problem. Zhu *et al.* (2012) investigated the application of the iterative A* algorithm in solving both versions of the CRP. Since then, a number of studies have been proposed to tackle the restricted version (Jovanovic, Voß, 2014; Ting, Wu 2017; Quispe *et al.* 2018; Bacci *et al.* 2019, 2020; Zhang *et al.* 2020) or the unrestricted version (Petering, Hussein 2013; Jin *et al.* 2015; Tricoire *et al.* 2018; Feillet *et al.* 2019; Jin, Tanaka 2023), and some scholars studied both versions at the same time (Jovanovic *et al.*, 2019). This article is focused on the restricted version.

There have been several IP models proposed for the restricted CRP in the literature. Caserta *et al.* (2012) gave an IP formulation named BRP-II, which was later corrected by Zehendner *et al.* (2015). Galle *et al.* (2018) presented a new integer program called CRP-I, which uses a smart pairwise encoding to describe the relative positions of the containers. Bacci *et al.* (2020) proposed a new IP formulation as well as a branch-and-cut algorithm. Recently, Tanaka & Voß (2022) proposed a novel IP model based on truncated relocation sequences. This model is solved repeatedly with an iteratively expanding set of truncated relocation sequences, until an optimal solution is found. Another branch of exact approaches to the restricted CRP includes branch-and-bound search (Kim, Hong 2006; Expósito-Izquierdo *et al.* 2014, 2015; Ku, Arthanari 2016) and iterative deepening search (Zhu *et al.* 2012; Tanaka, Takii 2016; Quispe *et al.* 2018).

Despite the fact that the above exact approaches are guaranteed to obtain optimal solutions, the computation time required increases dramatically as the instance size increases. Even the most efficient exact algorithm (Tanaka, Voß 2022) requires hours to solve an instance with 10 stacks and 10 tiers. As a result, a substantial amount of research has been dedicated to the design of heuristic approaches that can yield high-quality solutions in a relatively short computation time. The existing heuristic approaches can be broadly classified into rule-based heuristics and meta-heuristics depending on the complexity of the algorithmic design. Rule-based heuristics are greedy algorithms that repeatedly determine and perform the next relocation until the bay becomes empty. The existing rule-based heuristics for the restricted CRP include the heuristic based on the expected number of additional relocations (Kim, Hong 2006), the reshuffle index heuristic (Murty *et al.* 2005), the Min–Max heuristic (Caserta *et al.* 2012), the PR4 heuristic (Zhu *et al.* 2012), the chain heuristic (Jovanovic, Voß 2014), the Greedy1 heuristic (Ünlüyurt, Aydın 2012), the group assignment heuristic (Wu, Ting 2012), the VRH (Ting, Wu 2017), and the machine learning-driven upper bound method (Zhang *et al.* 2020). The existing meta-heuristics include beam search (Ting, Wu 2017; Zhang *et al.* 2020) and its bounded version (Bacci *et al.* 2019), corridor method (Caserta *et al.* 2011), ant colony algorithm (Jovanovic *et al.* 2019), and genetic algorithm (Maglić *et al.* 2020).

In this article, we are concerned with rule-based heuristics. Murty *et al.* (2005) designed the reshuffle index heuristic, which moves the blocking container to the stack where it will block the least number of containers. It accounts for all containers causing additional relocations, without giving special attention to the container with the smallest priority value. This poses a major shortcoming because other containers are likely to be moved anyway when retrieving the container with the smallest priority value, making their inclusion irrelevant. The Min–Max heuristic by Caserta *et al.* (2012) corrected this by focusing on the container with the smallest priority value in each stack. It 1st aims to place the blocking container in a stack that will not cause further relocations. 2nd aims to choose a stack whose smallest priority value closely matches the priority of the blocking container. The PR4 heuristic by Zhu *et al.* (2012) modified the Min–Max heuristic by intricately handling a special case. When moving the blocking container to any other stack inevitably causes additional relocations, the Min–Max heuristic chooses the stack with the largest priority value. If the stack selected by the Min–Max heuristic has only one available slot, the PR4 heuristic instead chooses the stack with the 2nd-largest priority value. The aforementioned heuristics make relocation decisions for just one blocking container at a time. However, a more comprehensive approach that considers multiple containers simultaneously has the potential to yield better solutions. This line of research is exemplified by the chain heuristic (Jovanovic, Voß 2014) and the VRH (Ting, Wu 2017), and is further extended in our study.

3. 3SH

3.1. Motivation of the proposed method

A heuristic for the restricted CRP describes a rule that decides the destination stacks for the blocking containers placed above the target container. Most existing heuristics make relocation decisions for these containers, one by one, from top to bottom; these individual decisions do not consider the overall impact of each decision. Generally speaking, it is beneficial to take multiple containers into account and make a more thoughtful decision by evaluating the overall cost of relocating them. 2 existing heuristics in the literature have adopted this idea. One is the chain heuristic proposed by Jovanovic & Voß (2014). When deciding the destination stack for the current container to be relocated, the chain heuristic checks whether it is better to reserve a potential stack for relocation of the subsequent container. The major limitation of the chain heuristic is that it considers only 2 containers in a row. The other heuristic considering multiple containers is the VRH proposed by Ting & Wu (2017), which extends the decision horizon to all blocking containers above the target container. However, VRH deteriorates in the case where there is only one container in the decision horizon.

In this section, we introduce a new heuristic called 3SH, which combines the advantages of the 2 above-mentioned heuristics while alleviating their disadvantages. Similar to VRH, 3SH determines the destination stacks for all blocking containers above the target container simultaneously. As the name implies, all the decisions will be completed in 3 stages. The 1st 2 stages are derived from VRH with effective enhancements. In the 1st stage, containers that do not cause additional relocations are assigned to suitable destination stacks 1st, while the rest will be assigned in the 2nd stage. Finally, in the last stage, the complete assignment identified in the previous stages is adjusted to further improve the solution. Figure 3 provides an overview of the steps completed at each stage.

3.2. 1st stage: compute a partial assignment

Relocating a container onto a stack whose priority is higher than the relocated container results in an additional relocation, since the relocated container has to be relocated again. Here, the priority of a stack is defined by the smallest priority value among all the containers in it; if the stack is empty, its priority value is set to $N+1$. The goal of the 1st stage is to assign as many blocking containers as possible without causing any additional relocations.

Let us start by recalling the 1st phase of VRH. In the 1st phase of VRH, the blocking containers are processed in descending order of their priority values, which we call the priority order. For each container c, if there exists at least one stack such that: (1) it differs from the source stack, (2) it is not fully occupied, (3) its priority value is larger than c, and (4) putting c onto the top of this stack will not violate the LIFO constraint, then container c is virtually moved to the stack with the smallest priority value; otherwise, container c remains unassigned. Conditions (1) and (2) are necessary for a feasible relocation, condition (3) ensures that container c would be non-blocking after the relocation, and condition (4) is mandatory to prevent from violating the natural relocation order.

However, such a priority order is not always better than the natural order from top to bottom. Based on the priority order, after container c chooses its destination stack, this chosen stack will be excluded for all containers placed above c due to the LIFO constraint. Obviously, this exclusion may make the priority order worse than the natural order. Figure 4 demonstrates an example, in which containers 11, 8, and 15 are the blocking containers from top to bottom, and there is only one empty stack with a sufficiently large priority value of 16. The circled numbers next to the containers represent the respective decision sequence in each order. In the priority order, only container 15 can be relocated to this stack without causing additional relocations, while in the natural order, containers 11 and 8 can be relocated to this stack without causing additional relocations.

To generate a better partial assignment in the 1st stage, we examine both the priority order and the natural order and choose the one that assigns more containers without causing additional relocations. Note that in the natural order, the LIFO constraint is automatically satisfied, so we do not need to check it during the decision process.

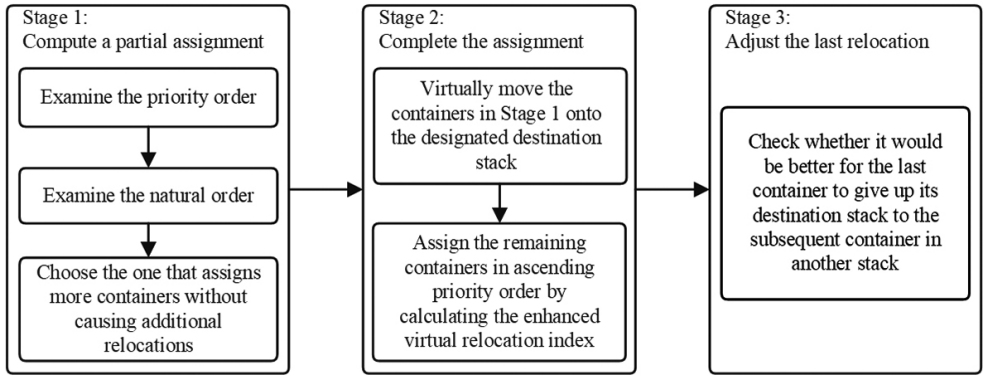

Figure 3. Flowchart of the 3SH

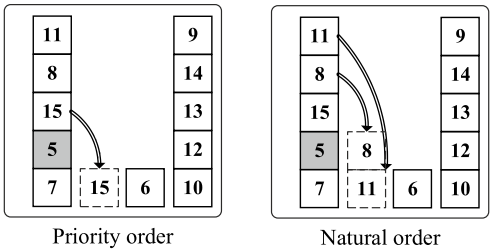

Figure 4. Example for comparing the priority order and the natural order in the 1st stage

3.3. 2nd stage: complete the assignment

The 1st stage obtains a partial assignment for the containers that can be relocated without causing any additional relocations. For the sake of simplicity, we consider these containers to have been virtually moved onto the designated destination stack. The task of the 2nd stage is to assign the remaining containers after the 1st stage, which will cause additional relocations. Due to the LIFO constraint, these containers may have to be inserted into a middle position between those that have been virtually relocated. For each blocking container c considered in the 2nd stage, the specific position to be inserted to each candidate stack s is unique due to the LIFO constraint. In this way, the containers in stack s, including the original containers placed in it and those having been virtually relocated to it, can be divided into 2 parts. Let U denote the number of upper containers whose priority values are larger than c, and let u and l denote the smallest priority values within the upper and lower-parts, respectively. The VRH processes the remaining containers in ascending order of priority values, and for each container$c$, it calculates the so-called VRI for each candidate stack s as follows:

$$\text{original VRI} = \begin{cases} \min(l-c, c-u), & \text{if } U \le 1; \\ c-u-N, & \text{otherwise,} \end{cases}$$

where: the differences between container c and the lower and upper-parts, i.e., $l - c$ and $c - u$, are referred to as the lower-part difference and upper-part difference, respectively. If no more than one additional relocation exists (i.e., $U \le 1$), both the lower-part and upper-part differences are considered. Otherwise, only the upper-part difference is considered with a penalty of $-N$ added to reduce the weight of the candidate stack. Finally, the stack with the largest VRI is selected as the destination stack for relocating container c.

In the proposed 3SH, we enhance the VRI by dividing the 1st case into 3 finer cases as follows:

$$\text{enhanced VRI} = \begin{cases} l-c, & \text{if } U=0 \text{ and } E>1; \\ -(l-c)-2N, & \text{if } U=0 \text{ and } E=1; \\ \min(l-c, c-u), & \text{if } U=1; \\ c-u-N, & \text{otherwise,} \end{cases}$$

where: E represents the number of empty slots in the considered candidate stack s. The situations of $U = 1$ and $U > 1$ are the same as that in VRH, while for $U = 0$, we in addition consider 2 cases depending on E. Note that $U = 0$ implies that either the upper-part is empty or all the upper containers have smaller priority values than c. If the considered candidate stack s has more than one empty slot (i.e., $E > 1$), we only consider the lower-part difference. Otherwise, we consider the opposite of the lower-part difference with a penalty of $-2N$ added to reduce the weight of the candidate stack. The purpose of using the opposite of the lower-part difference in the case of $E = 1$ is to prevent wasting potential candidate stacks with large priority values.

Figure 5 shows an example demonstrating how the enhanced VRI can improve the decision. With the original VRI, container 15 chooses stack 4 as its destination stack, making stack 4 fully occupied. Subsequently, after container 5 is retrieved, container 6 becomes the target container, and container 9 inevitably causes an additional relocation. However, in the proposed 3SH, container 15 chooses stack 2 as its destination stack according to the enhanced VRI, because the scores for stacks 2, 3, and 4 are −7, −9, and −25, respectively. Thus, stack 4 will be reserved to better accommodate other blocking containers. Specifically in the example shown, container 9 will be relocated to the last empty slot of stack 4, with no additional relocation caused.

3.4. Last stage: adjust the last relocation

After the previous 2 stages, all blocking containers above the target container have been assigned to the proper destination stacks. The last stage attempts to further improve the obtained arrangement. We denote the bottommost blocking container above the current target container as c and the topmost blocking container above the next target container as d. After all the containers above container c have been relocated to their designated destination stacks, we check whether it would be better for container c to give up its destination stack s to container d.

2 situations will be compared in a similar way to that in research by Jovanovic & Voß (2014). In the 1st situation, container c will be relocated to stack s as planned, and the score D_1 is computed as the priority of stack s minus c. After that, container d chooses its destination stack using the Min–Max rule (Caserta *et al.* 2012), and the score D_2 is equal to the priority of the chosen stack minus d. In the 2nd situation, container c re-selects its destination stack using the Min–Max rule with the exclusion of stack s, and the score R_1 is computed as the chosen stack minus c. With regard to container d, the score R_2 is computed as the priority of stack s minus d. The 2nd situation is considered to be a better choice if the following conditions are satisfied: (1) $D_1 > R_2 > 0$ and (2) $D_2R_1 > 0$. The 1st condition implies that both containers c and d will not cause additional relocations if they are relocated to stack s, and that the difference of priority between stack s and container c is larger than that of d. The 2nd condition implies that the relocations specified by D_2 and R_1 both cause additional relocations, or neither of them cause additional relocations.

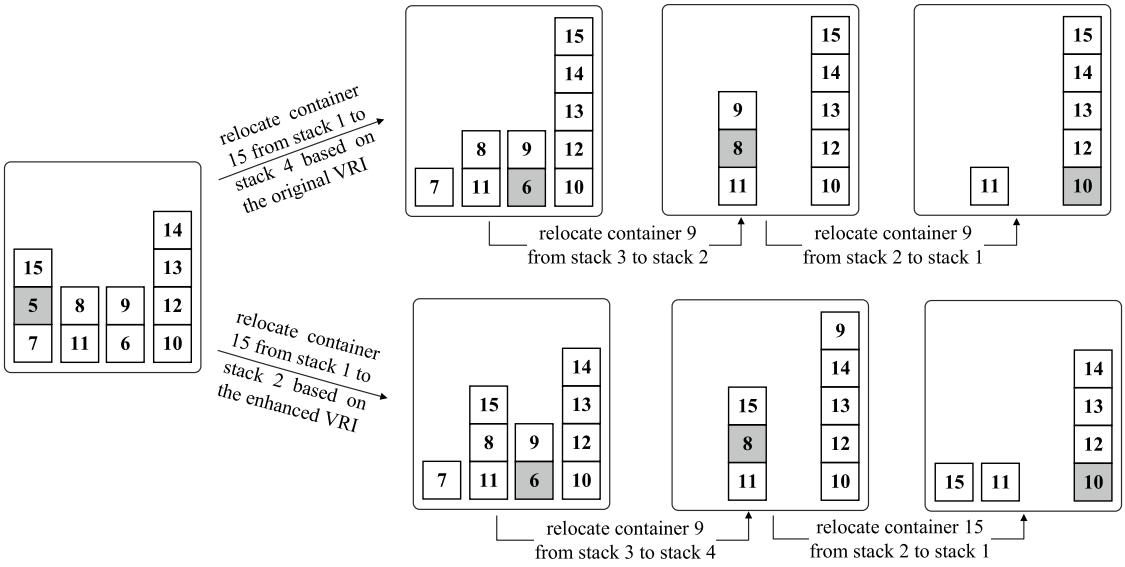

Figure 5. Example for comparing the original VRI and the enhanced VRI in the 2nd stage

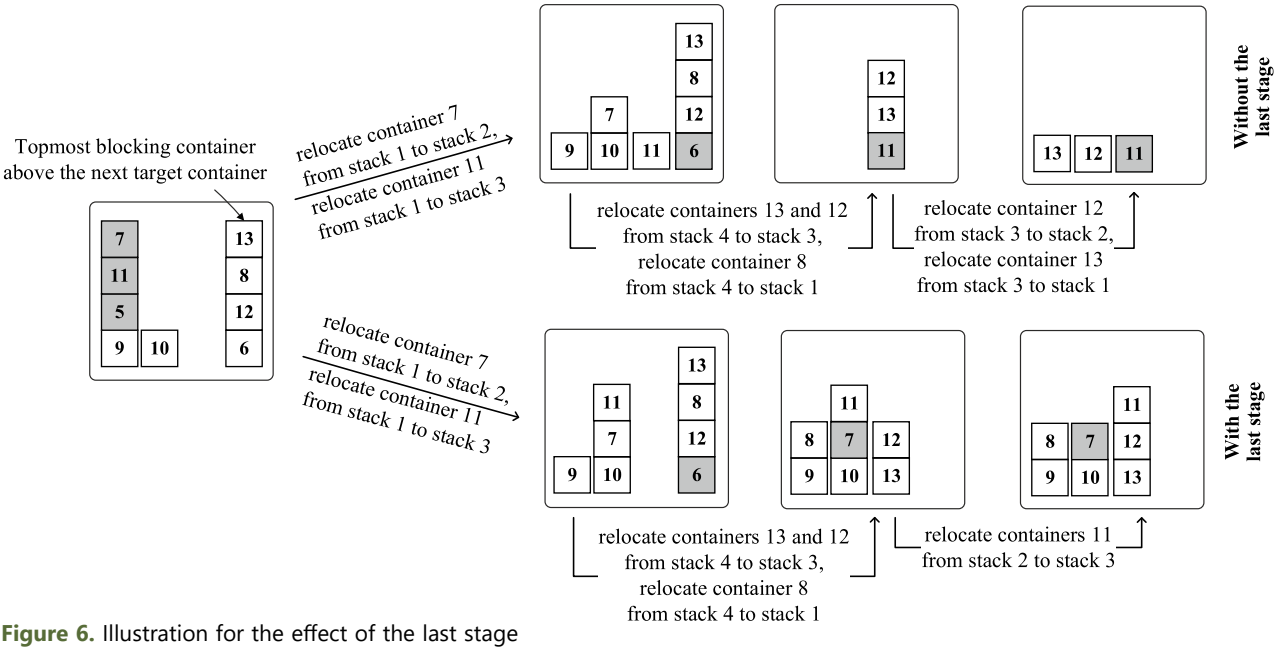

Figure 6. Illustration for the effect of the last stage

Figure 6 demonstrates an example in which the last stage reduces one relocation. The last blocking container to be relocated for retrieving container 5 is container 11. The last stage decides that stack 3 should be reserved for container 13, which is the topmost blocking container above the next target container 6. Although by making container 11 cause an additional relocation, it prevents container 13 from causing any additional relocation and yields a better place for receiving the other blocking containers that follow.

3.5. Merge consecutive decision horizons

Roughly speaking, the decision horizon of the proposed 3SH is the set of all blocking containers above the current target container. However, in some special cases, consecutive decision horizons could be merged to further improve the solution quality. Specifically, if the next target container $c + 1$ is at a lower tier in the same stack of the current target container c, we can simply omit the current target container c from the bay and make more comprehensive decisions for the merged set of blocking containers above container $c + 1$.

4. Computational experiments

4.1. Experimental settings

To evaluate the performance of the proposed 3SH, 3 sets of instances, which are generated by Wu & Ting (2010), Caserta *et al.* (2011), and Zhu *et al.* (2012), are tested in the experiments. 5 existing heuristic approaches in the lit-

erature are used for comparison, namely, (1) Min–Max by Caserta *et al.* (2012), (2) PR4 by Zhu *et al.* (2012), (3) Chain and (4) ChainF by Jovanovic & Voß (2014), and (5) VRH by Ting & Wu (2017). Among them, ChainF and VRH are the most promising ones, which have been used as an important component in the meta-heuristic approaches by Bacci *et al.* (2019) and Zhang *et al.* (2020), respectively.

All heuristic algorithms are programmed in *Java* and run on a desktop computer with an *Intel Core i5 1.4 GHz CPU and 16 GB of RAM*. The exact algorithm for calculating the optimal values for the instances from Wu & Ting (2010) is programmed in *C* and run on the same computer under a time limit of one minute for each instance. The source code of the exact algorithm is available online on *https://github.com/jinboszu/rcrp-idbb/*. Optimal values and computation times for the instances from Caserta *et al.* (2011) and Zhu *et al.* (2012) are directly obtained from Tanaka & Voß (2022). Their experiments were conducted on a desktop computer equipped with an *Intel Core i9-9900K 3.6 GHz CPU and 64 GB of RAM* under a time limit of one hour for each instance.

For each test dataset, we present a table containing the detailed results of all 6 compared heuristics and the optimal values of the exact algorithm, accompanied by another table containing the computation times of all compared heuristics and the exact algorithm. The instances in each dataset are grouped by the maximum height H and the number of stacks S. For each heuristic, the average number of relocations and the average computation time in milliseconds for each group of instances are reported. A value in bold indicates that the corresponding heuristic provides the best result for the corresponding group of instances. Lastly, we conduct an ablation study of 3SH to further assess the effect of the enhancements introduced in Section 3.

4.2. Results for the instances from Wu & Ting (2010)

The 1st experiment is carried out on the instances generated by Wu & Ting (2010). This dataset is divided into 48 groups, where each group is characterized by the maximum height $H \in \{3, \ldots, 8\}$ and the number of stacks $S \in \{3, ..., 10\}$, and the number of containers N in each group is defined by $N = (S-1) \cdot H + 1$. There are 40 instances in each group, and hence the total number of instances in the dataset is 1920. Table 1 shows the computational results of all 6 of the above-mentioned heuristics and the optimal values. The comparison indicates that the proposed 3SH performs the best among all the compared heuristics, winning 35 out of a total of 48 groups. Additionally, 3SH also performs the best in terms of average performance, with an optimal gap of 7.29%. The reason is that our method utilizes more future blocking container information than the other heuristics to help optimize the relocations. Granted that the enhancements of 3SH do not always improve the solution quality compared to VRH, we find that the groups on which 3SH does not provide the best results are mostly small-scale, and that the advantage of 3SH over the other heuristics becomes more significant as the instance size increases. Table 2 presents the computation times of all 6 compared heuristics and the exact algorithm. Among the heuristics, 3SH takes shorter average time compared to VRH. The exact algorithm performs faster than the heuristics for small-scale instances where $H < 5$ or $S < 5$, but is significantly slower for large-scale instances.

4.3. Results for the instances from Caserta *et al.* (2011)

Next, we assess the performance of 3SH on the instances provided by Caserta *et al.* (2011). This dataset includes 21 groups of instances, where each group consists of 40 randomly generated instances of the same instance size. In the original dataset, all stacks in each instance have the same initial height K, but the maximum height H is not given. A widely accepted practice is to add 2 empty tiers above the containers in each instance, i.e., $H = K + 2$. Table 3 summarizes the computational results of all 6 compared heuristics and provides the optimal values. Similar to the results in the previous experiment, 3SH takes a lead over the other heuristics by winning 15 out of a total of 21 groups. Also, 3SH achieves the best average results with an optimal gap of 10.67%. It is also interesting to observe from the comparison that the heuristics based on simpler rules (i.e., Min–Max, PR4, Chain, and ChainF) turn out to perform better on smaller instances, for example, those with $K = 3$. Table 4 lists the computation times of all 6 compared heuristics and the exact algorithm. Similar to the results in the above experiment, in terms of average computation time, PR4 executes the fastest and 3SH runs faster than VRH, and the exact algorithm takes much more time in large-scale scenarios.

4.4. Results for the instances from Zhu *et al.* (2012)

The instances from Zhu *et al.* (2012) are characterized by 3 parameters: the maximum height $H \in \{3, ..., 7\}$, the number of stacks $S \in \{6, \ldots, 10\}$, and the number of containers $N \in \{SH - H, ..., SH - 1\}$. There are in total 125 combinations of (H, S, N), and 100 instances are randomly generated for each combination. This dataset represents the situation where the bay is relatively fully occupied, which increases the problem-solving difficulties. Table 5 shows the computational results of the 6 compared heuristics and the optimal results for the instances grouped by H and S. The proposed 3SH outperforms the other competitors remarkably, winning 23 out of a total of 25 groups. The best results for the rest 2 groups are given by VRH, while 3SH comes in 2nd only by a small margin. 3SH has the best average performance among all methods with an optimal gap of 6.48%. We deduce the reason for the superior performance of VRH and 3SH as follows. The density of instances in this experiment is considerably higher than that in the previous 2 experiments, so it is more likely to have more blocking containers during the retrieval process.

Table 1. Computational results for the instances from Wu & Ting (2010)

H	*S*	*N*	#inst	Min–Max	PR4	Chain	ChainF	VRH	3SH	Exact
3	3	7	40	3.43	3.38	3.42	3.38	3.37	**3.35**	3.30
3	4	10	40	4.95	4.88	4.95	4.95	**4.87**	4.90	4.85
3	5	13	40	**5.75**	5.78	**5.75**	5.80	**5.75**	5.78	5.75
3	6	16	40	7.73	**7.70**	7.72	7.85	**7.70**	**7.70**	7.65
3	7	19	40	9.03	**9.00**	9.05	9.17	9.02	9.03	8.95
3	8	22	40	9.88	9.75	9.87	9.75	**9.72**	9.73	9.73
3	9	25	40	11.63	11.58	11.60	11.57	**11.55**	**11.55**	11.45
3	10	28	40	12.03	**11.93**	12.02	11.97	11.95	**11.93**	11.88
4	3	9	40	5.83	5.73	5.82	5.95	**5.70**	5.73	5.68
4	4	13	40	8.75	8.55	8.72	8.57	**8.52**	8.55	8.43
4	5	17	40	11.40	11.43	11.35	11.40	**11.30**	11.43	10.98
4	6	21	40	12.68	12.55	12.55	12.60	12.50	**12.38**	12.03
4	7	25	40	16.35	16.03	16.45	16.07	**15.97**	15.98	15.48
4	8	29	40	18.73	18.58	18.52	18.52	**18.30**	**18.30**	17.95
4	9	33	40	19.85	19.83	19.72	19.87	19.60	**19.53**	19.15
4	10	37	40	23.60	23.23	23.32	23.25	23.20	**23.08**	22.35
5	3	11	40	9.10	**8.48**	9.10	8.70	8.50	8.53	8.40
5	4	16	40	13.13	12.70	13.02	13.17	**12.60**	12.63	12.25
5	5	21	40	17.08	16.45	16.95	16.65	16.20	**16.13**	15.58
5	6	26	40	20.60	20.50	20.42	20.52	20.22	**20.05**	19.33
5	7	31	40	23.15	23.05	22.92	22.70	22.60	**22.45**	21.35
5	8	36	40	27.68	27.05	27.37	26.82	26.67	**26.43**	25.40
5	9	41	40	31.00	30.60	30.35	30.55	29.70	**29.65**	28.65
5	10	46	40	34.50	33.83	33.80	33.45	33.35	**33.05**	31.70
6	3	13	40	12.98	12.33	12.77	12.30	12.20	**12.18**	11.50
6	4	19	40	17.15	16.88	17.05	16.92	**16.52**	16.55	15.63
6	5	25	40	23.93	23.33	23.52	23.57	22.55	**22.35**	21.05
6	6	31	40	29.03	28.63	28.77	28.57	28.22	**27.78**	25.98
6	7	37	40	34.40	33.65	34.07	34.07	33.15	**32.68**	30.75
6	8	43	40	40.48	39.60	39.65	38.90	38.97	**38.50**	35.68
6	9	49	40	44.90	44.20	44.50	43.30	42.57	**42.45**	39.68
6	10	55	40	48.80	48.35	47.77	47.45	46.95	**46.10**	43.60
7	3	15	40	16.63	16.43	16.25	16.40	16.27	**16.00**	15.03
7	4	22	40	26.38	25.60	25.97	25.62	25.17	**24.40**	22.60
7	5	29	40	31.93	31.35	31.15	31.42	30.42	**30.33**	27.53
7	6	36	40	40.43	39.43	39.22	38.65	38.20	**37.78**	34.40
7	7	43	40	45.63	44.03	44.92	44.27	43.35	**43.15**	38.95
7	8	50	40	52.10	51.73	51.15	50.82	49.00	**48.50**	44.48
7	9	57	40	59.88	58.95	57.92	57.35	**56.00**	56.03	50.80
7	10	64	40	65.73	64.98	64.37	63.60	62.55	**61.35**	56.58
8	3	17	40	22.03	21.60	21.52	21.20	20.57	**20.35**	18.63
8	4	25	40	33.68	31.90	33.05	32.82	**30.95**	30.98	27.70
8	5	33	40	43.08	42.10	42.57	42.00	39.52	**38.58**	35.80
8	6	41	40	52.20	51.18	50.87	50.25	48.67	**47.88**	42.18
8	7	49	40	59.03	58.48	57.97	56.32	54.40	**54.35**	48.83
8	8	57	40	66.38	65.48	65.12	64.55	62.85	**61.95**	55.78
8	9	65	40	76.33	74.83	75.05	73.50	71.92	**70.75**	65.60
8	10	73	40	83.90	82.58	82.02	79.35	78.30	**77.15**	70.28
Summary			1920	28.85	28.33	28.37	28.05	27.46	**27.21**	25.36

Note: a value in bold indicates that the corresponding heuristic provides the best result for the corresponding group of instances.

Table 2. Computation times for the instances from Wu & Ting (2010)

H	*S*	*N*	#inst	Min–Max	PR4	Chain	ChainF	VRH	3SH	Exact
3	3	7	40	0.43	**0.33**	0.53	0.42	0.84	0.60	0.02
3	4	10	40	0.25	**0.21**	0.31	0.37	0.46	0.41	0.02
3	5	13	40	**0.23**	0.25	0.35	0.31	0.45	0.28	0.02
3	6	16	40	0.33	**0.29**	0.55	0.47	0.34	0.50	0.03
3	7	19	40	0.29	**0.22**	0.25	0.25	0.40	0.25	0.03
3	8	22	40	0.21	**0.18**	0.27	0.25	0.45	0.26	0.03
3	9	25	40	**0.18**	0.19	0.23	0.31	0.33	0.25	0.05
3	10	28	40	0.25	**0.17**	0.34	0.20	0.41	0.27	0.06
4	3	9	40	0.21	**0.16**	**0.16**	0.18	0.22	0.28	0.02
4	4	13	40	0.15	**0.14**	0.19	0.17	0.54	0.30	0.03
4	5	17	40	0.16	**0.13**	0.18	0.22	0.42	0.19	0.05
4	6	21	40	0.19	**0.16**	0.29	0.20	0.34	0.20	0.06
4	7	25	40	0.17	**0.15**	0.17	0.22	0.30	0.30	0.13
4	8	29	40	0.17	**0.14**	0.19	0.30	0.46	0.34	0.14
4	9	33	40	**0.16**	0.17	0.17	0.27	0.32	0.23	0.38
4	10	37	40	0.20	0.16	**0.14**	0.24	0.43	0.24	0.27
5	3	11	40	**0.10**	0.13	0.14	0.11	0.21	0.24	0.03
5	4	16	40	**0.11**	0.16	0.17	0.12	0.29	0.29	0.05
5	5	21	40	**0.11**	0.21	0.27	0.12	0.39	0.21	0.13
5	6	26	40	0.16	**0.11**	0.22	0.15	0.38	0.24	0.32
5	7	31	40	0.17	**0.10**	0.21	0.16	0.36	0.22	0.49
5	8	36	40	0.19	**0.09**	0.13	0.17	0.34	0.24	1.97
5	9	41	40	0.20	**0.12**	0.14	0.16	0.25	0.23	12.91
5	10	46	40	0.21	**0.14**	0.15	0.18	0.27	0.26	342.38
6	3	13	40	**0.10**	**0.10**	0.12	0.16	0.13	0.16	0.05
6	4	19	40	0.12	**0.11**	**0.11**	0.20	0.16	0.23	0.10
6	5	25	40	**0.11**	0.13	0.13	0.13	0.15	0.17	0.80
6	6	31	40	**0.12**	0.20	0.17	**0.12**	0.16	0.26	4.38
6	7	37	40	**0.14**	0.18	0.16	0.16	0.16	0.37	19.95
6	8	43	40	**0.16**	0.19	**0.16**	0.18	**0.16**	0.34	150.68
6	9	49	40	0.14	**0.12**	0.17	0.19	0.17	0.35	4765.07
6	10	55	40	0.15	**0.13**	0.18	0.21	0.24	0.36	5618.77
7	3	15	40	0.13	**0.09**	0.18	0.11	0.16	0.14	0.10
7	4	22	40	0.18	**0.13**	0.21	**0.13**	0.30	0.17	0.64
7	5	29	40	0.19	**0.11**	0.22	0.14	0.30	0.18	9.63
7	6	36	40	0.25	0.14	0.19	0.20	**0.13**	0.22	359.50
7	7	43	40	0.25	**0.14**	**0.14**	0.40	0.30	0.21	1401.14
7	8	50	40	0.29	**0.15**	**0.15**	0.36	0.25	0.23	4250.03
7	9	57	40	0.36	**0.14**	0.16	0.17	0.23	0.38	17242.84
7	10	64	40	**0.15**	0.34	0.20	0.19	0.27	0.54	32143.32
8	3	17	40	**0.09**	0.11	0.10	0.59	0.13	0.17	0.21
8	4	25	40	0.12	**0.10**	0.12	0.30	0.15	0.26	3.08
8	5	33	40	0.13	**0.10**	0.13	0.21	0.18	0.34	501.56
8	6	41	40	0.13	**0.11**	0.14	0.21	0.20	0.57	3826.94
8	7	49	40	**0.14**	0.15	**0.14**	0.23	0.20	0.47	21664.32
8	8	57	40	**0.13**	0.15	0.16	0.26	0.27	0.41	32384.26
8	9	65	40	0.18	**0.13**	0.16	0.24	0.23	0.31	53675.46
8	10	73	40	**0.16**	0.18	0.18	0.43	0.39	0.29	52462.15
Summary			1920	0.18	**0.16**	0.20	0.23	0.30	0.29	4809.26

Note: a value in bold indicates that the corresponding heuristic provides the best result for the corresponding group of instances.

Table 3. Computational results for the instances from Caserta *et al.* (2011)

H	*S*	*N*	#inst	Min–Max	PR4	Chain	ChainF	VRH	3SH	Exact
5	3	9	40	**5.08**	**5.08**	**5.08**	5.15	5.15	5.13	5.00
5	4	12	40	**6.30**	**6.30**	6.35	6.33	6.35	6.35	6.18
5	5	15	40	**7.05**	**7.05**	**7.05**	7.08	**7.05**	**7.05**	7.03
5	6	18	40	8.45	8.45	**8.43**	8.45	8.48	8.45	8.40
5	7	21	40	9.33	9.33	**9.30**	**9.30**	9.33	9.33	9.28
5	8	24	40	10.73	10.73	**10.68**	**10.68**	10.73	**10.68**	10.65
6	4	16	40	10.98	10.93	10.80	10.93	10.80	**10.73**	10.20
6	5	20	40	13.55	13.63	13.45	13.63	13.45	**13.40**	12.95
6	6	24	40	14.68	14.60	14.63	14.58	14.60	**14.35**	14.03
6	7	28	40	16.90	16.80	16.88	16.73	16.80	**16.58**	16.13
7	4	20	40	16.75	16.75	16.40	16.65	**16.30**	**16.30**	15.43
7	5	25	40	21.23	20.98	20.55	20.38	20.50	**20.23**	18.85
7	6	30	40	24.25	24.05	23.93	23.98	**23.75**	23.80	22.08
7	7	35	40	26.33	26.35	25.88	25.90	25.80	**25.70**	24.25
7	8	40	40	29.60	29.60	29.05	28.80	28.98	**28.60**	27.70
7	9	45	40	32.35	32.35	32.15	31.98	32.18	**31.83**	30.45
7	10	50	40	35.50	35.30	35.08	**34.58**	34.75	34.70	33.28
8	6	36	40	35.90	35.80	34.98	34.60	34.25	**34.00**	30.88
8	10	60	40	49.85	49.63	49.20	48.98	48.88	**48.60**	45.50
12	6	60	40	101.25	100.35	97.13	94.70	91.38	**90.73**	74.38
12	10	100	40	139.28	138.60	136.18	132.20	127.78	**127.10**	104.75
Summary			840	29.30	29.17	28.72	28.36	27.96	**27.79**	25.11

Note: a value in bold indicates that the corresponding heuristic provides the best result for the corresponding group of instances.

Table 4. Computation times for the instances from Caserta *et al.* (2011)

H	*S*	*N*	#inst	Min–Max	PR4	Chain	ChainF	VRH	3SH	Exact
5	3	9	40	0.25	0.29	**0.25**	0.26	0.30	0.32	<10
5	4	12	40	**0.14**	**0.14**	0.25	0.16	0.21	0.25	<10
5	5	15	40	0.16	**0.15**	0.32	0.17	0.21	0.22	<10
5	6	18	40	0.15	**0.13**	0.15	0.24	0.16	0.17	<10
5	7	21	40	**0.12**	**0.12**	0.14	0.15	0.13	0.16	<10
5	8	24	40	0.12	**0.11**	0.18	0.13	0.22	0.15	<10
6	4	16	40	**0.09**	0.12	0.13	0.15	0.17	0.11	<10
6	5	20	40	0.14	0.15	0.15	**0.12**	0.21	0.13	<10
6	6	24	40	0.16	0.14	0.16	**0.12**	0.13	0.13	<10
6	7	28	40	0.13	0.12	0.12	**0.11**	0.13	0.16	<10
7	4	20	40	0.13	0.13	**0.09**	0.10	0.14	0.22	10
7	5	25	40	**0.08**	0.09	0.10	0.10	0.16	0.13	20
7	6	30	40	**0.11**	**0.11**	**0.11**	0.15	0.15	0.15	20
7	7	35	40	0.14	**0.09**	0.11	0.19	0.15	0.21	20
7	8	40	40	0.13	**0.10**	0.13	0.14	0.13	0.21	40
7	9	45	40	0.14	0.12	0.13	**0.11**	0.33	0.20	40
7	10	50	40	0.13	**0.08**	0.10	0.12	0.42	0.25	100
8	6	36	40	0.13	**0.08**	0.11	0.11	0.41	0.25	270
8	10	60	40	0.14	**0.10**	0.28	**0.10**	0.37	0.21	870
12	6	60	40	**0.27**	0.28	0.31	0.40	0.44	0.35	1214430
12	10	100	40	**0.18**	**0.18**	0.28	0.23	0.78	0.73	1519130
Summary			840	0.14	**0.13**	0.17	0.16	0.25	0.22	130236

Note: a value in bold indicates that the corresponding heuristic provides the best result for the corresponding group of instances.

Because VRH and 3SH make more comprehensive decisions for all blocking containers above the target container at a time, they are able to perform better in this experiment. Table 6 provides the computation times of all 6 compared heuristics and the exact algorithm, showing that 3SH outperforms both VRH and Chain in terms of average computation time. Similarly, the exact algorithm consumes significantly more time when dealing with large-scale groups.

4.5. Comparative analysis

An analysis is conducted to compare the solution quality and computation times of 3SH with other methods across 3 datasets. In Table 7, "Solution Improvement" represents the improvement percentage in solution quality achieved by 3SH over the other compared algorithms. It is calculated as the difference of the result of the compared algorithm and the result of 3SH over the result of 3SH. "Time Increase" signifies the percentage increase in computational time when using 3SH compared to the other algorithms. This is the result of the difference of the time of 3SH and the time of the compared algorithm over the time of 3SH. Based on the average results from the table, 3SH is able to obtain better solutions than VRH in a shorter amount of time over all 3 datasets. Besides, on average, all the heuristics are capable of solving an instance within one millisecond, which is an exceptionally short amount of time that can be negligible in most cases.

To delve deeper into the relationship between VRH and 3SH, we categorize all instances from the 3 datasets based on *H*, *S*, and *N*. The computational results and times are plotted using bubbles in Figure 7a and 7b, respectively. The size of each bubble represents the relative magnitude of the value, with larger bubbles indicating larger values. For each combination of *H*, *S*, and *N*, only the smaller value of VRH and 3SH is plotted; red bubbles signify values from VRH, while blue bubbles indicate values from 3SH. We examine the proportion, location, and size of red and blue bubbles. From the perspective of solution quality, 3SH wins the majority of instances, with the losing (red) bubbles mainly concentrated in the lower left corner, which are of small size. However, in terms of computation time, 3SH wins just over half of the instances, but the winning (blue) bubbles are often larger in size. This experimentally demonstrates again why, on average, 3SH outperforms VRH in both effectiveness and efficiency. Subsequently, we try to elucidate this methodologically.

Table 5. Computational results for the instances from Zhu *et al.* (2012)

H	*S*	*N*	#inst	Min–Max	PR4	Chain	ChainF	VRH	3SH	Exact
3	6	15–17	300	6.71	6.68	6.71	6.72	**6.67**	**6.67**	6.64
3	7	18–20	300	7.85	7.83	7.84	7.85	**7.82**	**7.82**	7.77
3	8	21–23	300	9.01	8.98	9.00	9.03	**8.97**	8.98	8.93
3	9	24–26	300	10.46	10.42	10.45	10.48	**10.40**	**10.40**	10.37
3	10	27–29	300	11.73	11.68	11.70	11.69	11.67	**11.66**	11.59
4	6	20–23	400	12.91	12.84	12.83	12.86	12.79	**12.76**	12.51
4	7	24–27	400	14.96	14.87	14.90	14.95	**14.82**	14.84	14.50
4	8	28–31	400	17.35	17.24	17.19	17.22	17.15	**17.07**	16.72
4	9	32–35	400	18.98	18.93	18.92	18.97	18.81	**18.79**	18.47
4	10	36–39	400	21.21	21.11	21.07	21.08	20.99	**20.96**	20.54
5	6	25–29	500	20.24	20.08	20.00	20.08	19.90	**19.76**	19.01
5	7	30–34	500	24.01	23.85	23.78	23.82	23.56	**23.45**	22.52
5	8	35–39	500	27.45	27.22	27.04	27.03	26.88	**26.74**	25.73
5	9	40–44	500	30.16	29.97	29.86	29.93	29.75	**29.50**	28.31
5	10	45–49	500	33.44	33.28	33.06	32.96	32.93	**32.68**	31.45
6	6	30–35	600	29.68	29.40	29.17	29.19	28.80	**28.60**	26.96
6	7	36–41	600	34.27	33.95	33.74	33.65	33.35	**33.01**	31.00
6	8	42–47	600	38.89	38.47	38.19	38.05	37.77	**37.46**	35.31
6	9	48–53	600	43.49	43.23	42.85	42.69	42.46	**42.10**	39.52
6	10	54–59	600	47.80	47.48	46.88	46.69	46.56	**46.23**	43.31
7	6	35–41	700	40.29	39.82	39.61	39.29	38.73	**38.41**	35.45
7	7	42–48	700	46.94	46.34	46.07	45.75	45.05	**44.77**	41.10
7	8	49–55	700	52.80	52.31	51.93	51.42	50.81	**50.32**	46.25
7	9	56–62	700	59.16	58.60	58.13	57.40	57.08	**56.59**	51.91
7	10	63–69	700	65.14	64.40	63.82	63.30	62.80	**62.10**	57.02
Summary			12500	33.36	33.07	32.86	32.71	32.44	**32.20**	30.24

Note: a value in bold indicates that the corresponding heuristic provides the best result for the corresponding group of instances.

Table 6. Computation times for the instances from Zhu *et al.* (2012)

H	*S*	*N*	#inst	Min–Max	PR4	Chain	ChainF	VRH	3SH	Exact
3	6	15–17	300	**0.17**	0.18	0.19	0.22	0.33	0.26	<10
3	7	18–20	300	0.07	**0.06**	0.24	0.11	0.09	0.13	<10
3	8	21–23	300	0.08	0.08	0.11	**0.07**	**0.07**	0.08	<10
3	9	24–26	300	**0.06**	0.07	0.43	0.09	0.14	0.07	<10
3	10	27–29	300	**0.07**	0.09	0.08	0.09	0.10	0.08	<10
4	6	20–23	400	0.12	**0.11**	0.12	0.15	0.42	0.18	<10
4	7	24–27	400	**0.06**	0.07	0.13	0.08	0.20	0.15	<10
4	8	28–31	400	**0.07**	**0.07**	0.18	0.08	0.08	0.09	<10
4	9	32–35	400	**0.06**	0.07	0.31	0.08	0.20	0.09	10
4	10	36–39	400	**0.07**	0.10	0.17	0.09	0.10	0.09	10
5	6	25–29	500	0.09	**0.08**	0.11	0.18	0.41	0.17	30
5	7	30–34	500	**0.07**	0.08	0.17	0.12	0.22	0.16	50
5	8	35–39	500	**0.07**	**0.07**	0.22	0.10	0.08	0.10	70
5	9	40–44	500	**0.07**	0.08	0.24	0.09	0.31	0.10	100
5	10	45–49	500	**0.07**	0.15	0.19	0.10	0.13	0.11	160
6	6	30–35	600	**0.08**	0.09	0.11	0.19	0.21	0.18	300
6	7	36–41	600	0.15	0.13	0.15	**0.09**	0.14	0.15	490
6	8	42–47	600	**0.08**	**0.08**	0.16	0.09	0.09	0.12	620
6	9	48–53	600	**0.08**	0.09	0.18	0.14	0.17	0.12	1240
6	10	54–59	600	**0.08**	0.16	0.19	0.13	0.11	0.13	1350
7	6	35–41	700	**0.11**	0.12	0.26	0.20	0.16	0.16	1980
7	7	42–48	700	0.11	**0.10**	0.26	0.11	0.11	0.13	4220
7	8	49–55	700	**0.08**	0.09	0.24	0.11	0.16	0.16	5520
7	9	56–62	700	**0.10**	**0.10**	0.15	0.13	0.22	0.14	15330
7	10	63–69	700	**0.09**	0.13	0.15	0.12	0.11	0.16	18500
Summary			12500	**0.09**	0.10	0.19	0.12	0.17	0.13	2760

Note: a value in bold indicates that the corresponding heuristic provides the best result for the corresponding group of instances.

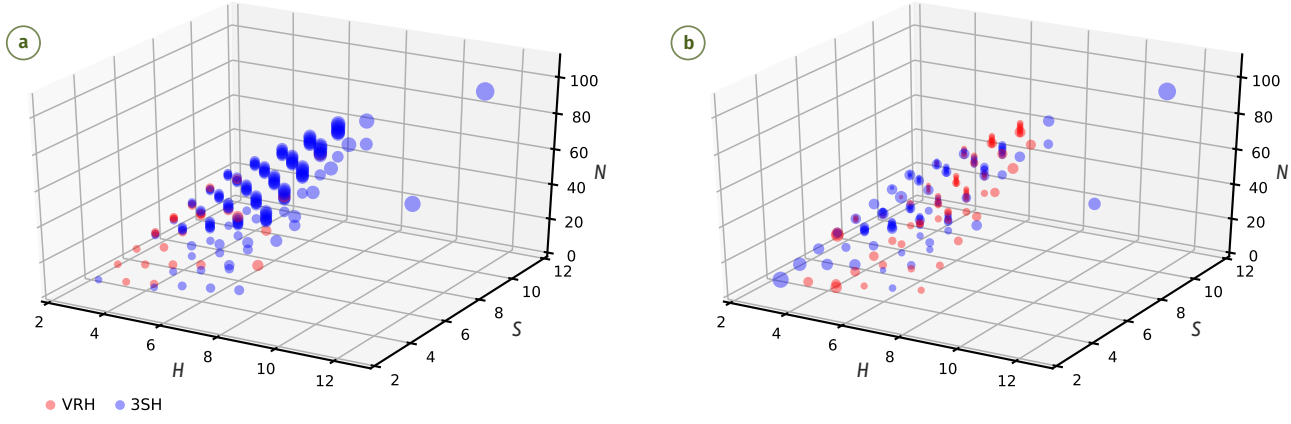

Figure 7. Comparative results of 3SH and VRH for all instances from 3 datasets:
(a) – computational results; (b) – computation times

From Figure 7a, we observe that the relative performance of the 2 methods is more strongly correlated with *H* than with *S*. Specifically, 3SH is more likely to outperform VRH when *H* exceeds 5. Below this number, the likelihood of 3SH beating VRH decreases. This is likely because 3SH, being well-designed, can handle long decision horizons more accurately, leading to better results. From Figure 7b, similar patterns emerge. When $H < 5$, 3SH tends to compute faster. For $H > 5$, 3SH is possible to takes longer time. At $H = 5$, 3SH is likely to be quicker when density is higher, i.e., larger *N* for the same *S*. Therefore, while 3SH generally outperforms VRH in both solution quality and speed across all 3 datasets, a closer look reveals that 3SH still spends more time to yield enhanced results.

Table 7. Solution improvement and time increase for 3SH compared with all other heuristics

	Wu & Ting (2010)		Caserta *et al.* (2011)		Zhu *et al.* (2012)	
	solution improvement [%]	time increase [%]	solution improvement [%]	time increase [%]	solution improvement [%]	time increase [%]
Min–Max	6.03	37.93	5.43	36.36	3.60	30.77
PR4	4.12	44.83	4.97	40.91	2.70	23.08
Chain	4.26	31.03	3.35	22.73	2.05	–46.15
ChainF	3.09	20.69	2.05	27.27	1.58	7.69
VRH	0.92	–3.45	0.61	–13.64	0.75	–30.77

Table 8. Ablation analysis of 3SH on the instances from Zhu *et al.* (2012)

H	*S*	*N*	#inst	3SH	Without enhancement 1	Without enhancement 2	Without stage 3
3	6	15–17	300	**6.67**	**6.67**	6.68	**6.67**
3	7	18–20	300	**7.82**	**7.82**	**7.82**	**7.82**
3	8	21–23	300	8.98	8.98	8.98	**8.97**
3	9	24–26	300	**10.40**	**10.40**	10.41	**10.40**
3	10	27–29	300	**11.66**	**11.66**	11.67	**11.66**
4	6	20–23	400	**12.76**	12.79	12.77	12.78
4	7	24–27	400	14.84	14.88	**14.78**	14.84
4	8	28–31	400	**17.07**	17.11	17.08	17.08
4	9	32–35	400	18.79	18.82	**18.78**	18.79
4	10	36–39	400	**20.96**	20.98	20.97	20.99
5	6	25–29	500	**19.76**	19.82	19.78	19.77
5	7	30–34	500	23.45	23.53	**23.43**	23.45
5	8	35–39	500	26.74	26.80	**26.70**	26.79
5	9	40–44	500	**29.50**	29.59	29.55	29.61
5	10	45–49	500	**32.68**	32.76	32.77	32.80
6	6	30–35	600	**28.60**	28.68	28.64	28.68
6	7	36–41	600	**33.01**	33.10	33.11	33.12
6	8	42–47	600	**37.46**	37.59	37.51	37.55
6	9	48–53	600	**42.10**	42.20	42.15	42.21
6	10	54–59	600	**46.23**	46.34	46.32	46.35
7	6	35–41	700	**38.41**	38.55	38.43	38.50
7	7	42–48	700	**44.77**	44.84	44.88	44.82
7	8	49–55	700	**50.32**	50.36	50.61	50.42
7	9	56–62	700	**56.59**	56.62	56.78	56.67
7	10	63–69	700	**62.10**	62.20	62.40	62.27
Summary			12500	**32.20**	32.26	32.27	32.26

Note: a value in bold indicates that the corresponding heuristic provides the best result for the corresponding group of instances.

4.6. Ablation analysis

2 enhancements and a new stage are introduced in 3SH; in this subsection we assess the effects of these 3 components. We choose the instances from Zhu *et al.* (2012) in this experiment. Table 8 demonstrates the results of the ablation analysis, in which "without enhancement 1" refers to using the priority order only in Stage 1, "without enhancement 2" refers to using the original VRI in Stage 2, and "without stage 3" refers to not adding the last stage in 3SH. As we can see from the comparison, each component can help improve the performance of the proposed heuristic. In Section 3 we describe the motivation for the enhancements introduced in 3SH but from the results we have to admit that these enhancements also have limitations and that they do not always improve the solution quality. However, the complete 3SH is still the best choice from an overall perspective.

5. Conclusions

In this article, we address the restricted CRP in maritime container terminals and design a new heuristic to solve it. The proposed heuristic combines the advantages of existing heuristic approaches while ameliorating their disadvantages. 3 benchmark datasets are used to test the proposed heuristic, and the computational results show

that the proposed heuristic outperforms the other existing heuristics in general, especially on the large-scale instances. Besides, the proposed heuristic wins in both efficiency and effectiveness, delivering shorter computation time and higher-quality solutions, compared to the state-of-the-art heuristics. From a management perspective, the proposed heuristic offers a practical tool for rapid decision-making in logistics scenarios, due to its high computational speed and quality solutions. Its quick computation is valuable in time-sensitive logistics environments, and its high-quality solution is beneficial to increase port operational efficiency and thus affect the supply chain by speeding up cargo throughput. For the future research, there are several promising directions. As the proposed heuristic is fast enough to provide a high-quality solution for a given instance, it could be incorporated as an efficient evaluation component in a mature meta-heuristic framework, e.g., beam search, look-ahead search, etc. We will further investigate this topic in our future work. Moreover, the idea of considering multiple containers simultaneously has the potential to better solve relevant container operation problems. Expanding this approach to address these variants will also be a significant focus in our future research.

Funding

This work was supported by:
- the National Natural Science Foundation of China (Grant No 72101160);
- the Stable Support Plan Program of Shenzhen Natural Science Fund (Grant No 20200810160835003).

Disclosure statement

The authors declare no conflicts of interest.

References

Bacci, T.; Mattia, S.; Ventura, P. 2020. A branch-and-cut algorithm for the restricted block relocation problem, *European Journal of Operational Research* 287(2): 452–459. https://doi.org/10.1016/j.ejor.2020.05.029

Bacci, T.; Mattia, S.; Ventura, P. 2019. The bounded beam search algorithm for the block relocation problem, *Computers & Operations Research* 103: 252–264. https://doi.org/10.1016/j.cor.2018.11.008

Carlo, H. J.; Vis, I. F. A.; Roodbergen, K. J. 2014. Storage yard operations in container terminals: literature overview, trends, and research directions, *European Journal of Operational Research* 235(2), 412–430. https://doi.org/10.1016/j.ejor.2013.10.054

Caserta, M.; Schwarze, S.; Voß, S. 2012. A mathematical formulation and complexity considerations for the blocks relocation problem, *European Journal of Operational Research* 219(1): 96–104. https://doi.org/10.1016/j.ejor.2011.12.039

Caserta, M.; Voß, S.; Sniedovich, M. 2011. Applying the corridor method to a blocks relocation problem, *OR Spectrum* 33(4): 915–929. https://doi.org/10.1007/s00291-009-0176-5

Expósito-Izquierdo, C.; Melián-Batista, B.; Moreno-Vega, J. M. 2014. A domain-specific knowledge-based heuristic for the blocks relocation problem, *Advanced Engineering Informatics* 28(4): 327–343. https://doi.org/10.1016/j.aei.2014.03.003

Expósito-Izquierdo, C.; Melián-Batista, B.; Moreno-Vega, J. M. 2015. An exact approach for the blocks relocation problem, *Expert Systems with Applications* 42(17–18): 6408–6422. https://doi.org/10.1016/j.eswa.2015.04.021

Feillet, D.; Parragh, S. N.; Tricoire, F. 2019. A local-search based heuristic for the unrestricted block relocation problem, *Computers & Operations Research* 108: 44–56. https://doi.org/10.1016/j.cor.2019.04.006

Galle, V.; Barnhart, C.; Jaillet, P. 2018. A new binary formulation of the restricted container relocation problem based on a binary encoding of configurations, *European Journal of Operational Research* 267(2): 467–477. https://doi.org/10.1016/j.ejor.2017.11.053

Jin, B.; Tanaka, S. 2023. An exact algorithm for the unrestricted container relocation problem with new lower bounds and dominance rules, *European Journal of Operational Research* 304(2): 494–514. https://doi.org/10.1016/j.ejor.2022.04.006

Jin, B.; Zhu, W.; Lim, A. 2015. Solving the container relocation problem by an improved greedy look-ahead heuristic, *European Journal of Operational Research* 240(3): 837–847. https://doi.org/10.1016/j.ejor.2014.07.038

Jovanovic, R.; Tuba, M.; Voß, S. 2019. An efficient ant colony optimization algorithm for the blocks relocation problem, *European Journal of Operational Research* 274(1): 78–90. https://doi.org/10.1016/j.ejor.2018.09.038

Jovanovic, R.; Voß, S. 2014. A chain heuristic for the blocks relocation problem, *Computers & Industrial Engineering* 75: 79–86. https://doi.org/10.1016/j.cie.2014.06.010

Kim, K. H.; Hong, G.-P. 2006. A heuristic rule for relocating blocks, *Computers & Operations Research*, 33(4): 940–954. https://doi.org/10.1016/j.cor.2004.08.005

Kim, K. H.; Park, Y. M.; Ryu, K.-R. 2000. Deriving decision rules to locate export containers in container yards, *European Journal of Operational Research* 124(1): 89–101. https://doi.org/10.1016/S0377-2217(99)00116-2

Ku, D.; Arthanari, T. S. 2016. On the abstraction method for the container relocation problem, *Computers & Operations Research* 68: 110–122. https://doi.org/10.1016/j.cor.2015.11.006

Lee, Y.; Hsu, N.-Y. 2007. An optimization model for the container pre-marshalling problem, *Computers & Operations Research* 34(11): 3295–3313. https://doi.org/10.1016/j.cor.2005.12.006

Lehnfeld, J.; Knust, S. 2014. Loading, unloading and premarshalling of stacks in storage areas: survey and classification, *European Journal of Operational Research* 239(2): 297–312. https://doi.org/10.1016/j.ejor.2014.03.011

Lersteau, C.; Shen, W. 2022. A survey of optimization methods for block relocation and premarshalling problems, *Computers & Industrial Engineering* 172: 108529. https://doi.org/10.1016/j.cie.2022.108529

Maglić, Li.; Gulić, M.; Maglić, Lo. 2020. Optimization of container relocation operations in port container terminals, *Transport* 35(1): 37–47. https://doi.org/10.3846/transport.2019.11628

Murty, K. G.; Wan, Y.-W.; Liu, J.; Tseng, M. M.; Leung, E.; Lai, K.-K.; Chiu, H. W. C. 2005. Hongkong international terminals gains elastic capacity using a data-intensive decision-support system, *Interfaces* 35(1): 61–75. https://doi.org/10.1287/inte.1040.0120

Petering, M. E. H.; Hussein, M. I. 2013. A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem, *European Journal of Operational Research* 231(1): 120–130. https://doi.org/10.1016/j.ejor.2013.05.037

Quispe, K. E. Y.; Lintzmayer, C. N.; Xavier, E. C. 2018. An exact algorithm for the Blocks Relocation Problem with new lower bounds, *Computers & Operations Research* 99: 206–217. https://doi.org/10.1016/j.cor.2018.06.021

Tanaka, S.; Takii, K. 2016. A faster branch-and-bound algorithm for the block relocation problem, *IEEE Transactions on Automation Science and Engineering* 13(1): 181–190. https://doi.org/10.1109/tase.2015.2434417

Tanaka, S.; Voß, S. 2022. An exact approach to the restricted block relocation problem based on a new integer programming formulation, *European Journal of Operational Research* 296(2): 485–503. https://doi.org/10.1016/j.ejor.2021.03.062

Ting, C.-J.; Wu, K.-C. 2017. Optimizing container relocation operations at container yards with beam search, *Transportation Research Part E*: *Logistics and Transportation Review* 103: 17–31. https://doi.org/10.1016/j.tre.2017.04.010

Tricoire, F.; Scagnetti, J.; Beham, A. 2018. New insights on the block relocation problem, *Computers* & *Operations Research* 89: 127–139. https://doi.org/10.1016/j.cor.2017.08.010

Ünlüyurt, T.; Aydın, C. 2012. Improved rehandling strategies for the container retrieval process, *Journal of Advanced Transportation* 46(4): 378–393. https://doi.org/10.1002/atr.1193

Wu, K.-C.; Ting, C.-J. 2010. A beam search algorithm for minimizing reshuffle operations at container yards, in *LOGMS 2010*: *the 1st International Conference on Logistics and Maritime Systems*, 15–17 September 2010, Busan, Korea, 703–710.

Wu, K.-C.; Ting, C. J. 2012. Heuristic approaches for minimizing reshuffle operations at container yard, in *Proceedings of the 13th Asia Pacific Industrial Engineering* & *Management Systems Conference*, 2–5 December 2012, Phuket, Thailand, 1407–1451.

Zehendner, E.; Caserta, M.; Feillet, D.; Schwarze, S.; Voß, S. 2015. An improved mathematical formulation for the blocks relocation problem, *European Journal of Operational Research* 245(2): 415–422. https://doi.org/10.1016/j.ejor.2015.03.032

Zhang, C.; Guan, H.; Yuan, Y.; Chen, W.; Wu, T. 2020. Machine learning-driven algorithms for the container relocation problem, *Transportation Research Part B*: *Methodological* 139: 102–131. https://doi.org/10.1016/j.trb.2020.05.017

Zhu, W.; Qin, H.; Lim, A.; Zhang, H. 2012. Iterative deepening A^* algorithms for the container relocation problem, *IEEE Transactions on Automation Science and Engineering* 9(4): 710–722. https://doi.org/10.1109/tase.2012.2198642