Loughborough University

This item was submitted to Loughborough's Research Repository by the author.
Items in Figshare are protected by copyright, with all rights reserved, unless otherwise indicated.

Research into container reshuffling and stacking problems in container terminal yards

PLEASE CITE THE PUBLISHED VERSION

http://dx.doi.org/10.1080/0740817X.2014.971201

PUBLISHER

© Taylor and Francis Ltd.

VERSION

AM (Accepted Manuscript)

PUBLISHER STATEMENT

This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) licence. Full details of this licence are available at: https://creativecommons.org/licenses/by-nc-nd/4.0/

LICENCE

CC BY-NC-ND 4.0

REPOSITORY RECORD

Tang, Lixin, Wei Jiang, Jiyin Liu, and Yun Dong. 2015. “Research into Container Reshuffling and Stacking Problems in Container Terminal Yards”. Loughborough University. https://hdl.handle.net/2134/18119.

To cite this article: Lixin Tang, Wei Jiang, Jiyin Liu & Yun Dong (2015) Research into container reshuffling and stacking problems in container terminal yards, IIE Transactions, 47:7, 751-766, DOI: 10.1080/0740817X.2014.971201

Research into container reshuffling and stacking problems in container terminal yards

Lixin Tang [1] Wei Jiang

Liaoning Key Laboratory of Manufacturing System and Logistics, The Logistics Institute, Northeastern University, Shenyang, 110004, China

Jiyin Liu

School of Business and Economics, Loughborough University, Leicestershire, LE11 3TU, UK

Yun Dong

Liaoning Key Laboratory of Manufacturing System and Logistics, The Logistics Institute, Northeastern University, Shenyang, 110004, China

Abstract

Container stacking and reshuffling are important issues in the operations management of container terminals. Minimizing the number of reshuffles can increase productivity of the yard cranes and the terminal efficiency. In this research, we improve the existing static reshuffling model, develop five effective heuristics and analyze the performance of these algorithms. A discrete-event simulation model is developed to animate the stacking, retrieving and reshuffling operations and to test the performance of the proposed heuristics and their extended versions in the dynamic environment with arrivals and retrievals of containers. The experimental results for the static problem show that the improved model can solve the reshuffling problem more quickly than the existing model and the proposed extended heuristics are superior to the existing ones. The experimental results for the dynamic problem show that the results of the extended versions of the five proposed heuristics are superior or similar to the best results of the existing heuristics and consume very little time.

Keywords: Heuristics, Container terminal yard, Reshuffling and stacking, Improved model, Performance analysis, Dynamic simulation

[1] Corresponding author qhjytlx@mail.neu.edu.cn

1. Introduction

With the continued increase in global trade, the volume of container transportation around the world has been growing steadily. According to Zhang (2010), the total throughput of world's top 50 container terminals was approximately 306 million TEUs (Twenty-foot Equivalent Units) in 2009, which is an increase of 16.8% from that in 2005. To cope with the increasing volume and maintain service quality, it is critical for terminals to enhance their space utilization and operational efficiency in addition to expanding the capacities.

A container terminal can be roughly divided into two main areas: one is the quayside for berthing vessels, and the other is the terminal yard where containers are stored. The container handling process in the terminal yard is dynamic with containers continually being stored and retrieved. To better utilize the limited space at the terminal, the containers stored in the yard are stacked one on top of another. In a container yard, the storage area is generally divided into blocks. Fig. 1 adopted from Wan *et al*. (2009) but using slightly different terminology shows the configuration of a block in a terminal yard. The directions of the length, width and height of a block are defined in the lower right corner of the diagram. The set of containers in a block that share the same length coordinate is called a bay; the set sharing the same width coordinate is called a lane; the set sharing the same height coordinate is called a tier; the set of containers in a bay sharing the same width coordinate is called a column. Note that a bay, as defined in Fig. 1, is called a stack in Wan *et al*. (2009). The smallest storage unit is called a position, and the three-dimensional coordinate (bay number, column number, tier number) is used to represent a position. A typical block served by rubber tyre gantry cranes (RTGCs) may include more than 20 bays with each bay normally consisting of 6 columns, and in each column, the containers may be stacked up to 4 or 5 high.

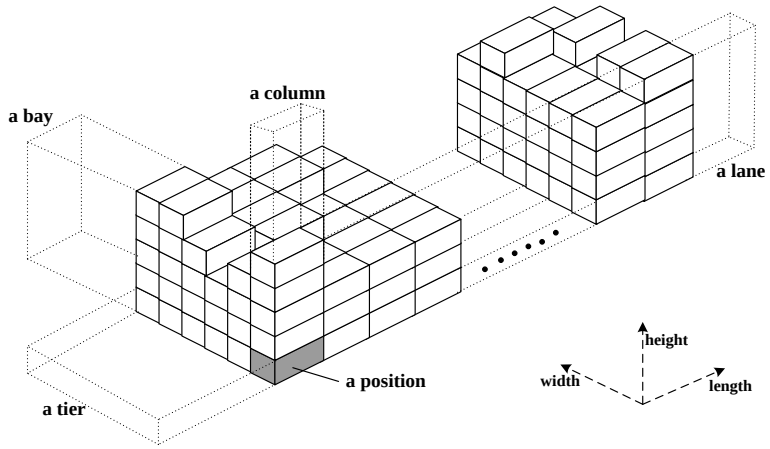

Fig. 1 The layout of a block

Although stacking containers high can improve the space utilization, it will give rise to high handling costs because of unproductive moves. If a container to be retrieved is not on top of a column, those blocking it need to be moved to other columns in the same bay. Because there is only one container being retrieved in the process, the moves of other containers are considered as unproductive moves. While storing new arrival containers is called stacking, storing blocking containers is called reshuffling. The action of reshuffling a blocking container to another position is called a reshuffle. The storage positions of incoming and reshuffled containers in a bay should be well determined to avoid future reshuffles as much as possible and to improve the operating efficiency of the yard cranes. In this article, we study the storage location assignment problem for one bay in static and dynamic cases.

In practice, the storage and retrieval operations in terminal yards are dynamic, and the configuration of each bay changes with arrivals and departures of containers. Due to the dynamic features, optimization models and solution methods performing well in static situations may not necessarily perform well in this dynamic situation. Therefore, optimization models and methods for the problem need to be tested in the dynamic environment. Furthermore, the solution methods must be fast enough to make decisions in real-time. Simulation can capture the characteristics of the dynamic system operations and, therefore, is an appropriate platform in evaluating the effectiveness and efficiency of decision rules and optimization methods in dynamic operations.

The remainder of the paper is organized as follows. Section 2 reviews the related research on reshuffling and stacking as well as simulation studies of terminal operations. Section 3 gives a more detailed description of the reshuffling and

dynamic stacking problems in one bay and develops an improved model for the reshuffling problem. Five new heuristics are then proposed and analyzed in Section 4. In Section 5, a discrete-event simulation model is developed to simulate the stacking, retrieving and reshuffling operations and to test the algorithms in a dynamic environment. Section 6 reports the results of the experiments to evaluate the algorithms. Finally Section 7 concludes the paper.

2. Literature review

In this section, we review previous studies related to reshuffling and stacking problems in container terminal yards. For the reshuffling problem in a bay where the initial configuration was given and there were no new container arrivals, which will be called the static reshuffling problem in this paper, Kim (1997) developed a methodology to evaluate the expected number of reshuffles to pick up a specific container, and the total number of reshuffles to empty all containers. Kim *et al*. (2000) proposed a method to determine the storage location of an arriving export container according to its weight. First, the arriving export containers were classified into three pre-determined weight groups: light, medium and heavy. Next, a dynamic programming model was presented to determine the optimal storage slot for each arriving container to minimize the total expected number of reshuffles. Finally, a decision tree rule was used to determine the storage slots for arriving containers instead of using time-consuming dynamic programming. Kang *et al*. (2006) presented a method for deriving a strategy for stacking containers with uncertain weight information, and the method can significantly reduce the number of reshuffles at the time of loading compared to the traditional same-weight-group-stacking strategy. Zhang (2000) proposed the lowest-slot (LS) heuristic to put an incoming or reshuffled container to the lowest available position of a bay. Murty *et al*. (2005) proposed the reshuffling index (RI) heuristic to determine a position for an incoming or reshuffled container. An incoming or reshuffled container was put in the column where the reshuffling index was the smallest. For an incoming or reshuffled container, the RI of a column represents the number of containers that will be picked up earlier than the container being considered. Kim and Hong (2006) studied the static reshuffling problem and proposed a branch and bound algorithm to determine the optimal storage positions for reshuffled containers. They also proposed a heuristic, ENAR, to quickly obtain satisfactory solutions, but

this algorithm's time complexity is exponential because of the recursive manner for calculating the expected number of additional reshuffles. For the static reshuffling problem, Wan *et al*. (2009) divided the process of emptying a bay into stages, each for retrieving one container, and defined binary variables to indicate the container positions in each stage. Using these and other variables they formulated the problem as an integer programming model. Heuristics based on the integer program were then developed and applied to the static reshuffling problem as well as the dynamic problem with continual retrievals and arrivals of containers. Experimental results showed that the model-based heuristics were competitive for both static and dynamic problems. Caserta *et al*. (2009) developed a binary description of the bay configuration to adapt heuristics and metaheuristics, and used the new description within a look ahead heuristic to solve the static reshuffling problem. Caserta *et al*. (2011) proposed a new metaheuristic approach based on dynamic programming for the static reshuffling problem. The above two methods are both metaheuristics, and their performances are only tested in a static situation. Lee and Lee (2010) presented a three-phase heuristic for the static reshuffling problem, but the objective was to minimize the number of container movements and the crane's working time, which was different from the problem investigated in our paper. Forster and Bortfeldt (2012) proposed a tree search procedure for the container relocation problem. The key of the algorithm is to determine a move sequence with minimum length, and it is also a metaheuristic approach. Caserta *et al*. (2011) reviewed recent contributions dealing with reshuffling operations in container terminals. The remarshalling problem, the premarshalling problem and the relocation problem were considered and the related algorithms to tackle such problems were summarized. Though most previous research on the static reshuffling problem focused on developing efficient approximate algorithms, the complexity of the problem was unknown until recently. Caserta *et al*. (2012) proved that the static reshuffling problem is NP-hard by reducing it to the decision problem of Mutual Exclusion Scheduling (MES).

The performance of the reshuffling algorithms in practice needs to be evaluated in a dynamic, operational environment with container arrivals and retrievals. Simulation is a suitable tool for evaluating the algorithms or rules. There have been some simulation studies on container yard operations, e.g., Duinkerken *et al*. (2001), Sgouridis and Angelides (2002), Hartmann (2004), Park *et al*. (2006), Stahlbock and Voss (2010), Borgman *et al*. (2010), Petering (2010) and Klaws *et al*. (2011). Most

of these studies simulate the operation of the entire terminal, which includes the yard operations and the transport between the yard and the vessels. Different stacking rules are compared for assigning storage positions to incoming containers. The criteria used for comparison include the percentage of reshuffle moves and the time for both container handling and crane travel.

In this paper, both the static reshuffling problem and the dynamic stacking problem are investigated. For the static reshuffling problem, five new construction heuristics and their extended versions are proposed, and then worst-case performance analysis is performed. The existing static reshuffling model proposed by Wan *et al*. (2009) is improved to reduce the required solution time. The optimal solutions from the model will be used to evaluate the heuristics. A simulation model is developed to compare the proposed heuristics with the existing heuristics in a dynamic situation.

3. Problem description and the improved model

3.1 Problem description

In actual container yards, containers are continually stored and retrieved. In this dynamic process, a bay has a specific configuration at any moment. The configuration can be described by the size of the bay, i.e., the numbers of columns and tiers of the bay, as well as the set of containers stored in the bay and the pattern they are stored in. Fig.2 shows a configuration of a bay at a given moment. The containers in the bay are numbered according to the order that they are retrieved, and a smaller number represents a higher priority in the retrieving order. A storage position in the bay is defined by a column-index and a tier-index. For a given configuration of a bay, if there are no new containers assigned to the bay before all the containers are retrieved, we have a static reshuffling problem to empty all containers in the bay according to their priorities to minimize the total number of reshuffles. If new containers continually arrive to the bay while the containers in the bay are retrieved, then the problem is a dynamic reshuffling and stacking problem. In the dynamic problem, when a new container arrives its storage position must be decided, and when a container is to be retrieved, decisions must be made on the positions where the blocking containers (if any) should be reshuffled. The objective of the dynamic problem is to minimize the average number of reshuffles needed to retrieve a container in the long run. Wan *et al*. (2009) demonstrated that this general problem is NP-hard by associating it with a one-bay vessel stowage problem.

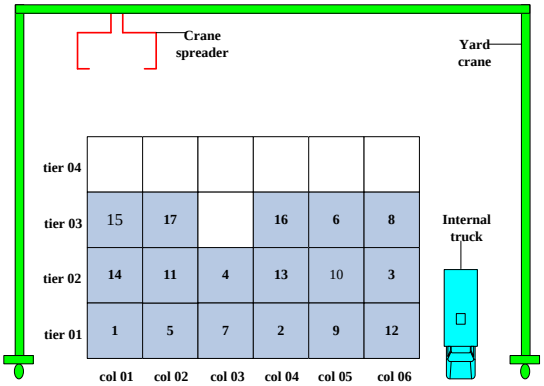

Fig. 2 Containers stored in a bay to be retrieved in a given order

3.2 The improved model

At any decision making point, the dynamic problem may be viewed as a static problem considering the available information at that time. Therefore, the study into the static problem is the basis of the dynamic problem and algorithms for solving the static problem can be used for making decisions in the dynamic problem. Even the static problem is difficult to formulate as a mathematical model because it has a dynamic feature, i.e., the decisions for the earlier reshuffling of containers will influence the later decisions. Wan *et al*. (2009) successfully formulated the first integer linear programming model for the static problem, which was called *MRIP model*. To develop the model, they defined the operations related to retrieve one container as a stage, and introduced innovative variables to represent the bay configuration as well as reshuffling decisions in each stage. They also skillfully constructed constraints to trace the transitions from the configuration in one stage to the next stage to ensure physical feasibility. The model uses column-relationship variables to identify whether a container is in the same column with the container to be retrieved in a stage. These variables in turn determine the reshuffle variables, which indicate whether a container needs to be reshuffled in that stage.

The MRIP model takes a long computation time to solve the problem for a bay with a large number of containers. In order to obtain an optimal solution more quickly, we improve the MRIP model by removing the column-relationship variables and some associated constraints. Determining the reshuffle variables, which the column-relationship variables were used for, will be achieved by introducing new reshuffling-related constraints for individual columns.

For easy comparison, we use the same notations for the parameters and decision variables as in the MRIP model. The notations are listed below for completeness.

Parameters:

S – The total number of containers initially stored in the bay.

P – The total number of storage positions (tiers) in each column of the bay.

C – The total number of columns in the bay.

s – Index for the container to be retrieved, and also the stage for retrieving this container, $1 \le s \le S$.

i,j – Indexes for the containers under consideration, $1 \le i, j \le S$.

p – Index for positions in a column by counting from the lowest position, $1 \le p \le P$.

c – Index for the columns in the bay, $1 \le c \le C$.

X_{1icp} – Indicating the initial locations of the containers in the bay. If container i is stored in position p of column c, X_{1icp}=1; otherwise, X_{1icp}=0.

Decision variables:

$$x_{sicp} = \begin{cases} 1 & \text{if container } i \text{ is at position } p \text{ of column } c \text{ at the beginning of stage } s \\ 0 & \text{otherwise} \end{cases}$$

$$y_{si} = \begin{cases} 1 & \text{if container } i \text{ is reshuffled in the retrieval of container } s \\ 0 & \text{otherwise} \end{cases}$$

$$w_{sij} = \begin{cases} 1 & \text{if containers } i \text{ and } j \text{ are reshuffled during stage } s \text{ and container } j \\ & \text{is at a higher position than container } i \text{ before reshuffling} \\ 0 & \text{otherwise} \end{cases}$$

The model:

With the above notations, the improved model can be formulated as follows:

(ILP)

$$\min \sum_{s=1}^{S-1} \sum_{i=s+1}^{S} y_{si} \tag{1}$$

s.t.

$$(1 - \sum_{p=1}^{P} x_{sscp})P + y_{si} \ge (\sum_{p=1}^{P} p x_{sicp} - \sum_{p=1}^{P} p x_{sscp}) / P \qquad 1 \le s < i \le S,\ 1 \le c \le C; \tag{2}$$

$$(\sum_{p=1}^{P} p x_{sscp} - \sum_{p=1}^{P} p x_{sicp}) / P \le 1 - y_{si} \qquad 1 \le s < i \le S,\ 1 \le c \le C; \tag{3}$$

$$\sum_{c=1}^{C} \sum_{p=1}^{P} x_{sicp} = 1 \qquad 1 \le s \le i \le S; \tag{4}$$

$$\sum_{i=s}^{S} x_{sicp} \le 1 \qquad 1 \le s \le S, 1 \le c \le C, 1 \le p \le P; \tag{5}$$

$$\sum_{i=s}^{S} x_{sicp} \le \sum_{i=s}^{S} x_{sic,p-1} \qquad 1 \le s \le S, 1 \le c \le C, 2 \le p \le P; \tag{6}$$

$$\sum_{p=1}^{P} x_{s+1,icp} \le 2 - y_{si} - \sum_{p=1}^{P} x_{sscp} \qquad 1 \le s < i \le S, 1 \le c \le C; \tag{7}$$

$$2 - y_{si} - y_{sj} + w_{sij} \ge (\sum_{c=1}^{C}\sum_{p=1}^{P} px_{sjcp} - \sum_{c=1}^{C}\sum_{p=1}^{P} px_{sicp}) / P \qquad 1 \le s < i \le S, 1 \le s < j \le S, i \ne j; \tag{8}$$

$$y_{si} + y_{sj} + w_{sij} \le 3 + (\sum_{c=1}^{C}\sum_{p=1}^{P} px_{sjcp} - \sum_{c=1}^{C}\sum_{p=1}^{P} px_{sicp}) / P \qquad 1 \le s < i \le S, 1 \le s < j \le S, i \ne j; \tag{9}$$

$$w_{sij} \le y_{si} \qquad 1 \le s < i \le S, 1 \le s < j \le S, i \ne j; \tag{10}$$

$$w_{sij} \le y_{sj} \qquad 1 \le s < i \le S, 1 \le s < j \le S, i \ne j; \tag{11}$$

$$\sum_{p=1}^{P} px_{s+1,icp} - \sum_{p=1}^{P} px_{s+1,jcp} \ge -P(1 - w_{sij}) - P(1 - y_{si}) - P(1 - y_{sj}) - P(1 - \sum_{p=1}^{P} x_{s+1,icp}) \qquad 1 \le s < i \le S, 1 \le s < j \le S, i \ne j, 1 \le c \le C; \tag{12}$$

$$x_{s+1,icp} - x_{sicp} \ge -y_{si} \qquad 1 \le s < i \le S, 1 \le c \le C, 1 \le p \le P; \tag{13}$$

$$x_{sicp} - x_{s+1,icp} \ge -y_{si} \qquad 1 \le s < i \le S, 1 \le c \le C, 1 \le p \le P; \tag{14}$$

$$x_{1icp} = X_{1icp} \qquad 1 < i \le S, 1 \le c \le C, 1 \le p \le P; \tag{15}$$

$$x_{sicp} = X_{1icp} \qquad 2 \le s \le \min\{i, s_i\}, s \le i \le S, 1 \le c \le C, 1 \le p \le P; \tag{16}$$

$$y_{si} \in \{0,1\} \qquad 1 \le s < i \le S; \tag{17}$$

$$w_{sij} \in \{0,1\} \qquad 1 \le s < i \le S, 1 \le s < j \le S, i \ne j; \tag{18}$$

$$x_{sicp} \in \{0,1\} \qquad 1 \le s \le i \le S, 1 \le c \le C, 1 \le p \le P. \tag{19}$$

The objective of the model is to minimize the total number of reshuffles, which is expressed in the same way as in the MRIP model. The MRIP model uses three types of variables, u_{si}, v_{si} and z_{si}, and five sets of constraints associated with them to identify whether a container i is in the same column as the container to be retrieved in stage s. In the improved model, these variables and the five sets of constraints are removed. Meanwhile, constraint sets (2) and (3) are used to determine the reshuffle variables y_{si}.

Note that for a pair of containers s and i, a constraint (2) is formulated for every column c. If both s and i are in this column and the position of i, which is p_i, is higher than the position of s, which is p_s, then the constraint becomes $y_{si} \ge (p_i - p_s) / P$. The right hand side of this constraint is a positive number less than 1 which correctly forces y_{si} to be 1, indicating that container i needs to be

reshuffled when retrieving s. In all other situations, this constraint does not set any restriction on y_{si}. In these situations, container i does not need to be reshuffled for retrieving s and Constraint (3) forces y_{si} to be 0.

Constraints (4) to (16) are the same as constraints in the MRIP model (with different numberings). Hence we only briefly explain them here. Constraints (4) to (6) ensure that each container $i \geq s$ must occupy a feasible slot. Constraints (7) ensure that a reshuffled container cannot be reshuffled to its current column. Constraints (8) to (11) and (12) consider the relative heights of the two reshuffled containers at stage s and s+1, respectively. Constraints (13) and (14) ensure that containers not moved keep their positions in the next stage. Constraints (15) and (16) assign known values to x_{sicp}. Constraints (17) to (19) define the nature and range of the decision variables.

Compared with the MRIP model, the number of binary decision variables in the improved model is reduced by $3S^2$ due to the removal of the column-relationship variables u_{si}, v_{si} and z_{si}, although the number of constraints increases by (C-4)S(S-1) because of using constraints (2) and (3) to replace the constraints related to these variables. As demonstrated in the computational results in Section 6, the improved model can obtain an optimal solution in shorter time.

4. Heuristics and performance analysis

4.1 Heuristics

Because the location assignment of a reshuffled container may cause further future reshuffles and affect remaining retrieval decisions, the number of possible bay configurations in the retrieval process increases exponentially as the number of containers to be retrieved increases. Therefore, methods to generate an optimal solution such as the branch and bound algorithm and the integer linear programming model, are time-consuming and not suitable for practical uses. It is necessary and realistic to develop fast and effective heuristics to obtain an approximate solution. Murty *et al*. (2005) and Kim and Hong (2006) have proposed the reshuffling index (RI) heuristic and the ENAR heuristic, respectively, for the static reshuffling problem to obtain approximate solutions. Wan *et al*. (2009) proposed the extended versions of RI and ENAR as well as a MRIP-based heuristic to obtain better solutions. In this section, we propose five new polynomial time heuristics, referred to as H1 through H5, for the reshuffling problem and try to analyze their properties and performance.

These heuristics share the same overall framework but use different heuristic rules to determine the positions for the reshuffled containers.

The main idea is to choose the position for each reshuffled container to avoid or reduce the number of possible further reshuffles in the future as much as possible.

The heuristic framework:

The basic framework of the heuristics is outlined as follows. In each iteration of the procedure, one container is retrieved. We use S as a dynamic parameter in this procedure. It initially represents the total number of containers in the original bay, and then in each iteration it represents the number of containers remaining to be retrieved. After retrieving a container, therefore, S will be reduced by 1. The remaining containers will also be renumbered from 1 to S while still keeping their retrieval order, i.e., smaller numbered containers are to be retrieved earlier. With the renumbering, the container to be retrieved in each iteration is always container 1. In this procedure M is used to denote the accumulated number of reshuffles, and $M1$ is used to denote the number of reshuffles for retrieving the container in the current iteration.

Step 0: Initialize $M = 0$.

Step 1: If $S = 1$, retrieve container 1; stop. The total number of reshuffles is M.

Step 2: Compute the number of containers blocking container 1, and denote it as $M1$. If $M1=0$, go to Step 4.

Step 3: Reshuffle the $M1$ blocking containers to new storage positions according to a heuristic rule.

Step 4: Retrieve container 1; let $M = M + M1$. Renumber the containers such that container $(i+1)$ becomes container i, $i=1,\ldots,S-1$; let $S=S-1$; go to Step 1.

The heuristic rules:

The heuristic rules used to determine the storage positions of the reshuffled containers in Step 3 of the proposed heuristics H1 through H5 are described below. In the descriptions, RI of a column is the total number of containers in this column to be retrieved earlier than the reshuffled container being considered. BI of a column is the number of containers that will block the container with the smallest number in the column if the reshuffled container being considered is put into this column.

H1: Consider each of the $M1$ blocking containers from the top down. For each of these containers k, define the smallest container number in each column c as n_c. For an empty column, n_c is defined as $S+1$. If there is an available column c

that satisfies $n_c>k$, put container k into column c. Break ties by putting container k into the column with n_c closest to k. If no column satisfies the above condition, put container k into an available column with the minimum RI. Break ties again by putting it into the column with n_c closest to k. Update the configuration of the bay, and consider the next blocking container in the same way until all $M1$ containers are considered.

H2: Consider each of the M_1 blocking containers from the top down. For each of these containers k, if there is a column c satisfying $n_c>k$, put container k into column c. Break ties by putting container k into the column with n_c closest to k. If no column satisfies the above condition, put container k into a column with the minimum BI. Break ties again by putting it into the column with n_c closest to k. Update the configuration of the bay and consider the next blocking container in the same way until all $M1$ containers are considered.

H3: If the number of the blocking containers is not greater than the number of the available columns and their numberings are strictly increasing from the top down in their current column, determine the new storage positions of these blocking containers in decreasing order of their numbers according to H1, but disallow any two reshuffled containers to be placed into the same column. Otherwise, determine the new storage positions of the reshuffled containers according to H1 directly.

H4: Determine the new storage positions of the blocking containers in decreasing order of their numbers according to H1, but using the adjusted RI values. When considering to place a blocking container into a target column, if its position in the original column is higher than those of any blocking containers already assigned to this target column, then this container should be placed below these containers in the target column, and the number of these containers are added to the RI value calculated in the normal way.

H5: The same as the rule in H4 above except that H1 is replaced by H2, and RI is replaced by BI.

It can be observed that the computation time complexities of heuristics H1, H2, H3, H4 and H5 are $O(SCP)$, $O(SCP)$, $O(SP\log P+SCP)$, $O(SP\log P+SCP)$ and $O(SP\log P+SCP)$, respectively.

Based on the description of the heuristics, it is clear that heuristics H1 and H2 are

the basis of the other heuristics. Both H1 and H2 first try to assign the reshuffled container k to a column c with $n_c>k$ so that the reshuffled container will leave earlier and, thus, not block other containers in the column. Choosing the column with the smallest n_c in case of a tie not only ensures container k does not block the other containers but also leaves the columns with a greater n_c available for the later reshuffled containers so that there will be a lower chance for them to cause blocking in these columns. For example, in the situation shown in Fig.3(a), containers 2 and 6 must be reshuffled when picking up container 1. When container 2 is considered, $n_c>2$ for all available columns ($n_2=4$, $n_3=3$ and $n_4=7$), and in such a case of a tie, container 2 will be reshuffled to column 3 according to H1 or H2. By doing so, the empty column 4 is left to accommodate the next reshuffled container 6 to avoid it blocking other containers. In the case where container k has to block the other containers ($n_c<k$ for every available column c), H1 assigns it to the column where it blocks the least other containers, while H2 assigns it to the column where the least reshuffles are needed for the next retrieval. In case of a tie, H1 and H2 both choose the column with the largest n_c, which delays the next reshuffling of k to the latest possible time to reduce the chances of container k being further reshuffled. An example is shown in Fig.3(b) where containers 5 and 3 must be reshuffled when picking up container 1. When the blocking container 5 is considered, it will be reshuffled to column 2 according to H1 because the smallest container number in column 2 is the largest among all available columns when the values of RI are the same, while container 5 will be reshuffled to column 3 according to H2 because $BI_3 <BI_2$.

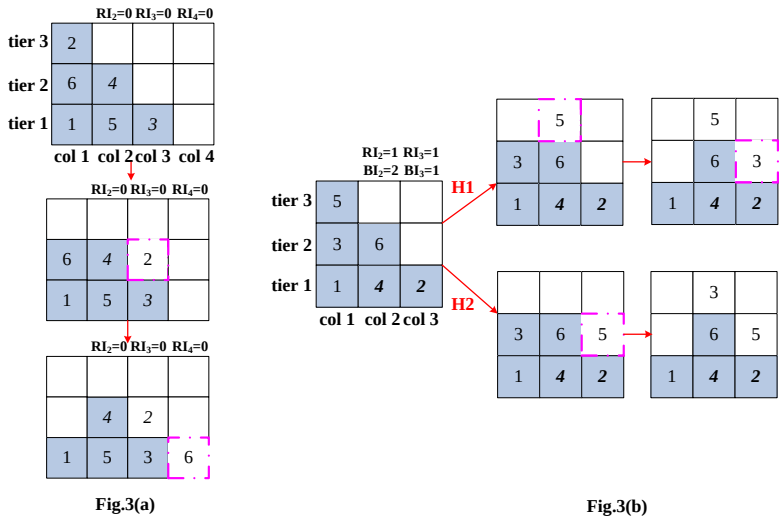

Fig.3(a) Fig.3(b)

Fig. 3 Examples to illustrate algorithms H1 and H2

When retrieving the target container in the current stage, if there are several blocking containers to be reshuffled and if a blocking container at a higher position has a smaller number than the container at a lower position, then assigning new storage positions to these containers in the order of their actual reshuffling may result in blocking among themselves in the new positions in case they are assigned to the same column. H3 through H5 assign new storage positions to the blocking containers in decreasing order of their numberings in an attempt to avoid this. H3 uses H1 as its base. H3 only changes the order of decisions and avoids assigning any two reshuffled containers to the same column in a special situation. H4 and H5 always make the assignment decisions for the reshuffled containers in decreasing order of their numberings, but H4 and H5 use H1 and H2 as their basis, respectively. Note that although the order of making the assignment decisions is different, the order of actual reshuffling operations in these heuristics is still from the top down, and in case any two reshuffled containers are put in the same column, the container at a higher position in the current column will be at a lower position in the new column.

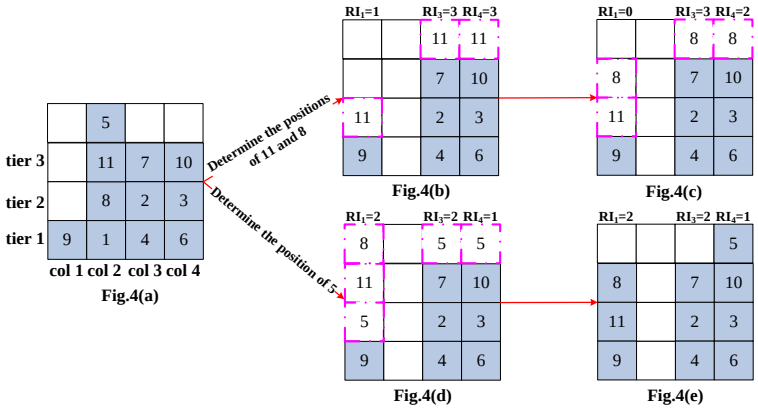

Fig. 4 An example to illustrate algorithm H4

To make algorithm H4 easier to understand, a small example is illustrated, as shown in Fig.4. To pick up container 1, the blocking containers 5, 11 and 8 must be reshuffled. According to H4, the new storage positions of containers 11, 8 and 5 are determined in this order. First by attempting to put container 11 to each target column, we can calculate their RI. Because the value of RI_1 is the smallest, container 11 is assigned to column 1 according to H1 and is temporarily put in position 2 as shown in Fig.4(b). With the new configuration, we then attempt to put container 8 to each target column and calculate their RI again. When attempting to put container 8 to column 1, we can see that containers 8 can be feasibly put on top of container 11 and so RI_1 can be calculated in the normal way. Because the value of

RI_1 is the smallest, container 8 is assigned to column 1 according to H1 and temporarily put in position 3 as shown in Fig.4(c). With the updated configuration, we finally attempt to put container 5 to each target column and calculate their RI. When attempting to put container 5 to column 1, we know that if container 5 is put to this column, it must be put below containers 8 and 11 as shown in Fig. 4(d) because the position of container 5 in the original column is higher than that of containers 11 and 8 and so the value of RI_1 should be increased by 2 based on the rule of H4. Because the value of RI_4 is the smallest this time, Container 5 is assigned to column 4 and the positions of containers 11 and 8 are also confirmed. Therefore, the final storage positions of blocking containers 5, 11 and 8, when picking up container 1, are shown in Fig.4(e).

4.2 Worst case analysis

To the best of our knowledge, there exists no related research on the static reshuffling problem with worst case analysis, because the problem presents some dynamic feature. In this paper, we try to analyze the worst-case performance of the heuristics for the static reshuffling problem. Because the lower bound of the static reshuffling problem may be zero when its objective function is to minimize the number of reshuffles, it cannot be used to calculate the worst case performance ratio bound. In order to avoid this case, we consider the objective to minimize the total number of crane lifting moves for the static reshuffling problem. Crane lifting moves include the actions of retrieving and reshuffling containers. In this case, optimal solution $C^{*\prime}$ and approximate solution $C^{H\prime}$ are more than C^* and C^H respectively by S, where C^* and C^H are the optimal solution and approximate solution for the static reshuffling problem to minimize the total number of reshuffles.

Theorem 1. For the static reshuffling problem to minimize the total number of crane lifting moves, the worst-case performance ratio of any heuristic with the above framework (including the five new heuristics) is $P-\frac{P(P-1)}{2(S-1)}$, when $2(S-1)\geq P$.

Proof. When the last container in the given bay is picked up, the total number of crane lifting moves is equal to 1. When the second last container in the given bay is picked up, the total number of crane lifting moves is at most 2. Similarly, when the Pth container from the bottom in the given bay is picked up, the total number of crane lifting moves is at most P. The number of crane lifting moves from the second

container to the (S-P)th container is overestimated respectively to P. Assume the number of crane lifting moves for the first container is x. Therefore, an upper bound for this problem is equal to $1+2+\ldots+P+[S-(P+1)]P+x$, and a lower bound is equal to $S-1+x$. The worst-case performance ratio is as follows:

$$\frac{C^{H\prime}}{C^{*\prime}} \le \frac{1+...+P+[S-(P+1)]P+x}{S-1+x} = \frac{\frac{P(P+1)}{2}+(S-1+x)P-P^2-xP+x}{S-1+x}$$

$$= P + \frac{x-xP-P^2+\frac{P(P+1)}{2}}{S-1+x} = P + \frac{2x(1-P)+P(1-P)}{2(S-1+x)}$$

$$= P-(P-1)\frac{P+2x}{2(S-1)+2x}$$

If $2(S-1) \ge P$, then $\frac{P+2x}{2(S-1)+2x} \ge \frac{P}{2(S-1)}$. Thus, when $2(S-1) \ge P$, the following equation holds.

$$\frac{C^{H\prime}}{C^{*\prime}} \le P-(P-1)\frac{P+2x}{2(S-1)+2x} \le P-\frac{P(P-1)}{2(S-1)} \qquad \square$$

4.3 Special cases

Lemma 1. For the static reshuffling problem to minimize the total number of reshuffles, if the number of tiers is P=2, the objective values obtained by the developed five heuristics (C^H) are equal to the optimal objective value (C^*).

Proof. Note that the five heuristics are equivalent when the number of tiers is equal to 2, and we will prove Lemma 1 using heuristic H1. In this special case, each column has at most two containers. If the numbering of a top container is higher than that of the one below it, then reshuffling of the top container is unavoidable. Consider the retrieval process and the first time when such a reshuffle is to be made. H1 will put the reshuffled container to a column such that this container will not need to be reshuffled again in the future, if such a column exists. If such a column does not exist, then the reshuffled container has to be reshuffled again unavoidably. After this stage, there is always at least one empty column, and so H1 retrieves the remaining containers without further reshuffles except the unavoidable ones. Therefore, the reshuffles made by H1 are all unavoidable and so the solution is optimal. $\square$

Lemma 2. For the static reshuffling problem to minimize the total number of reshuffles, if the initial configuration of a bay which has C columns and P tiers

satisfies: 1) an empty column exists; 2) containers 1 to (C-1) are stored in the first tier, containers C to 2(C-1) are stored in the second tier, containers 2C-1 to 3(C-1) are stored in the third tier and so on, the objective values obtained by the proposed five heuristics (C^H) are equal to the optimal objective value (C^*).

Proof. Because there exists an empty column in the initial configuration of the bay and the numberings of the containers in each nonempty column are increasing from the bottom upwards, the blocking containers on container 1 will be reshuffled to the empty column according to the proposed heuristics and they can be retrieved in the future sequentially without reshuffling. At the same time, the retrieval of container 1 will create a new empty column. Similarly, when each of the first C-1 containers is picked up, the blocking containers will be reshuffled to the empty column and will not need any further reshuffling. Meanwhile a new empty column will appear. After the first C-1 containers are picked up, all the remaining containers can be picked up without reshuffling. Clearly, each of the reshuffles in the above process is unavoidable. Therefore, the total number of reshuffles is optimal. □

Lemma 3 below analyzes the worst-case performance ratio of heuristics H1 and H2 for some special configurations of a bay. Unlike in Theorem 1, the objective function here is the number of reshuffles because the lower bound of the static reshuffling problem is not zero in this case.

Lemma 3. For the static reshuffling problem to minimize the total number of reshuffles, if the initial configuration of a bay which has C columns and P tiers satisfies: 1) the total number of containers in the initial configuration is (C-1)P+1; 2) containers 1 to C are stored in the first tier, containers C+1 to 2C are stored in the second tier,…, containers C(P-1)+1 to (C-1)P+1 are stored in the Pth tier, the absolute performance ratio of heuristics H1 and H2 in this case is bounded by 3.

Proof. Like the situation in Lemma 2, all the containers in tier 2 and above have to be reshuffled in order to retrieve the containers in tier 1. Therefore, a lower bound of the optimal total number of reshuffles is LB = (C-1)P+1-C = (C-1)(P-1). Next, we will derive an upper bound UB of the optimal total number of reshuffles. Because containers 1 to C are stored in the first tier, the total number of reshuffles is at most C(P-1) when containers 1 to C are picked up. After retrieving container 1, an empty column will appear and all the containers blocking container 1 in the initial configuration will be reshuffled to other columns, one in a column. Let U be the set

of these columns. In this stage, considering the given initial configuration, the (numberings of) containers in each column are in increasing order (from bottom) upwards, except that the top container of each column in *U* may be not in order. Because container 1 may have at most *P*-1 blocking containers, we know that $|U| \le P-1$. When retrieving each of containers 2 to *C*, the blocking containers will be reshuffled to the empty column and the column of the container being retrieved will become empty. If the column of the container being retrieved is not in *U*, then the blocking containers will be in decreasing order upwards after being reshuffled to the new column and will be retrieved in the future without further reshuffling. If the column of the container being retrieved is in *U*, some blocking container may be reshuffled to a non empty column and in this case it must not block any container there. In the worst case, the blocking containers will be reshuffled to the empty column with the container originally blocking container 1 placed at the bottom and the other blocking containers placed above it in the decreasing order upwards. In the new column, the number of containers above the one originally blocking container 1 is at most *P*-2. Hence, in the worst case, retrieving the containers originally blocking container 1 will need at most (*P*-1)(*P*-2) reshuffles. After retrieving each of these containers (at most *P*-1), the containers blocking it will be in a new column in increasing order upwards. Retrieving the bottom one in the new column will need at most *P*-3 reshuffles and the reshuffled containers will be in decreasing order upwards in another new column and can be retrieved in the future without further reshuffling. Thus an upper bound of the optimal total number of reshuffles is UB= $C(P-1)+(P-1)(P-2)+(P-1)(P-3)$

For these special cases, the worst-case performance ratio is then:

$$\frac{C^H}{C^*} \le \frac{\text{UB}}{\text{LB}} = \frac{C(P-1)+(P-1)(P-2)+(P-1)(P-3)}{(C-1)(P-1)} = \frac{C-1+(P-2)+(P-2)}{(C-1)}$$

$$\le 1+\frac{P-2}{C-1}+\frac{P-2}{C-1} \le 1+1+1=3 \qquad \square$$

Fig.5 shows an example of the situation described in Lemma 3 and the process of retrieving the containers in the bay (the stages without reshuffling are omitted). It can be seen from the figure that the actual reshuffles in each stage are no more than the upper bound calculated in the proof.

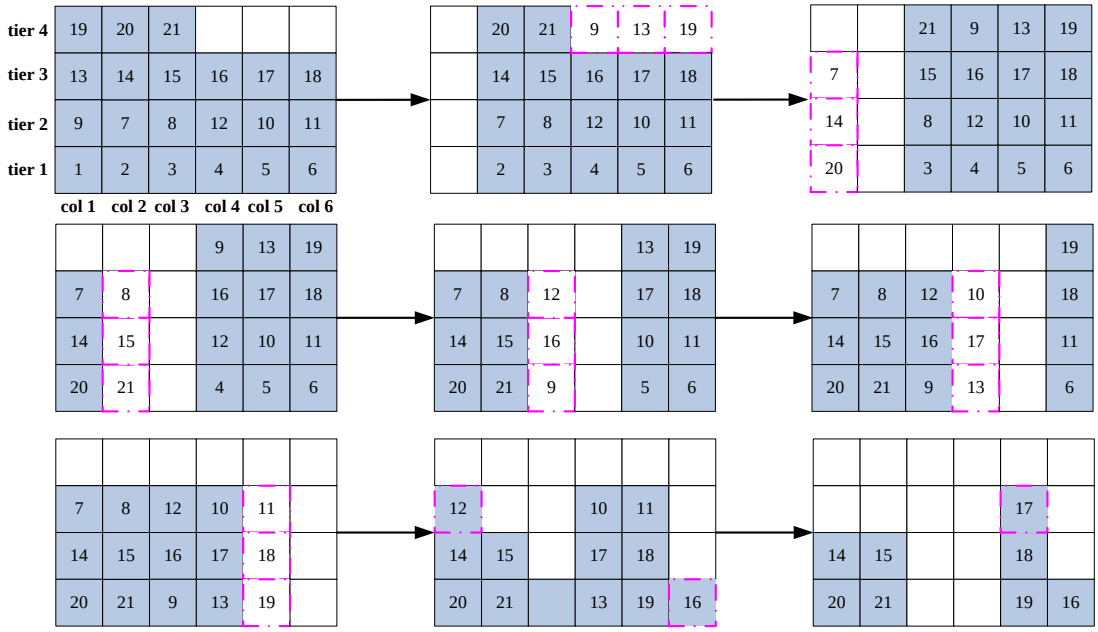

Fig. 5 An example of special configurations in Lemma 3

5. Simulation model

To evaluate the effectiveness and efficiency of different reshuffling and storage methods in a dynamic situation, we develop a discrete-event simulation model specifically to simulate the dynamic process of container storage and retrieval in a container bay. To ensure that the same series of containers are used for testing all methods, which insures for fair comparison, we follow most previous studies to separate the problem generation from the simulation.

According to the data in container terminal operations, the interarrival and dwell times of containers in a container yard follow exponential distributions, For any given problem setting with parameters *C*, *P* and space utilization, the data generation program is designed to randomly generate a series of containers with appropriate interarrival and dwell times and then the arrival and departure times of each container can be calculated accordingly. In case the bay is full when a new container arrives, this container is diverted to another bay and, therefore, removed from the problem data. The simulation model is then run to test any reshuffling and storage method using the data. Details of the simulation model are described in the following.

5.1 Input data

The input data include the structure of the bay, a series of containers with their arriving and departure times randomly generated as mentioned above, the warm-up period and the resulting collection period. The structure of the bay is defined by the

number of tiers and columns. The warm-up and result collection periods are both in terms of number of containers retrieved. According to the arriving and departure times of the generated container series, the stacking and retrieval sequences of containers are identified.

5.2 Numbering of the containers and the events in the simulation

We are most interested in the number of reshuffles, so the time needed for actual container moves are ignored. Because those containers arriving when the bay is full overflow to other bays, there will be no queues observed in the system apart from the containers staying in the bay. This reduces the types of events in the system. The simulation model contains two types of events: container arrival and container departure.

Comparing the arriving time of the next container with the departure times of all the containers in the bay, if the departure time of a container in the bay is earlier than the arriving time of the next container, the next event is a container departure. Recall that the departure times of the containers in the bay are known and the containers are numbered in ascending order of their departure times with the departing container as No. 1. If the departing container is not blocked by other containers, it will be retrieved directly. If the container is blocked by some containers, these blocking containers need to be reshuffled and their new positions will be determined using the reshuffling and storage decision method. When using any index based heuristic as the decision method, for each blocking container, the index is calculated for each available column and the new storage position of the container is chosen based on the rule of this heuristic. When using an IP model as the decision method, a model is formulated as if all the containers in the bay are to be retrieved assuming there is no container arrival in the process. Solving the model we can determine the new storage positions for all the containers blocking the departing container. Based on the decision, the blocking containers are moved to the new positions and the departing container is retrieved. In either case, the remaining containers are renumbered, the total number of containers in the bay is decreased by one, and the bay configuration is updated.

Conversely, if the arriving time of the next container is earlier than the departure time of any container in the bay, the next event is a new container arrival. In this case, the new container and all the containers in the bay are first renumbered from

No.1 in ascending order of their departure times, and the total number of containers in the bay is increased by one. Then, the reshuffling and storage decision method is called to assign a storage position in the bay for the new container. When using any index based heuristic as the decision method, the decision process is similar to that for a reshuffled container, i.e., the index is calculated for each available column and the storage position of the new container is chosen based on the rule of this heuristic. When using an IP model as the decision method, for each available column, assuming that the new container is put in this column, we formulate and solve a corresponding model, as if all the containers in the bay are to be retrieved without any further container arrival in the process, to obtain the total number of reshuffles needed. The new container is then assigned to the column with the minimum total number of reshuffles.

5.3 Output data

The final output of the simulation mainly includes the total number of reshuffles in the result collection period and the average number of reshuffles per retrieval, which is the ratio of the total number of reshuffles to the total number of containers retrieved in the result collection period. During the simulation, the configuration of the bay at any moment can be output if it is needed.

5.4 Interface

In a dynamic situation, the simulation model invokes the reshuffling and storage method being tested by passing the configuration of the container bay to the method and receiving the new state of the bay returned from the method. Therefore, with the correct input and output settings, any new decision method can be tested using the simulation system.

5.5 Animation display

The simulation model has an animation function. If we want to directly see how a container will be moved, we can switch on the animation function to display every detail for stacking, retrieving and reshuffling and observe the dynamic changes in the bay configuration. The animation function can simultaneously display the animations for multiple heuristics in multiple windows in addition to being displayed alone. The animation screen can be paused or closed at any time as needed. Fig. 6(a) to Fig. 6(f) illustrate the container numbering and retrieving, reshuffling and stacking operations in the dynamic environment for six successive states of a bay at

different times when one heuristic is used.

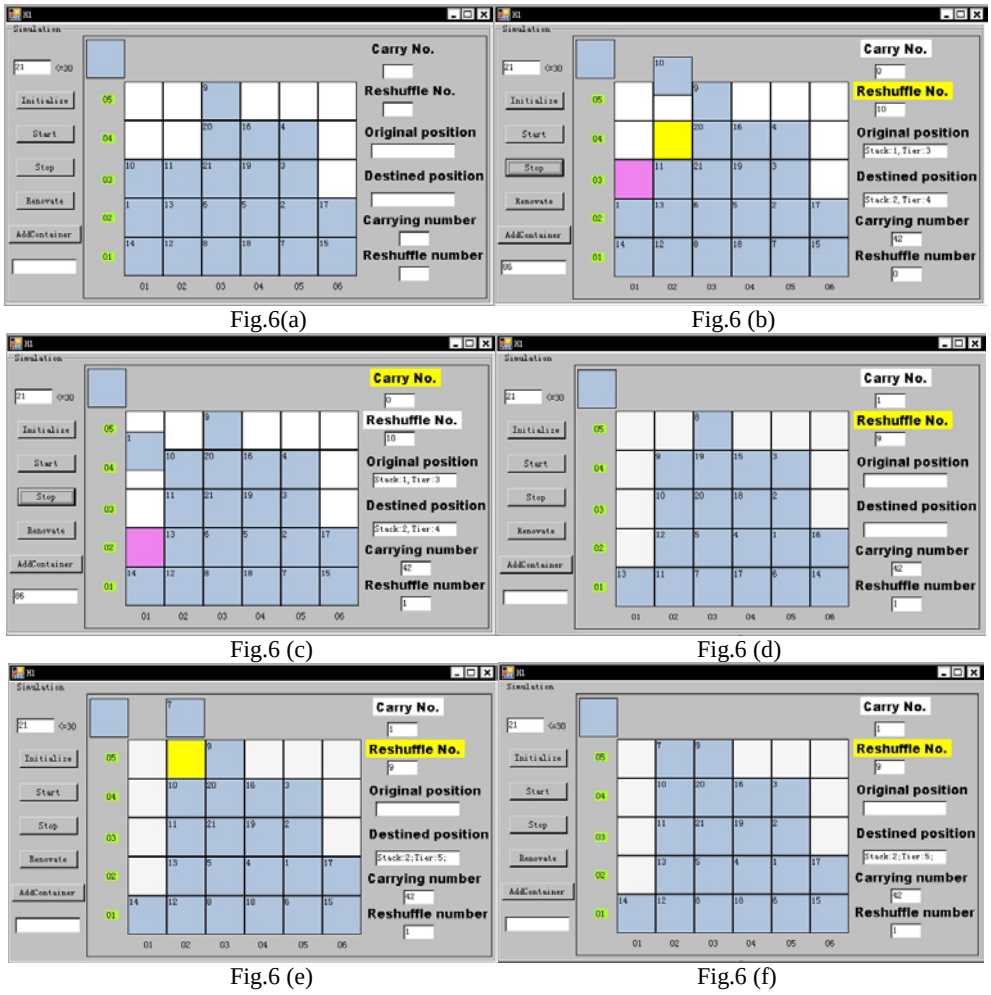

Fig.6(a) Fig.6 (b)

Fig.6 (c) Fig.6 (d)

Fig.6 (e) Fig.6 (f)

Fig. 6. The dynamical changes of the bay configuration

Fig.6(a) is a configuration of the bay at certain time point and all the containers are numbered from No.1 in ascending order of their departure times. Container 1 represents the first container to be picked up, but it is blocked by container 10. Fig.6(b) shows the reshuffling of container 10 to its new storage location in tier 4 of column 2. Fig.6(c) shows container 1 is being retrieved. The remaining containers in the bay are renumbered from No.1 in ascending order of their departure times, i.e. decreasing the number by one for all the containers, and the new configuration is shown in Fig.6(d). The next event is the arrival of a new container, which has the seventh earliest departure time among the containers in the bay and so will be numbered as No.7. The existing containers with later departure times will be renumbered, i.e., increasing the numbering for each of these containers by one, as shown in Fig.6(e). The storage location of the new arrival container is shown in Fig.6(f).

6. Computational experiments

6.1 Computational results for the static reshuffling problem

Comparison of the MRIP model and the improved model

In this section, we verify the effectiveness of our model. In order to avoid deadlock, at least P-1 empty positions need to be reserved for reshuffling. Therefore, as indicated in Wan *et al*. (2009), the storage capacity for a bay with C columns and P tiers is $(C\text{-}1)P+1$. We define the utilization of a bay as the percentage of its capacity being occupied by containers stored in it. A bay with $(C\text{-}1)P+1$ containers stored in it will be said to have a utilization of 100%. When the utilization of the bay is given, the total number of containers stored in a bay is equal to $[(C\text{-}1)P+1]\times$(the utilization). In general, the higher the utilization of a bay is, the fewer empty positions it has for reshuffling and the higher chances the reshuffled containers have to be reshuffled again. We consider the common bay structures (6 columns and 2 to 5 tiers) in the experiment. For each bay structure, we generate two classes of problem with bay utilizations of 80% and 100%, respectively. We use "the number of columns-the number of tiers-the number of containers stored in a bay" to represent the problem class. For each problem class, 50 instances are generated randomly. The MRIP model and the improved model are coded in C++ and solved using CPLEX 11.0, which is a commercial software package, on a computer with 2.83GHz Intel Core 2 CPU and 3.25GB RAM. We set a time limit (one hour) for solving each problem because some instances would be extremely time consuming. Table 1 shows the percentage of instances optimally solved within the one-hour time limit as well as the average computation time spent by the models for each problem class.

Table 1 Comparison between the MRIP model and the improved model ILP

Problem class (80%)		MRIP model	ILP model	Problem class (100%)		MRIP model	ILP model
6-2-9	%Opt	100%	100%	6-2-11	%Opt	100%	100%
	Time	0.084	0.079		Time	0.138	0.124
6-3-13	%Opt	100%	100%	6-3-16	%Opt	100%	100%
	Time	0.342	0.294		Time	0.680	0.547
6-4-17	%Opt	100%	100%	6-4-21	%Opt	100%	100%
	Time	16.11	2.71		Time	135.36	50.88
6-5-21	%Opt	98%	100%	6-5-26	%Opt	48%	58%
	Time	253.87	69.39		Time	783.13	201.97
	%Fea	100%	100%		%Fea	56%	90%

"%Opt" = the percentage of instances for which an optimal solution was obtained within the time limit.
"Time" = the average CPU time for instances that were solved optimally by both models.
"%Fea" = the percentage of instances where a feasible solution was obtained within the time limit.

As shown in Table 1, both models obtained optimal solutions for all instances in

all the problem classes except “6-5-21” and “6-5-26”, but the improved model consumes less time. For “6-5-21”, there was one instance for which the MRIP model did not obtain an optimal solution within one hour while the improved model ILP obtained an optimal solution for all instances within the time limit. Problem class “6-5-26” is the most difficult to solve because the number of tiers is the greatest and the utilization of the bay is the largest. For this problem class, the MRIP model found a feasible (integer) solution for 56% of the instances within one hour, which included 48% of the instances solved optimally, while the corresponding figures for the improved model ILP were 90% and 58% respectively. In other words, the improved model ILP obtained a feasible solution for 34% more instances and an optimal solution for 10% more instances in this class within the time limit. For each problem class, Table 1 also shows the average CPU time over the instances that were solved optimally by both models. The results show that the improved ILP model takes a shorter amount of time to obtain an optimal solution especially for larger problems where it only takes one quarter to one third of the time taken by the MRIP model. The optimal solutions obtained by the improved ILP model will be used as a benchmark to evaluate the performance of the heuristic rules.

Comparison of the different heuristics

The eight problem classes mentioned above are also used to compare the five new heuristics with the existing heuristics RI and ENAR. Table 2 shows the average number of reshuffles in the solution of each heuristic for each problem class. Because the heuristics consume very little time, their CPU times are not presented here.

In Table 2, the first column lists the problem classes tested, column “Opt” gives the average number of optimal reshuffles, and the other columns report the average numbers of reshuffles in the solutions for each heuristic. For problem class “6-5-26”, the place for “Opt” is left empty because optimal solutions were not obtained for some instances within one hour.

Table 2 Comparison of RI, ENAR and our heuristics

Problem class (80%)	Opt	RI	ENAR	H1	H2	H3	H4	H5
6-2-9	1.84	**1.84**	**1.84**	**1.84**	**1.84**	**1.84**	**1.84**	**1.84**
6-3-13	4.32	4.4	4.34	**4.32**	**4.32**	4.36	**4.32**	**4.32**
6-4-17	7.64	8.06	7.92	7.8	**7.7**	7.88	**7.7**	7.74
6-5-21	11.04	12.14	12	11.7	**11.52**	11.78	11.58	11.64

Problem class (100%)	Opt	RI	ENAR	H1	H2	H3	H4	H5
6-2-11	2.74	**2.74**	**2.74**	**2.74**	**2.74**	**2.74**	**2.74**	**2.74**
6-3-16	6.8	**6.88**	6.96	**6.88**	**6.88**	6.96	6.96	6.96

6-4-21	12.18	13.08	13.04	**12.7**	12.72	12.96	12.96	12.82
6-5-26		20.68	20.26	19.6	19.78	**19.56**	19.74	19.96

From the results in Table 2, we can see that the proposed new heuristics have a better performance in most cases compared with RI and ENAR. The best heuristic result for each problem class is highlighted in bold in Table 2. It can be observed that H1 shows the best performance for the instances where the bay has a 100% utilization, while H2 shows the best performance for those with an 80% utilization. Furthermore, the computational results for "6-2-9" and "6-2-11" further verify the conclusions of Lemma 1.

We apply an idea of extension to the five new heuristics and the existing heuristics and test the performances of all the extended heuristics on the problem instances mentioned above. The idea of the extended version Wan *et al*. (2009) for an original heuristic is as follows: a reshuffled container is tested for each potential and feasible location, and the location with the minimum number of reshuffles needed to empty the bay using the original heuristic is selected to store the reshuffled container. Break ties by putting the reshuffled container into the column determined by the original heuristic. Table 3 shows the average number of reshuffle results of different extended heuristics. In this table, the extended version of original heuristic * is denoted as *_E.

Table 3 Comparison results among different extended heuristics

Layout (80%)	Opt	RI_E	ENAR_E	H1_E	H2_E	H3_E	H4_E	H5_E
6-2-9	1.84	**1.84**	**1.84**	**1.84**	**1.84**	**1.84**	**1.84**	**1.84**
6-3-13	4.32	**4.32**	**4.32**	**4.32**	**4.32**	**4.32**	**4.32**	**4.32**
6-4-17	7.64	7.7	7.68	7.66	**7.64**	7.66	7.66	**7.64**
6-5-21	11.04	11.3	11.16	11.1	11.12	11.1	**11.08**	11.1

Layout (100%)	Opt	RI_E	ENAR_E	H1_E	H2_E	H3_E	H4_E	H5_E
6-2-11	2.74	**2.74**	**2.74**	**2.74**	**2.74**	**2.74**	**2.74**	**2.74**
6-3-16	6.8	**6.8**	6.82	**6.8**	**6.8**	**6.8**	**6.8**	**6.8**
6-4-21	12.18	12.36	12.34	**12.22**	12.24	12.3	12.26	**12.22**
6-5-26		18.78	18.7	**18.52**	18.6	18.56	**18.52**	18.54

Comparing the results in Tables 2 and 3, we can see that the extended heuristics give better solutions than the corresponding original heuristics. From the results in Table 3, it can also be observed that the performance differences between the extended heuristics are reduced when compared with the original heuristics. For the problem classes where extended heuristics do not perform the same, the extended versions of our proposed heuristics show superior performance to that of RI and ENAR. Among all the extended heuristics, "H1_E" and "H4_E" show the best performance in most cases.

In addition to the above problem instances, we have also tested the problem instances used in Wan *et al*. (2009). The experimental results show that the improved model can obtain optimal solutions or feasible solutions more quickly than the MRIP model proposed by Wan *et al*. (2009). This confirms the results on the instances we generated. The performance of the five proposed heuristics is superior to or the same as that of the existing heuristics. This conclusion also applies to their extended versions. The conclusions from the results on these problem instances are again similar to those on the instances we generated. The detailed comparison results are not included in our paper because of the space limitation.

Furthermore, we also tested our extended heuristics on the larger scale instances on problem data given by Caserta *et al*. (2011). The results are also compared to that of the corridor method (CM) in Caserta *et al*. (2011). The details of the comparison results can be seen in Table 4. As mentioned in Caserta *et al*. (2011), $h \times m$ is used to represent a bay size, where h is the number of tiers, m is the number of stacks and $h \times m$ is the total number of containers in the bay. The percentages in Table 4 are the utilization of its corresponding bays.

Table 4 Comparison results of CM and our extended heuristics

Bay size	CM	H1_E	H2_E	H3_E	H4_E	H5_E
6×6 (87%)	**32.4**	32.8	33.325	32.925	32.875	32.975
6×10 (82%)	49.5	**47.1**	47.5	47.4	47.15	47.625
10×6 (95%)	102.0	**86.925**	90.675	87.4	87.7	93.65
10×10 (90%)	128.3	**121.55**	126.425	122.825	122.575	134.375

The experimental results show that the extended heuristics are better than the corridor method in almost all cases, and the advantage of the extended heuristics is more obvious with the increase of the utilization of a bay.

6.2 Dynamic simulation results with incoming containers

The ultimate evaluation of any reshuffling method is whether it can be effectively applied to the dynamic environment with continual arrivals and retrievals of containers. Therefore, we will test and compare the performance of the above seven heuristics and their extended versions in the dynamic environment in a container bay. In this case, the simulation system needs to use the specified method to make not only the reshuffling decisions when retrieving each container but also the stacking decision when each new container arrives. In addition, we will also compare the performance of the above methods with that of the MRIP-based heuristics proposed in Wan *et al*. (2009). In a MRIP-based heuristic, the decisions are made using a reduced MRIP

model where the total number of reshuffles needed to retrieve the first K containers is minimized. Wan *et al*. (2009) show that the MRIP-based heuristics with K=5,6,7,8 (denoted as MRIP model-D_K) give better solutions than RI and ENAR. Therefore, we use MRIP model-D_K (K=5,6,7,8) in the comparison. We also test the model-based heuristics with the MRIP model replaced by the improved ILP model, which will be referred to as ILP-based heuristics.

General setup for experiments

In the dynamic experiments, the common bay structures (6 columns and 2 to 5 tiers) are considered, and the numbers of the available tiers should be specified as needed before an experiment. Here the average storage space utilization of a container bay is set to 75% to simulate actual terminal yards with a higher storage space utilization.

Simulation results

To evaluate the long-term performance of the different stacking and reshuffling methods in the dynamic environment, the simulation experiments should be performed for the four common bay configurations mentioned above for a longer period. For each bay configuration, we first generate ten independent tested data sets according to the data generation rule mentioned in Section 5; second, the simulation experiments based on the ten independent tested data sets are run until a total of 1000 containers are retrieved within the bay, and then, the total number of reshuffles and the total CPU time needed to retrieve 1000 containers is obtained for each data set. Finally, based on the results of the simulation experiments for each bay configuration, the average total number of reshuffles and the average total CPU time for the ten data sets are obtained and listed in Table 5. Here we performed ten simulation experiments for each bay configuration where 1000 containers were retrieved to test the universality and long-term performance of all the heuristics using a large number of reshuffling operations.

The simulation results of all the heuristics can be observed in Table 5.

Table 5. The average total number of reshuffles and the average total CPU time

	(Average total number of reshuffles, average total CPU time in seconds)			
	(C,P)=(6,2)	(C,P)=(6,3)	(C,P)=(6,4)	(C,P)=(6,5)
RI	(126, 0.35)	(251.5, 0.657)	(433, 0.825)	(626.5, 0.93)
ENAR	(127.2, 0.365)	(246.7,0.702)	(439.1, 6.53)	(654,73.1)
H1	(125.7, 0.416)	(247.8,0.582)	(416.3, 0.629)	(600, 0.908)
H2	(125.7, 0.429)	(244, 0.569)	(414.2, 0.762)	(606, 1.095)
H3	(125.7, 0.369)	(250.1, 0.581)	(421.4, 0.657)	(620.2,0.859)
H4	(125.7, 0.357)	(245.2, 0.639)	(420, 0.708)	(612.3, 1.017)

H5	(125.7, 0.414)	(246.7, 0.633)	(420.7, 0.681)	(600.9, 0.983)
RI_E	(131.1, 0.321)	(252, 0.812)	(422.4, 1.125)	(593.4, 3.788)
ENAR_E	(127.8, 1.515)	(249.9, 1.918)	(418.7,16.95)	(608, 245.641)
H1_E	**(124.1, 0.411)**	(244.6, 0.801)	(407.4, 1.662)	(591.5, 3.280)
H2_E	(125.7, 0.441)	**(243.1, 0.733)**	**(406.1, 1.808)**	**(588.7, 3.365)**
H3_E	**(124.1, 0.463)**	(245.1, 0.908)	(414, 1.482)	(589, 3.430)
H4_E	**(124.1, 0.496)**	(245.1, 0.956)	(417.6, 2.354)	(589.9, 5.167)
H5_E	(125.7, 0.497)	(243.1, 1.254)	(413.2, 1.892)	(589.2, 6.341)
MRIP model-D_5	(129.6, 107.690)	(250.7, 347.908)	(433.2, 863.19)	(657.2, 2543.465)
ILP model-D_5	(127.2, 100.783)	(256.2, 326.394)	(434, 806.244)	(640.1, 1731.957)
MRIP model-D_6	(131.6, 113.840)	(252.6, 382.539)	(433.8, 1012.517)	(618.9, 2587.367)
ILP model-D_6	(131.9, 106.238)	(252.1, 355.771)	(430, 910.989)	(623.9, 2529.296)
MRIP model-D_7	(127.7, 117.636)	(247.6, 411.015)	(432.5, 1172.624)	(622.4, 4604.497)
ILP model-D_7	(129.6, 108.370)	(252.4, 379.292)	(421.6, 1038.684)	(623.6, 3317.614)
MRIP model-D_8	(128, 119.714)	(244.9, 433.148)	(418.1, 1370.753)	(626.9, 8245.435)
ILP model-D_8	(131, 110.819)	(255.5, 398.506)	(429.2, 1116.538)	(624.1, 4504.546)

As shown in Table 5, the best heuristic result for each bay configuration is highlighted in bold. It can be seen that H1_E to H5_E are superior or similar to the best results of the existing heuristics listed in Table 5 and consume very little time. Among these five heuristics, H2_E has the best performance in most of the tested bay structures. Recall that H2_E performed the best for problems 80% utilization of a bay in the static environment. Since the average utilization of a bay is similar in the dynamic experiment, the excellent performance of H2_E can be expected. At the same time, we can observe that the performances of the ILP-based heuristics are close to that of the MRIP-based heuristics, but ILP-based heuristics take much less time, which is consistent with the results obtained by the static experiment.

7. Conclusions

In this paper, we have studied two different, but related, problems, which are the static reshuffling problem and the dynamic stacking problem in container terminal yards. For the static reshuffling problem, an improved static reshuffling model was formulated by removing the column-relationship variables and some associated constraints from an existing model and introducing new reshuffling-related constraints for individual columns. Five new effective heuristics and their extended versions were also developed and the worst performance was analyzed. For the dynamic problem with continual arrivals and retrievals of containers, the different heuristics of the static environment were applied and tested, and a simulation model was developed with an animation function to show the stacking, retrieving and reshuffling operations if needed. The experimental results have shown that the improved model can obtain optimal or feasible solutions more quickly than the existing model, and that the extended versions of the five proposed heuristics are

superior or similar to the best results of the existing heuristics and consume very little time for both the static and the dynamic problems.

Acknowledgements

This research is partly supported by the Fund for Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 71321001) and the State Key Program of the National Natural Science Foundation of China (Grant No.71032004).

References

Borgman, B., Van Asperen, E., & Dekker, R. (2010). Online rules for container stacking. *OR Spectrum, 32*, 687-716.

Caserta, M., Schwarze, S., & Voss, S. (2009). A New Binary Description of the Blocks Relocation Problem and Benefits in a Look Ahead Heuristic. *Evolutionary Computation in Combinatorial Optimization, Proceedings*, 5482, 37-48.

Caserta, M., Voss, S., & Sniedovich, M. (2011). Applying the corridor method to a blocks relocation problem. *OR Spectrum, 33*, 915-929.

Caserta, M., Schwarze, S., & Voss, S. (2011). Container rehandling at maritime container terminals. *Handbook of terminal planning*, 49, 247–269. New York: Springer.

Duinkerken, M. B., Evers, J. J. M., & Ottjes, J. A. (2001). A simulation model for integrating quay transport and stacking polices on automated container terminals. *Modelling and Simulation 2001*, 909-916.

Forster, F., & Bortfeldt, A. (2012). A tree search procedure for the container relocation problem. *Computers & Operations Research*, 39, 299-309.

Hartmann, S. (2004). Generating scenarios for simulation and optimization of container terminal logistics. *OR Spectrum, 26*, 171-192.

Kang, J. H., Ryu, K. R., & Kim, K. H. (2006). Deriving stacking strategies for export containers with uncertain weight information. *Journal of Intelligent Manufacturing, 17*, 399-410.

Kim, K. H. (1997). Evaluation of the number of rehandles in container yards. *Computers & Industrial Engineering, 32*, 701-711.

Kim, K. H., & Hong, G. P. (2006). A heuristic rule for relocating blocks. *Computers & Operations Research, 33*, 940-954.

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*, 89-101.

Klaws, J., Stahlblck, R., & Voss, S. (2011). Container Terminal Yard Operations – Simulation of a Side-Loaded Container Block Served by Triple Rail Mounted Gantry Cranes. *Lecture Notes in Computer Science*, 6971, 243–255.

Lee, Y., & Lee, Y.-J. (2010). A heuristic for retrieving containers from a yard. *Computers & Operations Research, 37*, 1139-1147.

Murty, K. G., Liu, J. Y., Wan, Y. W., & Linn, R. (2005). A decision support system for operations in a container terminal. *Decision Support Systems, 39*, 309-332.

Park, B. J., Choi, H. R., Kwon, H. K., & Kang, M. H. (2006). Simulation analysis on effective operation of handling equipments in automated container terminal. *AI 2006: Advances in Artificial Intelligence, Proceedings, 4304*, 1231-1238.

Petering, M. E. H. (2010). Development and simulation analysis of real-time, dual-load yard truck control systems for seaport container transshipment terminals. *OR Spectrum, 32*, 633-661.

Sgouridis, S. P., & Angelides, D. C. (2002). Simulation-based analysis of handling inbound container in a terminal. *2002 Winter Simulation Conference,2*, 1716-1724.

Stahlbock, R. & Voss, S. (2010). Efficiency considerations for sequencing and scheduling of double-rail-mounted gantry cranes at maritime container terminals. *International Journal of Shipping and Transport Logistics (IJSTL)*, 2, 95-123.

Wan, Y. W., Liu, J. Y., & Tsai, P. C. (2009). The Assignment of Storage Locations to Containers for a Container Stack. *Naval Research Logistics, 56*, 699-713.

Zhang, C. Q. (2000). Resource planning in container storage yards. *Unpublished PhD. Thesis*, The Hong Kong University of Science and Technology, Hong Kong.

Zhang, C. Q., Liu, J. Y., & Wan, Y. W. (2003). Storage space allocation in container terminals. *Transportation Research Part B-Methodological, 37*, 883-903.

Zhang, R. Z. (2010). Throughput statistics of world's top 50 container terminals over the years. *China Ports, 11*, 62.