Transportation Research Part E 103 (2017) 17–31

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

# Optimizing container relocation operations at container yards with beam search

Ching-Jung Ting [*], Kun-Chih Wu

*Department of Industrial Engineering and Management, Yuan Ze University, 135 Yuan-Tung Road, Taoyuan 32003, Taiwan, ROC*

A R T I C L E   I N F O

*Article history:*
Received 3 November 2016
Received in revised form 21 April 2017
Accepted 23 April 2017
Available online 29 April 2017

*Keywords:*
Container relocation
Logistics
Beam search
Container yard

A B S T R A C T

Container relocation problem (CRP) involves the retrieval of all containers from the container yard with a minimum number of relocations. The CRP is an NP-hard problem such that the large-scale instances cannot be solved to optimality by exact solution methods within a reasonable computational time. This article proposes a beam search (BS) algorithm embedded with heuristics to evaluate the problems. The proposed beam search is tested on benchmark instances and compared with other leading heuristics from the literature. Computational results demonstrate that the beam search algorithm is compatible with other heuristics and can obtain good solutions in short time.

## 1. Introduction

Recent increases in the demand for maritime transportation of containers have led to the construction of numerous mega container vessels, many of which have the capacity to carry more than 18,000 twenty-foot equivalent units (TEU) containers. The challenges faced at container terminals in handling this enormous traffic enforce the development of efficient strategies to increase terminal productivity and reduce operational costs.

This study considers an unloading problem from the perspective of relocating containers within the container yard. Containers are usually stored in stacks to maximize spatial utilization, particularly in terminals with limited storage space. As shown in Fig. 1, a *stack* is obtained by stacking containers vertically. Several stacks in a row form a *bay*, and bays are placed side by side to form a *container block*.

Relocation involves the movement of a container from one stack to another, which must be performed whenever a container is located atop a desired container that is immediately required (the highest priority). We called this immediately desired container as the *target container*. Relocating containers cannot be avoided because the locations of containers cannot always be arranged according to their retrieval priorities. The retrieval priorities for export containers depend on their loading sequence onto the vessel. However, outside trucks transporting export containers arrive at the container terminal in random and hence, containers are not necessarily placed within the bay according to the loading sequence. On the other hand, the stowage plan provided by the shipping companies regarding an export container at the time of its arrival might be inaccurate or changed. Clearly, relocations are non-value added activities that must be minimized to increase the efficiency of operations in the container terminal and impair the customer satisfaction. In this paper, we investigate the container relation problem for export containers which must be retrieved in a predetermined order based on the vessel's stowage plan.

* Corresponding author.
*E-mail addresses:* ietingcj@saturn.yzu.edu.tw (C.-J. Ting), s968907@gmail.com (K.-C. Wu).

http://dx.doi.org/10.1016/j.tre.2017.04.010
1366-5545/© 2017 Elsevier Ltd. All rights reserved.

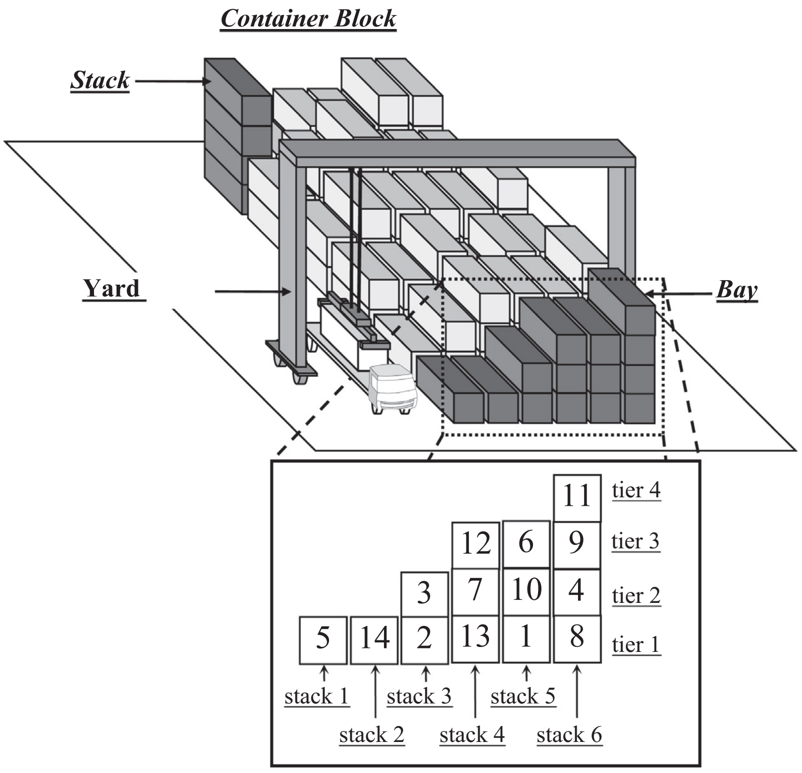

**Fig. 1.** Stacking containers in bays and container blocks.

The unloading problem can be classified as restricted or unrestricted problem according to the moving fashion of relocated containers. In the restricted problem, only the blocking containers above the target container can be relocated. On the other hand in the unrestricted problem, voluntary moves (or cleaning moves) of the containers that located at other stacks are allowed to make. This study focuses on the restricted problem, in which containers are retrieved according to a predetermined order with the minimum number of relocations.

To deal with the restricted relocation problem, we first propose a new novel heuristic to estimate the upper bound of the CRP by considering all the blocking containers above the target container. This heuristic is compared with other state-of-the-art algorithms on benchmark instances. The heuristic is embedded in a beam search (BS) algorithm as the evaluation heuristic. The structure of the beam search is similar to the tree search algorithm, but only permitted nodes is reserved in the search procedure. The permitted nodes are chosen depend on the proposed evaluation heuristic, which determines the location of relocated container with a changeable sequence. The performance of the BS is validated through experiments of benchmark instances from the literature.

The remainder of this study is organized as follows. Related works are reviewed in Section 2. Section 3 defines the container relocation problem. Section 4 introduces evaluation heuristics and Section 5 describes the beam search method embedded with the evaluation heuristics in Section 4. Section 6 presents experimental results drawn from a wide variety of scenarios. Finally, in Section 7, we summarize our findings.

## 2. Related literature

Container relocation in container terminals has attracted considerable attention since the 1980s. Reviews on related issues of container relocation problem can be found in Carlo et al. (2014) and Lehnfeld and Knust (2014). The former article focused on the layout, the material handling equipment, and other characteristics for the storage problem. The latter article reviewed the optimization problems specifically on the loading and unloading operations. Caserta et al. (2011a) provided a thorough overview on the rehandling of containers at maritime container terminals. We refer to the above articles for details.

Kim (1997) proposed a methodology to evaluate an expected number of relocations when a specific container is retrieved. The approach can solve the problem with the number of stacks ranges from 2 to 10 and from 4 to 12 for the number of tiers. Kim and Kim (1999) expanded this work by allocating import containers according to a segregating policy, in which containers are unloaded from the vessel to an empty slot. They defined the problem as a search for the stacking height which minimizes the expected number of relocations. Two mathematical models were formulated, one for import containers with cyclic arrival rates and the other for random arrival rates. In an attempt to minimize the number of relocations in an export container yard, Kim et al. (2000) classified assigned containers' precedence according to their weights, under the assumption that the probability of each type's arrival is known in advance and remains steady throughout the unloading process. They proposed a dynamic programming approach to optimize the stacking process by minimizing the expected number of relocations. Zhang et al. (2010) corrected an error that was found in the model transformation derived by Kim et al. (2000).

Murty et al. (2005) provided a decision-support system (DSS) for the Hong Kong International Terminals (HIT). The problem of relocation in the allocation of import containers was solved using a simple rule called the reshuffle index (RI). Yang and Kim (2006) addressed the general concept of the relocation problem. However, they assumed that items would be permanently removed from the storage area whenever they were relocated.

Kim and Hong (2006) was one of early studies for the unloading problem at container yards. Their objective was to minimize the number of relocations when all containers in a bay must be retrieved according to a predetermined precedence. They proposed a branch-and-bound algorithm and a heuristic to estimate the expected number of additional relocations during the retrieval process. Wan et al. (2009) formulated an integer programming model for the retrieval problem and developed a heuristic based on the model capable of obtaining near-optimal solutions efficiently. Later, Wu et al. (2010) developed a tabu search algorithm in which the solution is in changeable length. Computational results were compared with those of branch-and-bound for small size problems. Wu and Ting (2010) extended the reshuffle index heuristic of Murty et al. (2005) by giving more consideration to break ties with a look-ahead mechanism (RIL). A simple branch and bound was also developed for comparing the efficiency of the new heuristic. Caserta et al. (2011b) dealt with the CRP using an inspired heuristic, called corridor method (CM), which is based on the dynamic programming algorithm with additional restrictions on the solution space. Only the limited neighbourhood (or corridor) of the current bay configuration was explored.

Zhu et al. (2012) proposed an iterative deepening $A^*$ algorithm ($IDA^*$) for both restricted and unrestricted problems. Several heuristic algorithms for generating feasible solutions and two lower bounds were also developed as subroutines for $IDA^*$. Caserta et al. (2012) proved that the relocation problem is NP-hard by converting the CRP to a mutual exclusion scheduling (MES) problem, which is NP-complete. They also proposed a heuristic for the restricted CRP, and two mathematical models: BRP-I and BRP-II for the unrestricted and restricted relocation problems, respectively. Expósito-Izquierdo et al. (2014) developed an $A^*$ algorithm to obtain the optimal solution for both the restricted and unrestricted CRP. They also proposed a domain-specific knowledge-based heuristic, in order to obtain the good solution efficiently compared to the existing methods.

Forster and Bortfeldt (2012) tackled an unrestricted container relocation problem, in which containers are classified into separate groups and every container in a given group has the same retrieval priority. A lower bound and a tree search heuristic were proposed to deal with the problem. Petering and Hussein (2013) formulated an improved integer programming model, called BRP-III, with less number of decision variables for the unrestricted problem. Though BRP-III provides a less tight lower bound, it can achieve the optimal solution faster than BRP-I by branch-and-bound algorithm. The authors also developed a look-ahead heuristic called LA-N via extending the heuristic in Caserta et al. (2012). Jin et al. (2015) considered the unrestricted relocation problem of group containers, in which the containers in the same group have the identical retrieval priority. The authors addressed the problem by a look-ahead heuristic in a greedy fashion.

Several recent studies provided algorithms for the restricted problems. Jovanovic and Voss (2014) proposed a chain heuristic that extended the heuristic developed by Caserta et al. (2012). Their method further considered the next container that will require a relocation when relocating the top container. Zehendner and Feillet (2014) presented a branch and price algorithm for the restricted relocation problem, based on the BRP-I proposed by Caserta et al. (2012). The column generation is applied to overcome the drawback of BRP-I that required a huge number of decision variables as the size of the problem increases. Expósito-Izquierdo et al. (2015) studied the exact method for the restricted container relocation problem. A modified formulation was then proposed to fix the BRP-II model developed by Caserta et al. (2012). In addition, they also proposed a branch and bound based approach, which can reduce the computational time dramatically compared to solving BRP-II model.

Some studies considered the extension of the relocation problems. Lee and Lee (2010) considered a bi-objective container retrieval problem, which minimizes the number of movements and the amount of time required for crane processing. In the retrieval process, the relocation of containers is modelled as multiple stages and solved iteratively. They proposed a three-phase heuristic with two optimization models included in the last two phases based on the optimization software CPLEX. Zhao and Goodchild (2010) discussed the impact of truck arrival time information for reducing container relocation operation in the yard. Simulation results indicated that small improvements in terminal information on truck arrivals can provide significant reductions in container relocations at the yard. Ünluyurt and Aydin (2012) dealt with the relocation problem with two objectives: minimize both the number of relocations and the number of relocations multiplied by the travel distance of the crane. They also presented two heuristics, one based on the method of Kim and Hong (2006) and the other based on the difference in the retrieval priority of containers, respectively.

Rei and Pedroso (2012) considered an extension of the relocation problem in which the arrival time and departure time of containers were known in advance. Three greedy constructive heuristics were also proposed to enhance its practicality. Akyuz and Lee (2014) proposed an integer programming model and a beam search heuristic for the dynamic container relocation problem (DCRP), where various heuristics were used to provide the upper and lower bounds for the beam search. Ji et al. (2015) considered the stowage plan for ships and yards and integrated the loading sequence and the rehandling strategy. Mathematical models and genetic algorithm for three rehandling strategies were developed to solve the problem. Lin et al. (2015) extended the unrestricted problem proposed by Lee and Lee (2010), which takes into account the number of movements as well as the working time of cranes. They proposed a heuristic based on an index regarding both the number of movements and working time. Tang et al. (2015) improved the mathematical model formulated by Wan et al. (2009) and proposed five heuristics to solve it. A discrete-event simulation model was developed to test the performance of the heuristics and analysed the worst case performance.

This research considers the restricted container relocation problem (CRP), involving the selection of appropriate slots for relocated containers during the retrieval process such that the total number of relocations is minimized. A beam search (BS) algorithm is employed to solve this problem. The beam search is similar to a branch-and-bound along with the breadth-first search. However, only a few promising nodes in each level are kept for further search procedures. Two heuristics are also proposed to ensure efficient BS node selection.

## 3. Container relocation problem

The container relocation problem considers a container bay in which $N$ containers are initially allocated and each container has a different retrieval priority that is known in advance. The problem involves minimizing the total number of relocations when retrieving containers from a bay by following the retrieval priorities. A yard crane can relocate a container to a stack in the same bay more quickly than to a stack in a different bay. Yard cranes handle containers in the same bay using a hoist trolley; however, it must travel forward and back to reach containers across different bays. Thus, this study only considers the relocation problem within a single bay.

The terms used hereafter in this paper are defined as follows. The container that is immediately to be retrieved in the bay is called the *target container* and the stack containing the target container is called the *source stack*. The containers above the target container in the source stack are called *blocking containers*, which must be relocated to other stacks to allow access to the target container. The stack to place the relocated container is called the target stack. When the target container is removed, the next container of the highest retrieval priority becomes the new target container. The retrieval procedure is repeated until all containers of interest have been discharged from the bay.

The example in Fig. 2 illustrates the relocation problem with a two-dimensional rendering of a bay in Fig. 2(a), in which the columns and rows represent stacks and tiers, respectively. The squares represent containers and the number inside the squares (containers) indicates its retrieval priority. In this article, the smaller the value of the number allocated to a container is, the higher the retrieval priority will be. Thus, the first container to be retrieved is designated as container 1. Two blocking containers are on top of the target container (containers 4 and 5) in this example. Relocation is required to free the target container. In this case, this involves moving the blocking containers from stack 2 to stacks 3, and container 1 can be retrieved from the bay (Fig. 2(a)-(c)). Fig. 2(d) presents the configuration with a new target container (container 2) after the retrieval of container 1. A similar procedure is applied to the remaining containers until the bay is empty. In the final stage (Fig. 2(f)), no relocations are required after relocating container 5, and the total number of relocations is 4.

In this problem, the following assumptions are made.

1. All containers are of equivalent size.
2. The number of stacks W and tiers H in a bay are given.
3. Relocated containers may only be moved within the same bay.
4. The initial stacking configuration of the bay and the retrieval priority of each container are known in advance.
5. No new containers arrive during the retrieval process.
6. Only blocking containers that are placed atop the target container may be relocated.

Since we focus our research on retrieving export containers, it is reasonable that we will not load new containers into the bay as the fifth assumption stated. The last assumption is reasonable in practical aspect that limits relocation only to the blocking containers at source stack, making this a *restricted* problem (Zehendner and Feillet, 2014). Without assumption A6, this is rendered an *unrestricted* problem oppositely, which allows relocation of the containers at other stacks rather than at source stack during the process of freeing the target container. Since the relocation problem is NP-hard (Caserta et al., 2012), it cannot be solved using an exact solution method, such as branch-and-bound, in reasonable computational time. To overcome this difficulty, this study applies constructive heuristics and a beam search algorithm.

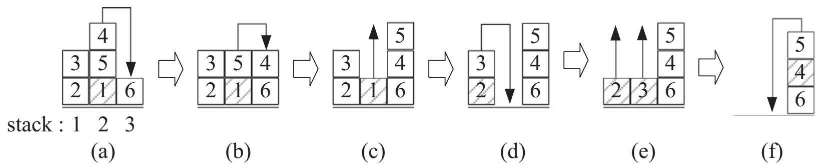

**Fig. 2.** Example of retrieval and relocation process of containers.

## 4. Evaluation heuristics

The heuristic which is embedded in the beam search is a main factor to affect the solution quality. The heuristics are based on the predefined rule to decide the destination for a relocated container. An accurate evaluation of the heuristic can efficiently guide the beam search to obtain good solutions. In this section, two heuristics that are developed for solving container relocation problem are introduced in the following subsections.

### 4.1. Smallest difference heuristic

This article proposes a smallest difference heuristic (SDH), which adopts a simple rule to consider the difference in retrieval priorities between two containers. The same idea was also proposed by Ünluyurt and Aydin (2012) and Caserta et al. (2012). The SDH introduced in this study resembles the heuristic in Caserta et al. (2012). It provides the basis upon which we develop a novel heuristic in next section. We called this heuristic the smallest difference heuristic due to that the destination stack is selected based on the smallest difference in the priorities between the relocated container and the highest priority of the container in the stack. Consider a blocking container with retrieval priority $b$ that must be relocated, the index $DI_s$ for stack $s$ is defined by the difference in retrieval priorities between container $b$ and the highest retrieval priority in stack $s$ (i.e. $f_s$), as shown in Eq. (1). If the stack is empty, $f_s$ is set to $N + 1$.

$$DI_s = f_s - b \tag{1}$$

The destination of a relocated container is determined based on the following two criteria. The first criterion is to check whether there exists a stack with $DI_s > 0$, which indicates no additional relocation is made, because the retrieval priority of the blocking container is higher than all the other containers in stack $s$. The second criterion is to select a stack in which $f_s$ is closest to $b$ (*i.e.* the smallest difference between $f_s$ and $b$) as the target stack. Thus, the strategy of SDH is to find the stacks with positive DIs first. When there are multiple stacks with positive DIs, the stack with the smallest DIs is selected. Conversely, if no such stack exists, the stack with the largest DIs should be chosen.

In SDH, blocking containers with similar priorities have more chance to be placed into the same stack, and containers are implicitly sorted into groups by their retrieval priorities. This process avoids the containers of the low retrieval priorities being relocated frequently, and thus the overall number of relocations can be reduced. Another advantage of SDH is that the tie-breaking rule is not necessary. The rule avoids the case of ties in index (excepting empty stacks), because the difference in retrieval priorities between the relocated container and arbitrary containers is unique.

The example in Fig. 3 illustrates the SDH process. Fig. 3(a) presents the initial container bay of three stacks and three tiers. According to the rule, $DI_1$ and $DI_3$ are −2 and 2 for stacks 1 and 3, respectively. Blocking container 4 is moved to stack 3 because it has the lowest positive $DI_s$. The stacks in Fig. 3(b) do not have any positive $DI_s$; therefore, stack 3 with the highest negative index is selected. The process is repeated until every container in the bay has been removed, resulting in a total of four relocations.

### 4.2. Virtual relocation heuristic

Most heuristics for the relocation problem aim to find a criterion which determines a target stack for a relocating container. These heuristics greedily seek a target stack only for the topmost blocking container but ignore the influence of these actions on other blocking containers. To further improve the solution quality, the mutual effect of multiple blocking containers should be considered simultaneously. In this study, we develop a heuristic to deal with a set of blocking containers by changing their *decision sequence*, which is the order that the blocking containers determine their target stacks. Though the blocking containers have to be relocated from top to bottom, the decision sequence is not necessary to follow the same order. A similar idea called chain heuristic was introduced by Jovanovic and Voss (2014), in which only two consecutive containers were considered at a time. Their method evaluates target stacks for both the highest two containers at a step, to prevent a stack that is more suitable for the lower container from being taken by the upper one. However, it is difficult to extend their method to more than two containers, because the number of the possible evaluations may increase exponentially as the number of blocking containers grows.

In this article, a virtual relocation heuristic (VRH) is proposed, in which the strategy changes the decision sequence and all blocking containers are taken into account simultaneously. Rather than evaluating all possible decision sequences, as

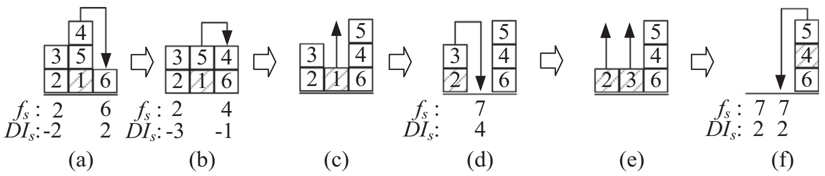

**Fig. 3.** Example of the smallest difference heuristic (SDH).

mentioned in Jovanovic and Voss (2014), the sequence in the VRH is given according to the retrieval priorities of blocking containers. In this manner, the VRH does not need to suffer from large computational cost. Obviously, the relocation of a lower container may be handled earlier than upper containers, and thus it is quite possible to violate the Last-In-First-Out (LIFO) policy. To keep the containers following their initial order as in the source stack, a blocking container is either placed at a stack that does not violate the LIFO policy, or moved as insertion into a mid-slot below other relocated containers. An example shown in Fig. 4 illustrates this situation, in which containers {5,8,6,7} are recognized as the blocking containers. Let the decision sequence of the blocking containers be {8,7,6,5}. As can be seen from Fig. 4(c), relocating container 6 above container 7 violates the LIFO policy, because container 7 is at lower tier than container 6 in the source stack. Hence, to avoid violating the LIFO policy, container 6 can be either placed at the top of stack 2, or inserted into mid-slot between container 7 and 8 in stack 3. The move during the process of relocating blocking containers is called virtual relocation, because the actual target position is not confirmed until all blocking containers determine their target stacks.

The pseudo code of VRH is rendered as follows. The outer while loop (line 2) keep retrieving the target container iteratively until the container bay is empty. When the target container cannot be accessed, VRH is then required to free the target container by relocating all blocking containers. Two phases including essential features to handle the blocking containers are described in the following paragraphs.

```
Input: Bay configuration, number of containers $N$
Output: the solution consisting of relocation operations $(b, x_b)$
        total relocation number $R$
1:   target container $t = 1$ and $R = 0$
2:   while $t \le N$ do
3:     if Container $t$ is the topmost container
4:       Retrieve(container $t$)
5:     else
6:       $Q \leftarrow$ Obtain the set of blocking containers of container $t$
7:       Obtain $f_s$ and let $T_s = H$ for each stack $s$
         [Phase I]
8:       $Q' = Q$
9:       Sort($Q'$) by priority in descending order
10:      for each container $b$ in $Q'$
11:        Obtain $DI_s$ for each stack $s$.
11:        $s^* \leftarrow \min_s\{DI_s: DI_s > 0$ and $t_b < T_s, \forall s\}$
12:        if $s^*$ exist
13:          $x_b = s^*$
14:          update $f_{s*}$ and $T_{s*}$
15:          $Q' \leftarrow Q'/\{b\}$
16:        end if
17:      end for
         [Phase II]
18:      for each container $b$ in $Q'$ in reverse order
19:          obtain $VRI_s$ for each stack $s$.
20:          $x_b \leftarrow \max_s \{VRI_s, \forall s\}$
21:          $Q' \leftarrow Q'/\{b\}$
22:        end for
23:        Relocate container $b$ to stack $x_b$ for each $b \in Q$
24:        $R = R + |Q|$
25:      end if
26:      $t = t + 1$
27:  end while
```

In Phase I, the decision sequence is given in order according to the priority from lowest to highest (line 9), because the containers with lower retrieval priority may stay longer in the bay. With such decision sequence, the blocking container with lower priority has more chance to select a suitable stack and avoid additional relocations. Additionally, this phase only accepts the movement that will not result in any additional relocation. If more than one stack satisfies the condition, then the stack that has the smallest *difference index* ($DI_s$) will be selected as the target stack (line 11). When no such stack exists, the movement of the blocking container is skipped in the first phase, and this kind of container is called *skipping container*. It is worth noting that additional relocations for the skipping containers are inevitable, and the second phase therefore attempts to reduce the future relocations as many as possible.

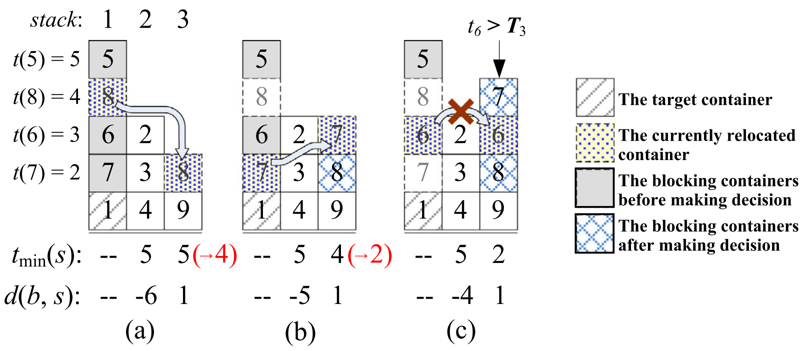
stack: 1 2 3
t(5) = 5
t(8) = 4
t(6) = 3
t(7) = 2
t6 > T3
tmin(s): -- 5 5(→4) -- 5 4(→2) -- 5 2
d(b, s): -- -6 1 -- -5 1 -- -4 1
(a)
(b)
(c)
The target container
The currently relocated container
The blocking containers before making decision
The blocking containers after making decision

**Fig. 4.** Illustration of the handling procedure in Phase I: (a) An initial layout and the first virtual relocation of container 8 due to its lowest priority; (b) handling container 7 due to its second lowest priority; (c) an additional relocation caused by placing container 6 at the same stack as container 7.

For quickly recognizing whether a virtual relocation will generate any additional relocation, two conditions have to be verified in Phase I. Let $b$ represent the blocking container that needs to determine its target stack, $f_s$ be the highest retrieval priority in stack $s$. Then, we shall define $t_b$ as the tier where container $b$ initially located in the source stack, and $T_s$ as the minimum $t_x$ among all blocking containers $x$ that have been virtually relocated to stack $s$. The following two conditions lead to no additional relocation.

Condition 1: $b < f_s$ (or $DI_s > 0$)
Condition 2: $t_b < T_s$

The first condition shows that container $b$ can be relocated to stack $s$ only if the priority of the container is higher than the highest priority $f_s$ in the stack. The second condition avoids the case that the blocking container is moved into mid-slot between other blocking containers. Because in phase I the containers that have been virtually relocated imply that their priorities are lower than the current container $b$, placing container $b$ below any virtual relocated container will result in additional relocation. As the example shown in Fig. 4(c), when container 6 is relocated to the top of stack 2, additional relocation will be occurred because container 2 will be retrieved earlier than container 6. On the other hand, $T_3$ is equal to 2 by taking the minimum value from $\{t_7, t_8\}$ (=$\{2,4\}$) in stack 3. When container 6 is moved to stack 3, additional relocation occurred due to $t_6$ (=3) > $T_3$ (=2).

In phase II, the skipping containers are handled according to their retrieval priority from highest to lowest. When a skipping container is moved to a target stack containing relocated blocking containers, the skipping container may be moved into a mid-slot below other blocking containers. The relocated containers in the stack are then divided by the skipping container into upper containers and lower containers. The upper containers would be considered as additional relocations, if the skipping container has higher priority than the upper ones. Then, the target stack is determined based on *virtual relocation index* (*VRI*), which considers the additional relocations of upper containers, and the highest priorities of upper and lower containers. More precisely, let $u_s$ and $l_s$ represent the highest priorities among upper containers and among lower containers in stack $s$, respectively. $U$ is defined as the number of additional relocations of upper containers. The $VRI_s$ for stack $s$ is computed as Eq. (2). When no more than one additional relocation exists, the differences between skipping container $b$ and both upper and lower containers are considered. In case of more than one of such additional relocation, a penalty of $-N$ is added to reduce the weight of the candidate stack. The stack with largest $VRI_s$ will be selected as the target stack of skipping container $b$.

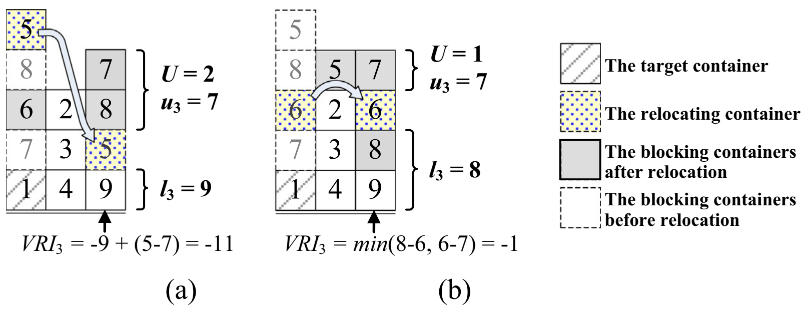

**Fig. 5.** Illustration of the handling procedure in Phase II: (a) Virtual relocation for container 5 causing two additional relocation; (b) an additional relocation caused by placing container 6 at the same stack as container 7.

$$VRI_s = \begin{cases} \min\{l_s - b, b - u_s\} & \text{if } U \leqslant 1 \\ -N + (b - u_s) & \text{otherwise} \end{cases} \tag{2}$$

Fig. 5 illustrates the calculation of *VRI*, in which nine containers are stacked in a bay with three stacks and at most five tiers. After the first phase of VRH, container 7 and 8 can be virtually relocated to stack 3 without any additional relocation. As the example shown in Fig. 5(a), the case involves more than one additional relocation. Fig. 5(b) illustrates the *VRI* of stack 3 involving the upper container and lower container divided by container 6. The *VRI* is obtained by the minimal value of the two difference indexes from upper and lower containers.

## 5. Beam search algorithm

Beam Search (BS) is a heuristic adapted from the branch-and-bound algorithm. It was first applied in the field of artificial intelligence to deal with problems such as speech recognition and image understanding (Lowerre, 1976; Rubin, 1978). A number of researchers have since applied BS to scheduling problems (Fox, 1983; Ow and Morton, 1988; Ow and Smith, 1988). An extension of BS, called filtered beam search, was proposed by Ow and Morton (1988) to achieve a compromise between solution quality and computational time. Other researchers have recently improved the conventional BS through the incorporation of new mechanisms. Valente and Alves (2005) developed a filtered beam search for the early/tardy scheduling problem, in which a recovering step introduces an interchange operator to modify a partial sequence of the current solution. Sabuncuoglu et al. (2008) employed an enhanced BS for assembly line scheduling problems, which allows backtracking to a previous level and exchanging of partial solutions by different beams.

Beam search is a heuristic based on the breadth-first tree search, in which nodes are generated and visited level by level. Unlike a branch-and-bound, which visits every node in the unbounded feasible solution to locate an optimal solution, a beam search maintains only $\beta$ promising nodes at each level, where $\beta$ is the so-called beam width. A promising node is known as a *beam node*, and is usually identified by a lower or upper bound obtained from a cost evaluation function. Only the beam node can create descendant nodes at each level and the other nodes are discarded permanently, which precludes the option of backtracking. BS can reduce a search space considerably by visiting only a limited number of nodes. However, BS is simply a constructive based heuristic, the optimality of which cannot be guaranteed due to the likelihood that the optimal solution may be discarded during the search process.

An example of a beam search with $\beta$ equal to two is presented in Fig. 6, where the relocation of a container from the top of stack $i$ to stack $j$ is denoted as $R(i, j)$. The root carries an initial configuration comprising six containers. Because the target container is blocked from retrieval by container 4, nodes $N_1$ and $N_2$ are generated by relocating container 4 from stack 2 to stack 1 or stack 3, respectively. Because $\beta = 2$, both $N_1$ and $N_2$ must be selected as beam nodes in the first level. Then, the upper bounds for nodes $N_3$, $N_4$, and $N_5$ in the third level are evaluated by an arbitrary heuristic. Two of these are selected as beam nodes according to their evaluation cost. In this example, $N_3$ and $N_5$ have better evaluation values than $N_4$. The procedure is repeated until the node that involves an empty bay is acquired.

There are generally two kinds of BS, in terms of selection criteria for the beam nodes. The *independent* BS selects one node from the descendants of each beam node, while the *dependent* BS selects $\beta$ nodes from among all of the descendants of beam nodes (Sabuncuoglu and Bayiz, 1999). The difference between the two procedures is illustrated by Fig. 6. When independent BS is applied, only descendent from $N_1$, node $N_3$ must be one of the beam nodes. The choices for beam nodes can be either ($N_3$, $N_4$) or ($N_3$, $N_5$), when $\beta = 2$. However, in dependent BS, any node with a better evaluation can be regarded as a beam node. Thus, the combination ($N_4$, $N_5$) could also be an option for beam nodes in this example. Ideally, the independent BS travels over a broader range of solution spaces to improve diversity in choices, while the dependent BS may search fewer beams to

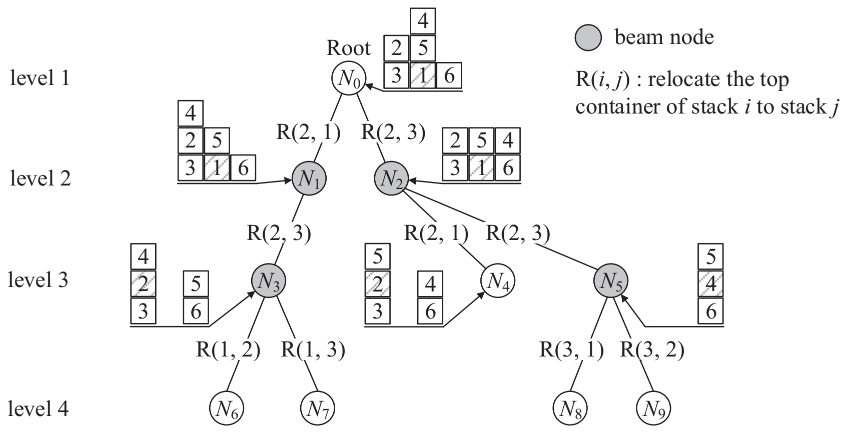

**Fig. 6.** Representation of beam search for the container relocation problem.

obtain a more focused solution. In this study, both strategies are applied to select beam nodes. In order to achieve good solution more quickly, the dependent strategy is the primary criterion for the selection.

In each level, the descendants are first sorted according to their evaluation cost. If more than $\beta$ descendants have evaluation costs less than or equal to that of the $\beta$th node, then the nodes with the evaluation cost less than that of the $\beta$th node is selected as beam nodes. For those descendants have the same evaluation costs of the $\beta$th node, the independent strategy is applied to break ties. The unfilled beam nodes are only picked from descendants with evaluation costs equal to the $\beta$th node. Moreover, selection process chooses the beam nodes according to the parents of those descendants. In order to enhance the diversity of the search space, the heuristic selects descendants from different as many parents as possible. Thus, during the selection process, the descendant of different parents from the selected beam nodes is selected first.

The pseudo code of the proposed beam search is shown as follows.

```
Beam Search
Notations:
$\beta$:      beam width
$k$:          the level of the search tree
$B$:          the current configuration of the container bay
$B_k$:        set of beam nodes at level $k$
$C_k$:        set of children nodes at level $k$
$C$:          the children node generated from $B_k$.
$E_\beta$:    the evaluation value of the $\beta$th node of the sorted $C_k$
1:    $k = 0$
2:    do retrieve the target container from $B_0$
3:    until (the target container is inaccessible)
4:    do
5:      $k = k + 1$
6:      $\mathbf{C}_k = \emptyset$
        [Branching]
7:      for each $B \in \mathbf{B}_k$:
8:        for each feasible selection of target stack $j$
9:          $C$ is obtained by relocating the top blocking container to stack $j$
10:         Use heuristic to evaluate the relocation# for $C$
11:         $\mathbf{C}_k = \mathbf{C}_k \bigcup C$
12:       end for
13:     end for
14:     Sort $\mathbf{C}_k$ in ascending order according to the value of the evaluation.
        [Selection]
15:     if $|\mathbf{C}_k| < \beta$ then
16:       $\mathbf{B}_{k+1} = \mathbf{C}_k$
17:     else
18:       $\mathbf{C}_k^{\mathbf{L}}$ = nodes of $\mathbf{C}_k$ involve the evaluation values less than $E_\beta$
19:       $\mathbf{C}_k^{\mathbf{R}} = (\beta - |\mathbf{C}_k^{\mathbf{L}}|)$ nodes randomly selected from the nodes of $\mathbf{C}_k$ with the evaluation values equal to $E_\beta$
20:       $\mathbf{B}_{k+1} = \mathbf{C}_k^{\mathbf{L}} \bigcup \mathbf{C}_k^{\mathbf{R}}$
21:     end if
        [Retrieval]
22:     for each $B \in \mathbf{B}_{k+1}$:
23:       do retrieve the target container from $B$
24:       until (the target container is inaccessible)
25:     end for
26:   until (there exists $B \in \mathbf{B}_{k+1}$ such that $B$ is empty)
27:   return $k$
```

Firstly, all parameters and the configuration of the bay are initialized. Before the root (beam node) sprouts descendants, line 2 and lines 21–24 perform a retrieval process until a target container is inaccessible, whereupon the node is ready to perform relocation. Lines 3 and 25 show that the beam search is terminated only if there is at least one beam node with an empty container bay inside. Level $k$ is increased by 1 in line 4. A dependent search is applied in lines 6–13, in which all of the children nodes in $\mathbf{C}_k$ generated by the set of beam nodes $\mathbf{B}_k$ are collected. These nodes are then sorted in ascending order according to the evaluated relocation number in line 13. Next, $\beta$ nodes are selected from the children nodes in lines 14–20. If multiple nodes have the evaluation value smaller than or equal to that of the $\beta$th node, then nodes with a smaller value will be retained as beam nodes for the next level. The remaining number of beam nodes is randomly selected from the nodes with an evaluation value equivalent to that of the $\beta$th node. In lines 21–24, a retrieval procedure removes all accessible target containers in every beam node. Finally, if a beam node of the empty container bay exists, then the beam search is terminated and level $k$, indicating the minimum number of relocations, is returned.

## 6. Computational experiment and analysis

The beam search and proposed heuristics were coded in C++ and run on a PC equipped with an Intel i5-2500 k 3.3 GHz and 4.0 GB RAM under the Windows XP operating system. To compare the performance of the proposed methods with a competitive algorithm from the literature, three sets of benchmark instances from Wu and Ting (2010), Caserta et al. (2011b), Lee and Lee (2010) were tested in this section. In the following experiments, the performance of the proposed two heuristics in Section 4 was tested first. The better heuristic will be the evaluation heuristic that embedded into the BS for further testing on the effectiveness of BS. Because only one parameter, beam width $\beta$, determines the quality of the

**Table 1**
Computational results of SDH and VRH using the instances from Wu and Ting (2010).

| *W* | *H* | RI | Min-Max | Chain | Chain-F | SDH | VRH |
|---|---|---|---|---|---|---|---|
| 3 | 3 | 3.42 | 3.42 | 3.42 | 3.38 | 3.42 | **3.37** |
| 3 | 4 | 6.10 | 5.82 | 5.82 | 5.95 | 5.82 | **5.70** |
| 3 | 5 | 9.80 | 9.10 | 9.10 | 8.70 | 9.10 | **8.50** |
| 3 | 6 | 13.60 | 12.97 | 12.77 | 12.30 | 12.97 | **12.20** |
| 3 | 7 | 18.10 | 16.62 | **16.25** | 16.40 | 16.62 | 16.27 |
| 3 | 8 | 24.10 | 22.02 | 21.52 | 21.20 | 22.02 | **20.57** |
| 4 | 3 | 5.03 | 4.95 | 4.95 | 4.95 | 4.95 | **4.87** |
| 4 | 4 | 9.05 | 8.75 | 8.72 | 8.57 | 8.75 | **8.52** |
| 4 | 5 | 14.50 | 13.12 | 13.02 | 13.17 | 13.12 | **12.60** |
| 4 | 6 | 19.10 | 17.15 | 17.05 | 16.92 | 17.15 | **16.52** |
| 4 | 7 | 28.90 | 26.37 | 25.97 | 25.62 | 26.37 | **25.17** |
| 4 | 8 | 37.90 | 33.67 | 33.05 | 32.82 | 33.67 | **30.95** |
| 5 | 3 | 5.90 | **5.75** | **5.75** | 5.80 | **5.75** | **5.75** |
| 5 | 4 | 12.20 | 11.40 | 11.35 | 11.40 | 11.40 | **11.30** |
| 5 | 5 | 18.10 | 17.07 | 16.95 | 16.65 | 17.07 | **16.20** |
| 5 | 6 | 25.60 | 23.92 | 23.52 | 23.57 | 23.92 | **22.55** |
| 5 | 7 | 36.30 | 31.92 | 31.15 | 31.42 | 31.92 | **30.42** |
| 5 | 8 | 49.80 | 43.07 | 42.57 | 42.00 | 43.07 | **39.52** |
| 6 | 3 | 7.92 | 7.72 | 7.72 | 7.85 | 7.72 | **7.70** |
| 6 | 4 | 13.20 | 12.67 | 12.55 | 12.60 | 12.67 | **12.50** |
| 6 | 5 | 22.60 | 20.60 | 20.42 | 20.52 | 20.60 | **20.22** |
| 6 | 6 | 32.60 | 29.02 | 28.77 | 28.57 | 29.02 | **28.22** |
| 6 | 7 | 45.50 | 40.42 | 39.22 | 38.65 | 40.42 | **38.20** |
| 6 | 8 | 57.20 | 52.20 | 50.87 | 50.25 | 52.20 | **48.67** |
| 7 | 3 | 10.10 | **9.02** | 9.05 | 9.17 | **9.02** | **9.02** |
| 7 | 4 | 20.10 | 16.35 | 16.45 | 16.07 | 16.35 | **15.97** |
| 7 | 5 | 30.90 | 23.15 | 22.92 | 22.70 | 23.15 | **22.60** |
| 7 | 6 | 45.00 | 34.40 | 34.07 | 34.07 | 34.40 | **33.15** |
| 7 | 7 | 59.80 | 45.62 | 44.92 | 44.27 | 45.62 | **43.35** |
| 7 | 8 | 76.60 | 59.02 | 57.97 | 56.32 | 59.02 | **54.40** |
| 8 | 3 | 11.90 | 9.87 | 9.87 | 9.75 | 9.87 | **9.72** |
| 8 | 4 | 20.90 | 18.72 | 18.52 | 18.52 | 18.72 | **18.30** |
| 8 | 5 | 34.70 | 27.67 | 27.37 | 26.82 | 27.67 | **26.67** |
| 8 | 6 | 50.60 | 40.47 | 39.65 | **38.90** | 40.47 | 38.97 |
| 8 | 7 | 68.40 | 52.10 | 51.15 | 50.82 | 52.10 | **49.00** |
| 8 | 8 | 88.90 | 66.37 | 65.12 | 64.55 | 66.37 | **62.85** |
| 9 | 3 | 12.40 | 11.62 | 11.60 | 11.57 | 11.62 | **11.55** |
| 9 | 4 | 24.70 | 19.85 | 19.72 | 19.87 | 19.85 | **19.60** |
| 9 | 5 | 37.50 | 31.00 | 30.35 | 30.55 | 31.00 | **29.70** |
| 9 | 6 | 55.70 | 44.90 | 44.50 | 43.30 | 44.90 | **42.57** |
| 9 | 7 | 76.70 | 59.87 | 57.92 | 57.35 | 59.87 | **56.00** |
| 9 | 8 | 97.90 | 76.32 | 75.05 | 73.50 | 76.32 | **71.92** |
| 10 | 3 | 14.60 | 12.02 | 12.02 | 11.97 | 12.02 | **11.95** |
| 10 | 4 | 26.20 | 23.60 | 23.32 | 23.25 | 23.60 | **23.20** |
| 10 | 5 | 41.20 | 34.50 | 33.80 | 33.45 | 34.50 | **33.35** |
| 10 | 6 | 60.20 | 48.80 | 47.77 | 47.45 | 48.80 | **46.95** |
| 10 | 7 | 85.10 | 65.72 | 64.37 | 63.60 | 65.72 | **62.55** |
| 10 | 8 | 111.00 | 83.90 | 82.02 | 79.35 | 83.90 | **78.30** |
| Average | | 34.95 | 28.85 | 28.37 | 28.05 | 28.85 | **27.46** |

solution and computational time of BS, the experiments of BS were conducted using four beam widths: $\beta = 5$, 10, 15, and 20. Each instance was tested with 10 runs.

## 6.1. Computational results of heuristics

The first experiment is carried out using the benchmark instances from Wu and Ting (2010) to compare SDH and VRH heuristics with the Min-Max heuristic by Caserta et al. (2012) and chain heuristic by Jovanovic and Voss (2014). As shown in Table 1, the benchmark instance includes 48 sets involving the range of the number of stacks ($W$) and the maximum number of tiers ($H$) from 3–10 and 3–8, respectively. The number of containers ($N$) of each set was defined by $N = \lceil((W-1)H+1)\rceil$, where symbol $\lceil x \rceil$ represents the smallest integer greater than or equal to $x$. Each set contains 40 instances, and the value shown in the cell is the average number of relocations obtained by each heuristic.

In Table 1, the first 2 columns show the number of stacks and tiers. Column 3–6 are the results of comparing heuristics, while the last two columns are heuristics proposed by the current study. The RI heuristic is from Murty et al. (2005), the Min-Max heuristic is proposed by Caserta et al. (2012), and chain heuristics (chain and chain-F) are from Jovanovic and Voss (2014). Values of the RI heuristic are taken from Wu and Ting (2010), while values of Min-Max, chain heuristics (chain and chain-F) are taken from Jovanovic and Voss (2014). Because such heuristics can find a feasible solution to the test problem with negligible computational time (less than one second for all instances), elapsed time is not provided by Jovanovic and Voss (2014) and neither by our methods. The results show that SDH outperforms RI in all 48 instance sets, but the results of SDH are the same as those by Min-Max. Note that SDH has the similar solving procedure as Min-Max and the chain heuristic algorithms can be considered as an extension of the Min-Max heuristic, therefore it is reasonable that either Chain or Chain-F outperforms SDH in most instances. On the other hand, VRH considered a whole set of blocking containers; therefore, it outperforms the other five heuristics in terms of the number of relocations. The VRH has improved the chain heuristics' results in almost all cases. More precisely, VRH obtained the best results in 46 over 48 sets. In the next computational experiments, VRH will be embedded into BS as the evaluation function, due to its better performance.

## 6.2. Computational results of the proposed beam search

Three instance sets were tested in this section to verify the performance of our BS. The first comparison was run using the instances provided by Wan et al. (2009). Their data sets include three types of instances in terms of the density at initial bay. Because the instance of the low density can be easily solved to optimality by the heuristics introduced in Section 6.1, we only took the instance set "Heavy" that has the highest density of 80% among three types of instances for comparison. In the "Heavy" data set, the number of containers ($N$) is defined as $N = \lceil 0.8 * ((W-1) * H + 1)\rceil$. To show the effectiveness of the proposed BS, the experiment was conducted by comparing to the optimal solutions of MRIP which is an integer programming model formulated by Wan et al. (2009). The first four columns of Table 2 indicate the number of stacks, the number of tiers, the optimal relocations, and the time consumed by MRIP, respectively. The machine used for MRIP was a PC with an Intel dual core Xeon 3 GHz and 4 GB of RAM. The results in the last eight columns were obtained by BS, representing the average number of relocations and run time under various beam widths. According to these results, MRIP took more computational times to achieve the optimal solution. BS required only a small beam width ($\beta = 10$) to obtain optimal solutions in a very short time (less than 0.002 s on average). Even when the beam width equals to 5, BS can obtain near-optimal solutions with 0.15% more on average than optimal relocations.

The second data set was obtained from Lee and Lee (2010), comprising two sets of instances. The first set randomly locates containers in the container bays, while the second set considers a more difficult case in which containers of high retrieval priority are always located below the containers with lower retrieval priority. The test data of Lee and Lee (2010) considered the problem of multiple container bays. To ensure a fair comparison, only the instances with single bay were used in this paper. Each bay includes 16 stacks, 6 or 8 tiers, and 70 or 90 containers. Computational results are outlined in Table 3. Columns 1–5 represent the instance ID, the number of stacks, the maximum number of tiers, the number of containers, and the lower bound from Lee and Lee (2010). The authors proposed a three-phase heuristics, denoted by LL, the second phase of which is to reduce relocation by an integer programming model and solved using optimization package

**Table 2**
Computational results of BS using the instances from Wan et al. (2009).

| Instances (Heavy) | | MRIP | | BS | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $\beta = 5$ | | $\beta = 10$ | | $\beta = 15$ | | $\beta = 20$ | |
| $W$ | $H$ | R# | Time (s) | R# | Time (s) | R# | Time (s) | R# | Time (s) | R# | Time (s) |
| 6 | 2 | **1.70** | 0.066 | **1.70** | 0.000 | **1.70** | 0.000 | **1.70** | 0.000 | **1.70** | 0.000 |
| 6 | 3 | **4.58** | 0.432 | **4.58** | 0.000 | **4.58** | 0.000 | **4.58** | 0.001 | **4.58** | 0.001 |
| 6 | 4 | **7.56** | 85.974 | **7.56** | 0.000 | **7.56** | 0.001 | **7.56** | 0.001 | **7.56** | 0.002 |
| 6 | 5 | **11.68** | 689.421 | 11.7 | 0.001 | **11.68** | 0.002 | **11.68** | 0.003 | **11.68** | 0.003 |
| Avg. | | **6.38** | 193.973 | 6.39 | 0.000 | **6.38** | 0.001 | **6.38** | 0.001 | **6.38** | 0.002 |

**Table 3**
Computational results of BS using the instances from Lee and Lee (2010).

| ID | W | H | N | LB | LL | | BS | |
|---|---|---|---|---|---|---|---|---|
| | | | | | R# | Time (s) | R# | Time (s) |
| R011606_0070_001 | 16 | 6 | 70 | 30 | 48 | 6304.28 | **37** | 0.016 |
| R011606_0070_002 | 16 | 6 | 70 | 34 | 47 | 11081.03 | **38** | 0.016 |
| R011606_0070_003 | 16 | 6 | 70 | 34 | 40 | 5501.92 | **38** | 0.015 |
| R011606_0070_004 | 16 | 6 | 70 | 38 | 88 | 9026.42 | **45** | 0.032 |
| R011606_0070_005 | 16 | 6 | 70 | 36 | 54 | 9107.97 | **40** | 0.015 |
| R011608_0090_001 | 16 | 8 | 90 | 53 | 100 | 13268.67 | **61** | 0.047 |
| R011608_0090_002 | 16 | 8 | 90 | 49 | 101 | 11134.63 | **61** | 0.031 |
| R011608_0090_003 | 16 | 8 | 90 | 52 | 126 | 21583.13 | **65** | 0.047 |
| R011608_0090_004 | 16 | 8 | 90 | 53 | 88 | 7042.38 | **61** | 0.047 |
| R011608_0090_005 | 16 | 8 | 90 | 53 | 92 | 13738 | **60** | 0.031 |
| U011606_0070_001 | 16 | 6 | 70 | 55 | 55 | 17326.31 | **55** | 0.016 |
| U011606_0070_002 | 16 | 6 | 70 | 54 | 60 | 11243.4 | **58** | 0.031 |
| U011608_0090_001 | 16 | 8 | 90 | 74 | 85 | 21586.81 | **76** | 0.047 |
| U011608_0090_002 | 16 | 8 | 90 | 74 | 90 | 8021.31 | **78** | 0.047 |
| Average | | | | 49.21 | 76.7 | 11854.73 | **55.2** | 0.031 |

CPLEX. The results and run time are reported in columns 6 and 7. The last two columns present the results of the proposed BS; beam width is not shown because the same outcome is obtained in all $\beta > 5$. Since the algorithm of Lee and Lee (2010) was solved using CPLEX, the algorithm took considerably longer. Clearly, BS outperforms LL in terms of computational time and solution quality.

The last data set is obtained from Caserta et al. (2011b). The data set includes 21 sets with problems of various sizes, and each set includes 40 randomly generated instances. Problem size depends on the number of stacks $W$ and the number of tiers $T$. The $W * T$ slots in each instance are filled with containers. In this experiment, both $W$ and $T$ range from 3 to 10. Hence, two more tiers are reserved for relocated containers and the limitation in the number of tiers is given by $H = T + 2$.

Table 4 compares the results obtained using BS with those of ENAR by Kim and Hong (2006) and the Corridor Method (CM) proposed by Caserta et al. (2011b). CM is a heuristic method based on a dynamic programming, in which the solution space can be narrowed by searching through only a limited range of stacks and tiers. The computer used to run CM was Pentium IV Linux Workstation with 512 Mb of RAM. Columns 1–4 in Table 4 outline the information related to the instances. Columns 5–7 present the average number of relocations obtained from ENAR and CM as well as the computational time

**Table 4**
Computational results of BS using the instances from Caserta et al. (2011b).

| | | | | | | | BS | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | ENAR | CM | | $\beta = 5$ | | $\beta = 10$ | | $\beta = 15$ | | $\beta = 20$ | |
| No | $W$ | $H$ | $T$ | | R# | Time (s) | R# | Time (s) | R# | Time (s) | R# | Time (s) | R# | Time (s) |
| 1 | 3 | 5 | 3 | 7.10 | 5.40 | 0.10 | **5.00** | 0.000 | **5.00** | 0.000 | **5.00** | 0.000 | **5.00** | 0.000 |
| 2 | 4 | 5 | 3 | 10.70 | 6.50 | 0.10 | **6.17** | 0.000 | **6.17** | 0.000 | **6.17** | 0.001 | **6.17** | 0.001 |
| 3 | 5 | 5 | 3 | 14.50 | 7.30 | 0.10 | **7.02** | 0.000 | **7.02** | 0.000 | **7.02** | 0.000 | **7.02** | 0.001 |
| 4 | 6 | 5 | 3 | 18.10 | **7.90** | 0.15 | 8.40 | 0.001 | 8.40 | 0.001 | 8.40 | 0.001 | 8.40 | 0.002 |
| 5 | 7 | 5 | 3 | 20.10 | **8.60** | 0.10 | 9.27 | 0.000 | 9.27 | 0.001 | 9.27 | 0.002 | 9.27 | 0.002 |
| 6 | 8 | 5 | 3 | 26.00 | **10.50** | 0.20 | 10.65 | 0.001 | 10.65 | 0.002 | 10.65 | 0.002 | 10.65 | 0.003 |
| 7 | 4 | 6 | 4 | 16.00 | **9.90** | 0.20 | 10.25 | 0.000 | 10.22 | 0.001 | 10.20 | 0.001 | 10.20 | 0.001 |
| 8 | 5 | 6 | 4 | 23.40 | 16.50 | 0.50 | **12.95** | 0.000 | **12.95** | 0.001 | **12.95** | 0.002 | **12.95** | 0.002 |
| 9 | 6 | 6 | 4 | 26.20 | 19.80 | 0.50 | **14.02** | 0.001 | **14.02** | 0.002 | **14.02** | 0.002 | **14.02** | 0.002 |
| 10 | 7 | 6 | 4 | 32.20 | 21.50 | 0.50 | **16.12** | 0.002 | **16.12** | 0.002 | 16.15 | 0.004 | **16.12** | 0.004 |
| 11 | 4 | 7 | 5 | 23.70 | 16.60 | 0.50 | **15.45** | 0.001 | **15.45** | 0.001 | **15.45** | 0.002 | **15.45** | 0.002 |
| 12 | 5 | 7 | 5 | 37.50 | **18.80** | 0.80 | 19.00 | 0.001 | 18.90 | 0.002 | 18.90 | 0.003 | 18.90 | 0.005 |
| 13 | 6 | 7 | 5 | 45.50 | **22.10** | 0.80 | 22.22 | 0.002 | 22.22 | 0.004 | 22.20 | 0.004 | 22.17 | 0.005 |
| 14 | 7 | 7 | 5 | 52.30 | 25.80 | 1.43 | 24.37 | 0.002 | 24.32 | 0.005 | **24.30** | 0.007 | **24.30** | 0.009 |
| 15 | 8 | 7 | 5 | 61.80 | 30.10 | 1.46 | 27.80 | 0.003 | 27.80 | 0.006 | **27.77** | 0.010 | **27.77** | 0.013 |
| 16 | 9 | 7 | 5 | 72.40 | 33.10 | 1.41 | 30.52 | 0.005 | 30.55 | 0.009 | **30.47** | 0.014 | **30.47** | 0.019 |
| 17 | 10 | 7 | 5 | 80.90 | 36.40 | 1.87 | 33.40 | 0.007 | 33.32 | 0.014 | **33.32** | 0.019 | **33.32** | 0.026 |
| 18 | 6 | 8 | 6 | 37.30 | 32.40 | 1.74 | 31.20 | 0.004 | 31.10 | 0.006 | **31.07** | 0.009 | **31.07** | 0.011 |
| 19 | 10 | 8 | 6 | 75.10 | 49.50 | 1.95 | 46.15 | 0.012 | 46.10 | 0.024 | 46.00 | 0.034 | **45.95** | 0.045 |
| 20 | 6 | 12 | 10 | 141.60 | 102.00 | 4.73 | 79.97 | 0.013 | 79.30 | 0.025 | 78.95 | 0.039 | **78.80** | 0.051 |
| 21 | 10 | 12 | 10 | 178.60 | 128.30 | 6.34 | 113.25 | 0.062 | 112.47 | 0.120 | 112.32 | 0.178 | **111.97** | 0.237 |
| Average | | | | 47.667 | 29.000 | 1.213 | 25.866 | 0.006 | 25.779 | 0.011 | 25.742 | 0.016 | **25.713** | 0.021 |

of CM from Caserta et al. (2011b). The next eight columns present the average number of relocations and average computational times for different beam widths.

The results in Table 4 show that the overall average number of relocations of the proposed method is lower than that of ENAR and CM, even when $\beta$ is equal to 5. The difference between the average number of relocations using different beam widths is only 0.152. This indicates that beam search can provide quite robust results. With the exception of six instance sets, BS outperformed CM in each instance for a range of beam widths. However, on the basis of personal communication with the author of Caserta et al. (2011b), CM cannot return solutions for some instances, and the values of CM in Table 4 may not be the average results over 40 instances. Although the comparison may be misleading, the computational results still illustrate the performance of BS on these data sets. In the instances of the largest size, BS with $\beta = 20$ reduces the average number of relocations (from 128.30 down to 111.97) by approximately 12.72% of CM. It should be noted that, ideally, increasing beam width should reduce the number of relocations. However, this is not necessarily true for all individual instances, such as the 10th instance set in which $\beta = 15$ does not provide better results than $\beta = 10$. This could be the fact that the selection of a beam node is based on estimated cost and even a beam node with smaller estimation cannot be guaranteed to obtain a better final solution. Thus, increasing the collection of beam nodes could direct the search to an entirely different region that enables better estimation but may eventually produce a worse solution.

**Table 5**
Comparison results among A$^*$ algorithm by Expósito-Izquierdo et al. (2014), B&B by Expósito-Izquierdo et al. (2015), and the beam search ($\beta = 5$).

| *W* | *H* | *T* | Inst | opt | A$^*$ | B&B | BS ($\beta = 5$) | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Time (s) | Time (s) | R# | Time (s) |
| 3 | 5 | 3 | 1 | 6 | 1.455 | 0.007 | **6** | 0.000 |
| | | | 2 | 5 | 1.738 | 0.007 | **5** | 0.000 |
| | | | 3 | 2 | 1.180 | 0.006 | **2** | 0.000 |
| | | | 4 | 4 | 1.351 | 0.007 | **4** | 0.000 |
| | | | 5 | 1 | 0.762 | 0.007 | **1** | 0.000 |
| 4 | 5 | 3 | 1 | 5 | 1.557 | 0.008 | **5** | 0.000 |
| | | | 2 | 3 | 1.814 | 0.007 | **3** | 0.000 |
| | | | 3 | 7 | 2.218 | 0.008 | **7** | 0.000 |
| | | | 4 | 5 | 1.826 | 0.007 | **5** | 0.000 |
| | | | 5 | 6 | 2.288 | 0.007 | **6** | 0.000 |
| 5 | 5 | 3 | 1 | 6 | 3.221 | 0.010 | **6** | 0.000 |
| | | | 2 | 7 | 3.492 | 0.007 | **7** | 0.000 |
| | | | 3 | 8 | 2.560 | 0.013 | **8** | 0.000 |
| | | | 4 | 6 | 2.297 | 0.008 | **6** | 0.000 |
| | | | 5 | 9 | 3.553 | 0.026 | **9** | 0.000 |
| 6 | 5 | 3 | 1 | 11 | 5.201 | 0.086 | **11** | 0.001 |
| | | | 2 | 7 | 4.182 | 0.008 | **7** | 0.001 |
| | | | 3 | 11 | 4.517 | 0.019 | **11** | 0.001 |
| | | | 4 | 7 | 3.117 | 0.016 | **7** | 0.001 |
| | | | 5 | 4 | 2.736 | 0.007 | **4** | 0.001 |
| 7 | 5 | 3 | 1 | 7 | 4.484 | 0.009 | **7** | 0.000 |
| | | | 2 | 10 | 12.370 | 0.011 | **10** | 0.000 |
| | | | 3 | 9 | 4.648 | 0.011 | **9** | 0.000 |
| | | | 4 | 8 | 5.080 | 0.010 | **8** | 0.000 |
| | | | 5 | 12 | 7.585 | 0.038 | **12** | 0.000 |
| 8 | 5 | 3 | 1 | 8 | 6.196 | 0.021 | **8** | 0.001 |
| | | | 2 | 10 | 5.727 | 0.055 | **10** | 0.001 |
| | | | 3 | 9 | 6.214 | 0.012 | **9** | 0.001 |
| | | | 4 | 10 | 5.854 | 0.013 | **10** | 0.001 |
| | | | 5 | 13 | 8785.752 | 0.022 | **13** | 0.002 |
| 4 | 6 | 4 | 1 | 10 | 3.839 | 0.017 | **10** | 0.000 |
| | | | 2 | 10 | 3.216 | 0.025 | **10** | 0.000 |
| | | | 3 | 10 | 3.229 | 0.009 | **10** | 0.000 |
| | | | 4 | 7 | 3.325 | 0.008 | **7** | 0.000 |
| | | | 5 | 9 | 2.903 | 0.013 | **9** | 0.000 |
| 5 | 6 | 4 | 1 | 16 | 7.384 | 0.540 | **16** | 0.000 |
| | | | 2 | 10 | 3.806 | 0.043 | **10** | 0.000 |
| | | | 3 | 13 | 4.492 | 0.018 | **13** | 0.000 |
| | | | 4 | 8 | 3.823 | 0.014 | **8** | 0.000 |
| | | | 5 | 16 | 6.807 | 0.579 | **16** | 0.000 |
| 6 | 6 | 4 | 1 | 17 | 39.554 | 17.476 | **17** | 0.001 |
| | | | 2 | 8 | 3.831 | 0.030 | **8** | 0.001 |
| | | | 3 | 13 | 6.124 | 0.069 | **13** | 0.001 |
| | | | 4 | 14 | 6.591 | 0.315 | **14** | 0.001 |
| | | | 5 | 15 | 7.045 | 0.076 | **15** | 0.001 |

To further analyse the performance of the beam search algorithm, Table 5 shows the comparison of beam search with the A* algorithm proposed by Expósito-Izquierdo et al. (2014) and branch and bound proposed by Expósito-Izquierdo et al. (2015). In the original paper, A* algorithm and branch and bound heuristic was limited to only 5 instances from 9 smaller instance sets. Columns 1–3 in Table 5 outline the information related to the instances. In each instance the number of empty heights at the top is 2, that is, $H = T + 2$. Columns 4 and 5 are the instances number and its optimal solution. Columns 5–6 are the computational times for both algorithms that are directly copied from Expósito-Izquierdo et al. (2015). These instances were carried out on a PC with an Intel Core 2 Duo E8500 3.16 GHz and 4 GB of RAM which is very similar to our PC. Columns 8 and 9 are the best solution and computational time by beam search with beam width at 5, respectively. Our beam search can obtain all the optimal solutions. The computational results show that our algorithm can obtain optimal solutions in all test instances with much shorter times.

## 7. Conclusions

Container relocation problem, in which containers are retrieved from the container yard with a minimum number of relocations, is one of the important operation problems at container terminals. This study proposes two simple and efficient heuristics, the smallest difference heuristic (SDH) and virtual relocation heuristic (VRH), for the container relocation problem. The concept of the SDH is based on the difference in priorities between two containers, while the VRH ignores the restriction that the top container must be treated first. The virtual relocation heuristic is embedded within a beam search algorithm to evaluate the node in the beam search. The beam search is based on a breadth-first search, and only promising nodes are kept at each level in the search tree. The beam search is designed with a simple procedure to be easily implemented, and only the beam width needs to be controlled.

Three sets of benchmark instances from literature were used to test the proposed methods. Computational results indicate that the proposed constructive heuristics outperform other related heuristics (RI, ENAR). Specifically, VRH outperformed SDH in the test instances of relatively large size proposed by Wan et al. (2009). In addition, embedding the proposed beam search algorithm with the VRH heuristic can efficiently achieve near-optimal solutions. The beam search achieved superior results in 18 of the 21 sets of instances comparing with CM proposed by Caserta et al. (2011b). It also achieved better solutions in 13 of the 14 instances presented by Lee and Lee (2010). These computational results demonstrate the efficacy of the proposed beam search compared with other algorithms.

The current research can be extended in several future directions. Only the constrained container relocation problem is considered in this paper, it would be interesting to investigate the unconstrained case. The beam search only considers partial search nodes, a branch and bound algorithm with different lower bounds and/or upper bound approaches could be developed to explore more promising nodes.

## Acknowledgement

The authors would to express their sincere thanks to Editor-in-Chief Professor J.B. Sheu and three anonymous referees for their valuable comments. This research was partially supported by a grant (MOST 105-2221-E-155-037) from the Ministry of Science and Technology of Taiwan, R.O.C.

## References

Akyuz, M.H., Lee, C.Y., 2014. A mathematical formulation and efficient heuristics for the dynamic container relocation problem. Naval Res. Logis. 61 (2), 101–118.
Carlo, H.J., Vis, I.F.A., Roodbergen, K.J., 2014. Storage yard operations in container terminals: literature overview, trends, and research directions. Eur. J. Oper. Res. 235 (2), 412–430.
Caserta, M., Schwarze, S., Voß, S., 2011a. Container rehandling at maritime container terminals. In: Böse, J., Sharda, R., Voß, S. (Eds.), Handbook of Terminal Planning, Operations Research/Computer Science Interfaces Series, vol. 49. Springer, New York, pp. 247–269.
Caserta, M., Schwarze, S., Voß, S., 2012. A mathematical formulation and complexity considerations for the blocks relocation problem. Eur. J. Oper. Res. 219 (1), 96–104.
Caserta, M., Voß, S., Sniedovich, M., 2011b. Applying the corridor method to a blocks relocation problem. OR Spectrum 33 (4), 915–929.
Expósito-Izquierdo, C., Melián-Batista, B., Marcos Moreno-Vega, J., 2014. A domain- specific knowledge-based heuristic for the blocks relocation problem. Adv. Eng. Inform. 28 (4), 327–343.
Expósito-Izquierdo, C., Melián-Batista, B., Marcos Moreno-Vega, J., 2015. An exact approach for the blocks relocation problem. Expert Syst. Appl. 42 (17), 6408–6422.
Forster, F., Bortfeldt, A., 2012. A tree search procedure for the container relocation problem. Comput. Oper. Res. 39 (2), 299–309.
Fox, M.S., 1983. Constraint-directed Search: A Case Study of Job-shop Scheduling (Ph.D. Dissertation). Carnegie Mellon University, USA.
Ji, M.J., Guo, W.W., Zhu, H.L., Yang, Y.Z., 2015. Optimization of loading sequence and rehandling strategy for multi-quay crane operations in container terminals. Transport. Res. Part E: Logis. Transport. Rev. 80, 1–19.
Jin, B., Zhu, W.B., Lim, A., 2015. Solving the container relocation problem by an improved greedy look-ahead heuristic. Eur. J. Oper. Res. 240 (3), 837–847.
Jovanovic, R., Voss, S., 2014. A chain heuristic for the blocks relocation problem. Comput. Ind. Eng. 75, 79–86.
Kim, K.H., 1997. Evaluation of the number of rehandles in container yards. Comput. Ind. Eng. 32 (4), 701–711.
Kim, K.H., Hong, G.P., 2006. A heuristic rule for relocating blocks. Comput. Oper. Res. 33 (4), 940–954.
Kim, K.H., Kim, H.B., 1999. Segregating space allocation models for container inventories in port container terminals. Int. J. Prod. Econ. 59 (1), 415–423.
Kim, K.H., Park, Y.M., Ryu, K.R., 2000. Deriving decision rules to locate export containers in container yards. Eur. J. Oper. Res. 124 (1), 89–101.
Lee, Y., Lee, Y.J., 2010. A heuristic for retrieving containers from a yard. Comput. Oper. Res. 37 (6), 1139–1147.
Lehnfeld, J., Knust, S., 2014. Loading, unloading and premarshalling of stacks in storage areas: survey and classification. Eur. J. Oper. Res. 239 (2), 297–312.
Lin, D.Y., Lee, Y.J., Lee, Y., 2015. The container retrieval problem with respect to relocation. Transport. Res. Part C: Emerg. Technol. 52, 132–143.

Lowerre, B.T., 1976. The Harpy Speech Recognition System (Ph.D. Dissertation). Carnegie Mellon University, USA.
Murty, K.G., Wan, Y.W., Liu, J.Y., 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.
Ow, P.S., Morton, T.E., 1988. Filtered beam search in scheduling. Int. J. Prod. Res. 26 (1), 35–62.
Ow, P.S., Smith, S.F., 1988. Viewing scheduling as an opportunistic problem-solving process. Ann. Oper. Res. 12 (1), 85–108.
Petering, M.E.H., Hussein, M.I., 2013. A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem. Eur. J. Oper. Res. 231 (1), 120–130.
Rei, R.J., Pedroso, J.P., 2012. Heuristic search for the stacking problem. Int. Trans. Oper. Res. 19 (3), 379–395.
Rubin, S., 1978. The ARGOS Image Understanding System (Ph.D. Dissertation). Carnegie Mellon University, USA.
Sabuncuoglu, I., Bayiz, M., 1999. Job shop scheduling with beam search. Eur. J. Oper. Res. 118 (2), 390–412.
Sabuncuoglu, I., Gocgun, Y., Erel, E., 2008. Backtracking and exchange of information: methods to enhance a beam search algorithm for assembly line scheduling. Eur. J. Oper. Res. 186 (3), 915–930.
Tang, L., Jiang, W., Liu, J., Dong, Y., 2015. Research into container reshuffling and stacking problems in container terminal yards. IIE Trans. 47 (7), 751–766.
Ünluyurt, T., Aydin, C., 2012. Improved rehandling strategies for the container retrieval process. J. Adv. Transport. 46 (4), 378–393.
Valente, J.M.S., Alves, R.A.F.S., 2005. Filtered and recovering beam search algorithms for the early/tardy scheduling problem with no idle time. Comput. Ind. Eng. 48 (2), 363–375.
Wan, Y.W., Liu, J., Tsai, P.C., 2009. The assignment of storage locations to containers for a container stack. Naval Res. Logis. 56 (8), 699–713.
Wu, K.C., Ting, C.J., Hernández, R., 2010. Applying tabu search for minimizing reshuffle operations at container yards. J. Eastern Asia Soc. Transport. Stud. 8, 2379–2393.
Wu, K.C., Ting, C.J., 2010. A beam search algorithm for minimizing reshuffle operations at container yards. In: Proceedings of the 2010 International Conference on Logistics and Maritime Systems, Busan, Korea, September 15–17, 2010.
Yang, J.H., Kim, K.H., 2006. A grouped storage method for minimizing relocations in block stacking systems. J. Intell. Manuf. 17 (4), 453–463.
Zehendner, E., Feillet, D., 2014. A branch and price approach for the container relocation problem. Int. J. Prod. Res. 52 (24), 7159–7176.
Zhang, C., Chen, W., Shi, L., Zheng, L., 2010. A note on deriving decision rules to locate export containers in container yards. Eur. J. Oper. Res. 205 (2), 483–485.
Zhao, W.J., Goodchild, A.V., 2010. The impact of truck arrival information on container terminal rehanling. Transport. Res. Part E: Logis. Transport. Rev. 46 (3), 327–343.
Zhu, W., Qin, H., Lim, A., Zhang, H., 2012. Iterative deepening A$^*$ algorithms for the container relocation problem. IEEE Trans. Autom. Sci. Eng. 9 (4), 710–722.