European Journal of Operational Research 259 (2017) 48–62

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Production, Manufacturing and Logistics

An algorithm with performance guarantee for the Online Container Relocation Problem

Elisabeth Zehendner[a], Dominique Feillet[a,*], Patrick Jaillet[b]

[a] Ecole des Mines de Saint-Etienne and LIMOS UMR CNRS 6158, CMP Georges Charpak, Gardanne F-13541, France
[b] Laboratory for Information and Decision Systems, Department of Electrical Engineering and Computer Science, and Operations Research Center, MIT, Cambridge, MA 02139, USA

ARTICLE INFO

Article history:
Received 27 November 2015
Accepted 8 September 2016
Available online 14 September 2016

Keywords:
Container terminal
Algorithm with performance guarantee
Container relocation
Online algorithm

ABSTRACT

This paper introduces and investigates the Online Container Relocation Problem, where containers have to be retrieved from a bay in a container terminal so as to minimize the number of relocations. Unlike the offline version of the problem, the order of container retrievals is revealed one at a time in an online fashion. We analyze the so-called leveling heuristic using the perspective of worst-case competitive analysis of online algorithms and derive its competitive ratio. We then provide some computational experiments which give insights on the actual average performance of the heuristic.

1. Introduction

Container terminals stack containers on top of each other so as to use their scarce land efficiently. The drawback of stacking is that only the topmost container of each stack can be accessed directly.

If another container has to be retrieved, parasite movements – called relocations, reshuffles or rehandles – are necessary to free the target container. Relocations increase the time needed to retrieve containers and thereby decrease the overall productivity of the terminal. However, relocations cannot be avoided completely as little information about future retrievals is known when containers have to be stored or relocated, or because of limited yard space.

The following problems have been addressed for yard optimization: the storage space allocation problem to determine storage locations for incoming containers; the yard crane assignment/scheduling problem to assign yard cranes to storage areas and to define a schedule of storage and retrieval operations for each crane; the remarshalling/premarshalling problem to reorganize parts of the storage area (a block/a bay) in less busy periods as new information becomes available to reduce the number of relocations during the subsequent retrieval process; the container relocation problem to retrieve all containers from a bay in a given sequence with a minimum number of relocations. For a comprehensive literature review on problems related to container storage, retrieval and rehandling at terminals see Caserta, Schwarze, and Voß (2011) and Lehnfeld and Knust (2014).

This study deals with the container relocation problem (CRP) which is NP-hard (Caserta, Schwarze, & Voß, 2012). Few exact and several heuristic solution approaches exist. Exact solution approaches use integer linear programs to model and solve the problem (Caserta et al., 2012; Lee & Hsu, 2007; Petering & Hussein, 2013; Tang, Zhao, & Liu, 2012; Zehendner, Caserta, Feillet, Schwarze, & Voß, 2015) or a branch and price approach (Zehendner & Feillet, 2014). Most of the heuristic approaches are based on branch and bound and apply different branching and exploring strategies (Caserta, Schwarze, & Voß, 2009; Caserta & Voß, 2009; Forster & Bortfeldt, 2012; Kim & Hong, 2006; Rei & Pedroso, 2012; Ünlüyurt & Aydin, 2012; Wu & Ting, 2010; Zhang, Guo, Zhu, Lim, & Cheang, 2010; Zhu, Qin, Lim, & Zhang, 2012). Other authors use tabu-search (Wu, Hernández, & Ting, 2009), or the so-called corridor method (Caserta et al., 2011).

The dynamic container relocation problem (DCRP) is an extension of the CRP where containers are both received and retrieved from a single yard-bay. The arrival (departure) sequences of containers to (from) the yard-bay is assumed to be known a priori. Akyüz and Lee (2014) present a BIP formulation and three types of heuristics for the DCRP. Borjian, Manshadi, Barnhart, and Jaillet (2015) introduce the time-based DCRP. They require that each retrieval and stacking operation is completed within a given service time window but do not impose a sequence of operations.

* Corresponding author.
E-mail addresses: zehendner@emse.fr (E. Zehendner), feillet@emse.fr (D. Feillet), jaillet@mit.edu (P. Jaillet).

http://dx.doi.org/10.1016/j.ejor.2016.09.011
0377-2217/© 2016 Elsevier B.V. All rights reserved.

They aim to jointly minimize the number of relocation moves and service time. They propose an IP for this problem.

All these papers deal with the offline problem where the entire retrieval/storage sequence of containers or their time windows are given in advance. In practice, the exact retrieval/storage sequence of containers in a bay is not known in advance. Especially, for import containers the retrieval sequence is revealed over time as trucks arrive at the terminal.

Only a few articles deal with stacking and/or relocation operations of containers with uncertain arrival and departure times. Borgman, van Asperen, and Dekker (2010); Dekker, Voogd, and van Asperen (2006); Duinkerken, Evers, and Ottjes (2001) use simulation to compare different storage space assignment and reshuffle strategies with different levels of departure time information. van Asperen, Borgman, and Dekker (2013) use simulation to analyze the impact of truck announcements on different storage space assignment and reshuffle strategies. Yu and Qi (2013) use simulation to compare single-period and multiple-period strategies to allocate arriving containers to storage space with regard to the average expected container retrieval time. These studies model the entire yard and consider storage and retrieval operations. Yang and Kim (2006) deal with the problem of assigning arriving containers to the yard so that the expected number of reshuffles is minimized. Jang, Kim, and Kim (2013) show that the number of relocations can be reduced if the load type of an arriving container is used to determine its storage location.

Zhao and Goodchild (2010) compare different levels of truck arrival information: complete truck arrival sequence, partial arrival sequence, information on arrival of groups of containers. They show that the complete arrival sequence is not required to substantially reduce the number of reshuffles. Wan, Liu, and Tsai (2009) tackle the CRP and the DCRP. They present an IP to minimize the total number of reshuffles for CRP, called MRIP. They propose a MRIP-based heuristic which generates a solution by solving a series of reduced MRIP$_K$ models which solve the problem optimally for the next K containers. For CRP and DCRP, they compare the MRIP$_K$ heuristics with different values of K to the lowest-slot heuristic, the reshuffling index heuristic (Murty et al., 2005) and the ENAR (expected number of additional relocation) heuristic (Kim & Hong, 2006). Borjian, Galle, Manshadi, Barnhart, and Jaillet (2015) introduce the so-called CRP with incomplete information, where the retrieval order of a subset of containers is known initially and the retrieval order of the remaining containers is revealed all at once at a later time. They assume a probabilistic distribution on the unknown retrieval orders and propose a 2-stage approximate stochastic optimization algorithm extending the A^* algorithm.

All these papers show that using information on container arrival and departure times (real or expected, for single containers or for groups of containers) reduces the number of reshuffles and the retrieval time considerably.

This study deals with the Online Container Relocation Problem (OCRP) where the retrieval sequence of containers is revealed over time. We investigate the worst-case situation where no information on the retrieval time or sequence is available. Kim (1997) proposes a methodology to estimate the expected number of relocations to pick up an arbitrary container and the total number of relocations to pick up all containers in a bay for a given initial bay configuration. They also run experiments to show the impact of the height and the width of a bay on the estimated number of relocations. They assume that all containers are equally likely to be picked up next and that a container is always relocated to the lowest empty position.

We provide evidence why relocating a container to the lowest empty position minimizes the expected number of relocations and give a worst-case analysis for this online heuristic. In addition, we report computational results giving insights about the performance of the heuristic in practice and compare it to other heuristics that also operate without the knowledge of the container retrieval sequence. The OCRP is introduced in Section 2. Section 3 presents the online relocation strategy: the leveling heuristic L. Section 4 provides a brief introduction to online optimization and proves results on the competitive ratio of leveling heuristic L. Section 5 reports our computational experiments and presents the average and worst case performance of the heuristic. Section 6 concludes the paper.

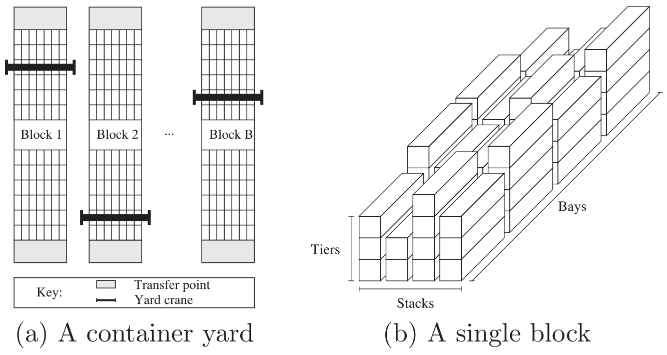


Fig. 1. Blocks, bays, stacks and tiers.

2. Online Container Relocation Problem

Generally, container terminals have no or little information on departure times as well as the departure sequence of containers (those picked-up by trucks) at bays. It is common that terminals obtain this information late when trucks check in at the terminal gate. If the target container is not on the top of its stack, blocking containers have to be relocated before the target container can be retrieved. The decision of where to relocate these containers has to be taken in real-time.

The container relocation problem studied in literature supposes that the whole retrieval sequence is known in advance. On the other hand, the Online Container Relocation Problem with incomplete information assumes that partial (or incomplete) information is available on container departures. Both cases apply to container terminals using yard cranes for storage operations. In this case, the yard area is divided into blocks as illustrated in Fig. 1. Each block consists of several bays, each bay has several stacks and each stack contains several tiers. The objective is to retrieve N containers from a bay with W stacks and H tiers in a given sequence with minimum number of relocations.

In the online case, the container retrieval sequence is revealed over time. Hence, relocation decisions have to be taken with no / limited knowledge of future retrievals. To represent this limited knowledge, we introduce a *look-ahead horizon* $\mathcal{H}$. It indicates how many future retrievals are known when a container is retrieved. If $\mathcal{H} = 0$, only the current retrieval container is known. If $\mathcal{H} \geq 1$, the current retrieval container and the next $\mathcal{H}$ retrieval containers are known. If $\mathcal{H} = N - 1$, the entire sequence is known in advance (offline case). We coin the OCRP with a *look-ahead horizon* $\mathcal{H}$ as OCRP_H.

The problem definition relies on assumptions A1–A8.

A1: The initial bay layout is known.
A2: The bay size is limited by the maximum numbers of stacks, W, and tiers, H.
A3: Only the topmost container of a stack can be picked up. A relocated container can only be put on top of another stack or on the ground.

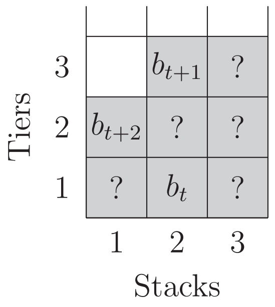


Fig. 2. Online Container Relocation Problem with incomplete information and $\mathcal{H} = 2$. Layout and retrieval information before retrieving container b_t.

A4: Containers are only relocated within the bay since relocations between bays are very time consuming.
A5: The distance traveled within one bay (horizontally and vertically) has little impact on the time to relocate or to retrieve containers.
A6: Containers in the same bay have the same size and can be piled up in any order.
A7: No new containers arrive during the retrieval process.
A8: The retrieval sequence is revealed over time. At each retrieval, only the next $\mathcal{H}$ containers to be retrieved are known.

Several variants of the offline container relocation problem exist which also apply to its online version: assumptions A9 and A10.

A9: Only containers located above the current target container may be relocated.
A10: Precedence constraints exist between groups of containers rather than between single containers.

We deal with the case where assumption A9 holds and assumption A10 does not. Removing assumption A9 or introducing assumption A10 leads to quite different stacking policies, which are outside the scope of this paper (see, e.g., Forster and Bortfeldt, 2012 or Petering and Hussein, 2013 for some results on these cases).

Containers are represented with $b_1, \ldots, b_N$, and assumed to be ranked in the order of their retrieval. Fig. 2 illustrates the problem with a look-ahead horizon $\mathcal{H} = 2$: we know the current retrieval container b_t and the next two retrieval containers b_{t+1} and b_{t+2}. But, we have no information on the retrieval sequence of the other containers. To retrieve container b_t, an unrevealed container and b_{t+1} have to be relocated. After relocating these two containers and retrieving b_t, container b_{t+3} is revealed. The information about the new bay configuration, the container to be retrieved and the next two retrieval containers become available afterwards.

In the rest of the paper we focus on the case $\mathcal{H} = 0$ (OCRP_0). The leveling heuristic we propose does not take account of any information on future retrievals.

We call the time interval (or the set of operations) associated with the retrieval of one container a *period*. At each period t, the container b_t has to be retrieved from the bay. Hence, periods go from period 1 (retrieval of container b_1 and associated relocations) to period N (retrieval of the last container b_N). We call *positions* the places in the bay, where containers can be located. A position is expressed by the pair (stack, tier) in $[1, \ldots, W] \times [1, \ldots, H]$. Position (1, 1) is the lowest position on the left. We denote $h(p)$ as the height of position p and $w(p)$ as the stack number of position p. The position of container b at period t is given by p_t^b; the initial container positions are given by p_1^b. Please note that for a given instance of the OCRP, the initial position of all containers are fixed but only revealed for some containers. We call blocking containers those that are located above a container with an earlier retrieval in the initial layout. We denote $\mathcal{B}$ as the set of blocking containers. It includes all containers that have to be relocated at least once, but does not indicate when and how often containers have to be relocated.

$$\mathcal{B} = \{b_t | t \in \{1, \ldots, N\}, \exists l \in \{1, \ldots, t-1\}, w(p_1^{b_t}) = w(p_1^{b_l}), h(p_1^{b_t}) > h(p_1^{b_l})\} \tag{1}$$

Note that containers $b \notin \mathcal{B}$ are never relocated.

3. Relocation strategy: the leveling heuristic

This section presents an online relocation strategy. It uses no information on future retrievals ($\mathcal{H} = 0$) and uses the simple strategy of relocating containers to the lowest stack. If several stacks have the same height, the leftmost stack among them is chosen. This heuristic is called leveling heuristic L or simply heuristic L.

If a feasible solution exists, heuristic L always respects the maximum number of tiers since it always relocates a container to the lowest empty position. Furthermore, if heuristic L does not find a feasible solution, no feasible solution exists. We prove this using the following property.

Property 1. *An instance of the OCRP admits a feasible solution if and only if condition (2) below holds for every $t \in \{1, \ldots, N\}$ such that $b_t \notin \mathcal{B}$:*

$$N - t \leq (W - 1) \times H + h(p_1^{b_t}) - 1. \tag{2}$$

Proof. Let us assume that there exists a period t such that $b_t \notin \mathcal{B}$ and condition (2) does not hold: $N - t > (W-1) \times H + h(p_1^{b_t}) - 1$. Because $b_t \notin \mathcal{B}$, it is never relocated: $p_t^{b_t} = p_1^{b_t}$. After the retrieval of b_t, $N - t$ containers remain in the bay, with exactly $h(p_1^{b_t}) - 1$ containers located in stack $w(p_1^{b_t})$. Given that condition (2) does not hold, the $W - 1$ remaining stacks are not sufficient to receive $N - t - (h(p_1^{b_t}) - 1)$ containers and no feasible solution exists.

Let us now show that if condition (2) holds for every $t \in \{1, \ldots, N\}$ such that $b_t \notin \mathcal{B}$, a feasible solution exists. We prove it by showing that under this assumption heuristic L constructs a feasible solution. Let t be a period such that $b_t \notin \mathcal{B}$. We know that the retrieval container b_t is located at position $p_t^{b_t} = p_1^{b_t}$ and that available positions exist for relocating containers located above b_t, by condition (2). Because heuristic L relocates containers to lowest stacks, it will only perform feasible relocations. Let now t be a period with $b_t \in \mathcal{B}$. Container b_t has been relocated at least once before period t. Let us call t' ($t' < t$) the period of its last relocation. Containers above b_t at period t have all been moved to stack $w(p_t^{b_t})$ between period t' and $t - 1$ and have left an equivalent number of empty positions in the rest of the bay. Consequently, again, heuristic L will apply feasible relocations. All relocations of heuristic L being feasible, the solution is feasible. □

Note that condition (2) might only be violated for small values of t: it is always true for $t \geq H$.

Property 2. *If an instance of the OCRP admits a feasible solution, heuristic L constructs a feasible solution.*

Proof. See proof of Property 1 □

We now give some insights on the rationale behind the design of heuristic L. For $\mathcal{H} = 0$, no information on future retrievals is given and all containers in the bay are assumed to be equally likely to be retrieved next. The *expected value of relocations*, $\mathbb{E}(R)$, represents the expected number of relocations needed to retrieve *the next* container from a given bay. The expected value $\mathbb{E}(R)$ depends on the number of containers in the bay, N, the number of stacks, W, and on the number of containers per stack,

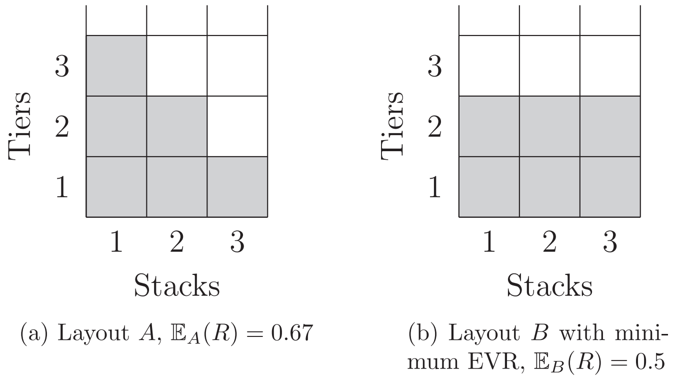

Tiers
3
2
1
1 2 3
Stacks
(a) Layout A, $\mathbb{E}_A(R) = 0.67$
(b) Layout B with minimum EVR, $\mathbb{E}_B(R) = 0.5$

Fig. 3. Expected value of relocations $\mathbb{E}(R)$ for two different layouts with 6 containers.

denoted with $s(w)$. It is defined by Eq. (3). Fig. 3 illustrates the computation $\mathbb{E}(R)$ for two examples. For layout A, we obtain $\mathbb{E}_A(R) = 1/6 \cdot ((0+1+2)+(0+1)+(0)) = 0.67$. For layout B, we obtain $\mathbb{E}_B(R) = 1/6 \cdot ((0+1)+(0+1)+(0+1)) = 0.5$.

$$\mathbb{E}(R) = \frac{1}{N} \cdot \sum_{w=1}^{W} \sum_{h=1}^{s(w)} (s(w) - h) = \frac{1}{N} \cdot \sum_{w=1}^{W} \sum_{h=0}^{s(w)-1} h \tag{3}$$

In the sequel, we show that $\mathbb{E}(R)$ is minimal if containers are equally distributed among all stacks. Therefore, by relocating containers to the lowest stacks, the leveling heuristic L aims at minimizing the expected value of relocations $\mathbb{E}(R)$. This proves the statement "When every container has the same probability to be picked up next, it minimizes the expected number of rehandles for the next pick-up to position rehandled containers at the lowest possible slots" (Kim, 1997).

Note that the minimum possible difference between the lowest and the highest stack equals 0 if $N \equiv 0 (\bmod W)$ and 1 if $N \not\equiv 0 \pmod W$. Indeed, the minimum difference between the lowest and the highest stack is obtained if containers are evenly distributed among stacks. In this case, either each stack has a height of N/W (when $N \equiv 0 (\bmod W)$) or some stacks have height $\lceil N/W \rceil$ and others $\lfloor N/W \rfloor$ (when $N \not\equiv 0 \pmod W$).

Property 3. *The expected value of relocations $\mathbb{E}(R)$ is minimal if and only if containers are evenly distributed among stacks.*

Proof. We first prove that if the difference is not minimal, the expected value of relocations is not minimal. We consider layout 1 such that the difference between the lowest stack and the highest stack is not minimal. Let i be the lowest stack in layout 1 and j the highest stack: $s_1(j) - s_1(i) > 1$. We show that $\mathbb{E}_1(R)$ is not minimal. We construct layout 2 by moving the topmost container from stack j to stack i. We compare the expected value of relocations for layouts 1 and 2 and show that it is negative:

$$\begin{aligned} N \times (\mathbb{E}_2(R) - \mathbb{E}_1(R)) \\ &= \left(\sum_{h=0}^{s_2(i)-1} h + \sum_{h=0}^{s_2(j)-1} h \right) - \left(\sum_{h=0}^{s_1(i)-1} h + \sum_{h=0}^{s_1(j)-1} h \right) \\ &= \left(\sum_{h=0}^{s_1(i)} h + \sum_{h=0}^{s_1(j)-2} h \right) - \left(\sum_{h=0}^{s_1(i)-1} h + \sum_{h=0}^{s_1(j)-1} h \right) \\ &= s_1(i) - (s_1(j) - 1) < 0. \end{aligned}$$

If the difference between the lowest stack and the highest stack is minimal, the expected value of relocations is constant whatever the positions of the containers; the expected value of relocations is thus always minimal, which concludes the proof. □

Different simulation studies on reshuffle strategies (Borgman et al., 2010; Dekker et al., 2006; Duinkerken et al., 2001; van Asperen et al., 2013) also show that the leveling heuristic outperforms other heuristics (e.g., random, closest free position) that do not use information on container retrieval times. However, it is obvious that the leveling heuristic which knows only the current retrieval container, does often lead to more relocations than the optimal offline relocation strategy that has complete information on the retrieval order. Consider a bay with a high stack where all containers have to be retrieved after the relocation container and a low stack where one container has to be retrieved before the relocation container. In this case, it is better to relocate the container on top of the high stack. This decision is based on the knowledge of the retrieval sequence which is not available to the leveling heuristic which assumes that all containers are equally likely to be retrieved next. In other cases, it might also be advantageous to relocate containers to higher stacks to prepare the stack for further relocations or to leave positions free for containers that have to be relocated in subsequent periods. This is especially true for bigger bays where containers need to be relocated several times.

4. Competitive ratio analysis of leveling heuristic *L*

4.1. Introduction to online optimization

The online optimization approach assumes that the problem data is revealed incrementally, but makes no assumption whatsoever about the structure of the uncertainty. *Competitive analysis* is used to evaluate the quality of online algorithms. It compares the performance of an online algorithm with no knowledge of the future (online) with the performance of an optimal strategy that has complete knowledge of the future (offline). The *competitive ratio* of an online algorithm is the worst-case ratio between the cost of the solution found by the online algorithm and the cost of an optimal offline solution. For minimization problems, an online algorithm is said to be *c-competitive* ($c \geq 1$) if

$$cost_{online}(I) \leq c \cdot cost_{optimal}(I), \qquad \text{for all problem instances } I.$$

For more information on online optimization please refer to (Borodin & El-Yaniv, 1998; Grötschel, Krumke, Rambau, Winter, & Zimmermann, 2001; Jaillet & Wagner, 2010).

In the context of the Online Container Relocation Problem, the objective is to minimize the number of relocations. The number of retrievals is not considered as it is always equal to the number of initial containers in the bay and does not depend on the chosen relocation strategy. Property 4 gives a simple lower bound on the number of relocations for an offline strategy.

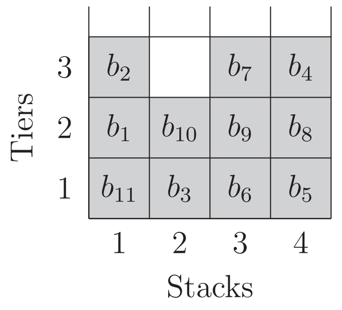
Property 4. *For a given instance I, the optimal offline solution $R^*(I)$ performs at least $|\mathcal{B}|$ relocations.*

Proof. In the offline case, the initial position of each container and the entire retrieval sequence is known for a given instance I. Blocking containers are placed above containers with earlier departure times. Each of these blocking containers has to be relocated at least once. Hence, $R^*(I) \geq |\mathcal{B}|$ □

Using the lower bound provided by Property 4 for offline algorithms, we prove that the competitive ratio of our leveling heuristic L is $2\lceil \frac{N}{W} \rceil - 1$ for a bay with W stacks and N containers.

4.2. Principle of the proof and notation

To determine the competitiveness ratio of the leveling heuristic L we have to determine an upper bound on the number of relocations that it performs. The idea behind this proof is to show that the height of positions to which blocking containers are relocated decreases over time. Since only blocking containers can be relocated, no container can be relocated out of positions



$$
\begin{aligned}
\mathcal{B}_1 &= \{b_2, b_7, b_8, b_9, b_{10}\}\\
\mathcal{B}_2 &= \{b_2, b_7, b_8, b_9, b_{10}\}\\
\mathcal{B}_3 &= \{b_7, b_8, b_9, b_{10}\}\\
\mathcal{B}_4 &= \mathcal{B}_5 = \mathcal{B}_6 = \mathcal{B}_7 = \{b_7, b_8, b_9, b_{10}\}\\
\mathcal{B}_8 &= \{b_8, b_9, b_{10}\}\\
\mathcal{B}_9 &= \{b_9, b_{10}\}\\
\mathcal{B}_{10} &= \{b_{10}\}\\
\mathcal{B}_{11} &= \emptyset
\end{aligned}
$$

Fig. 4. Illustration of $\mathcal{B}_t$ for $t = 1, \dots, N$ ($B = 5$).

at height 1. Consequently, the maximum number of relocations depends on the height of the stacks over time.

As explained in Section 3, heuristic L is not influenced by the stack level limit and it finds a feasible solution if there exists one (see Property 2). For the proof, we assume that the finite bay height H is high enough to guarantee a feasible solution. Then, the bay height has no impact on the solution proposed by the leveling heuristic and we can express the competitiveness ratio as a function of the bay width W and the number of containers N in the bay.

The proof is based on the following three families of sets.

1. Potentially blocking containers $\mathcal{B}_t$ for $t = 1, \dots, N$

The set $\mathcal{B}_1$ is the set of blocking containers (in the initial layout): $\mathcal{B}_1 = \mathcal{B}$. Sets $\mathcal{B}_t$ for $t = 2, \dots, N$ are the sets of blocking containers still present in the bay at period t: $\mathcal{B}_t = \mathcal{B} \setminus \{b_l | l < t\}$. Especially, $\mathcal{B}_t$ contains all containers that might be relocated at period t. We note B the number of blocking containers: $B = |\mathcal{B}|$. Fig. 4 illustrates the definition of sets $\mathcal{B}_t$.

2. Qualifying positions $\mathcal{D}_t$ for $t = 1, \dots, N$

The qualifying positions $\mathcal{D}_t$ indicate the set of lowest positions where potentially blocking containers are or can be (empty positions) located. It contains the lowest $|\mathcal{B}_t|$ positions that are empty or contain a container $b \in \mathcal{B}_t$, at period t.

Algorithm 1 describes the construction of $\mathcal{D}_t$. It iterates over positions and adds qualifying ones to the set $\mathcal{D}_t$ until $|\mathcal{D}_t| = |\mathcal{B}_t|$. Since we want to find the lowest positions, the algorithm starts at position (1, 1). For each tier h ($h = 1, \dots, H$), it iterates over all stacks w ($w = 1, \dots, W$). For the proof, we need (i) to give priority to positions containing a blocking container over empty positions and (ii) $\mathcal{D}_t$ to be similar to $\mathcal{D}_{t-1}$ which is obtained by priorizing positions with a blocking container that were already part of $\mathcal{D}_{t-1}$. Therefore, the algorithm adds positions (w, h) with the same height h in the following sequence: first positions that contain a container $b \in \mathcal{B}_t$ and that already belonged to $\mathcal{D}_{t-1}$, then positions containing containers $b \in \mathcal{B}_t$ that did not belong to $\mathcal{D}_{t-1}$ and finally empty positions.

We define $h(\mathcal{D}_t)$ as the height of set $\mathcal{D}_t$: $h(\mathcal{D}_t) = \sum_{p \in \mathcal{D}_t} h(p)$. B_t^{out} indicates how many containers $b \in \mathcal{B}_t$ are located at positions $p_t^b \notin \mathcal{D}_t$: $B_t^{out} = |\{b | b \in \mathcal{B}_t \wedge p_t^b \notin \mathcal{D}_t\}|$. Fig. 5b and a illustrate the definition of $\mathcal{D}_t$, $h(\mathcal{D}_t)$ and B_t^{out}. For the sake of simplifying the subsequent notation, we define $\mathcal{D}_{N+1} = \emptyset$ and $B_{N+1}^{out} = 0$.

3. Relocation containers $\mathcal{X}_t$ for $t = 1, \dots, N$

The set $\mathcal{X}_t \subseteq \mathcal{B}_t$ represents the set of containers that are actually relocated in period t. It does not include the retrieval container. We distinguish three different types of relocations:

- $\mathcal{X}_t^\circ$ the set of containers relocated from $p_t^b \in \mathcal{D}_t$ to $p_{t+1}^b \in \mathcal{D}_{t+1}$
  $\mathcal{X}_t^\circ = \{b | b \in \mathcal{X}_t \wedge p_t^b \in \mathcal{D}_t \wedge p_{t+1}^b \in \mathcal{D}_{t+1}\}$
- $\mathcal{X}_t^+$ the set of containers relocated from $p_t^b \notin \mathcal{D}_t$ to $p_{t+1}^b \in \mathcal{D}_{t+1}$
  $\mathcal{X}_t^+ = \{b | b \in \mathcal{X}_t \wedge p_t^b \notin \mathcal{D}_t \wedge p_{t+1}^b \in \mathcal{D}_{t+1}\}$

Algorithm 1 Construction of $\mathcal{D}_t$.

The algorithm evaluates positions in the bay (stack w, tier h) and adds qualifying positions to $\mathcal{D}_t$ until $|\mathcal{D}_t| = |\mathcal{B}_t|$.

Input: $\mathcal{B}_1$, $\mathcal{D}_{t-1}$, bay layout at period t

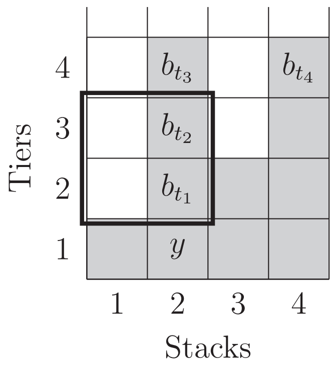
```
$\mathcal{B}_t \leftarrow \{b \mid b \in \mathcal{B}_1 \wedge b \geq t\}$
$\mathcal{D}_t \leftarrow \emptyset$
$h \leftarrow 1$
while $|\mathcal{D}_t| < |\mathcal{B}_t|$ do
    $w \leftarrow 1$
    while $|\mathcal{D}_t| < |\mathcal{B}_t|$ and $w \leq W$ do
        if a container $\in \mathcal{B}_t$ is placed at $(w, h)$ and $(w, h) \in \mathcal{D}_{t-1}$ then
            add $(w, h)$ to $\mathcal{D}_t$
        end if
        $w \leftarrow i + 1$
    end while
    $w \leftarrow 1$
    while $|\mathcal{D}_t| < |\mathcal{B}_t|$ and $w \leq W$ do
        if a container $\in \mathcal{B}_t$ is placed at $(w, h)$ and $(w, h) \notin \mathcal{D}_{t-1}$ then
            add $(w, h)$ to $\mathcal{D}_t$
        end if
        $w \leftarrow i + 1$
    end while
    $w \leftarrow 1$
    while $|\mathcal{D}_t| < |\mathcal{B}_t|$ and $w \leq W$ do
        if if position $(w, h)$ is empty then
            add $(w, h)$ to $\mathcal{D}_t$
        end if
        $w \leftarrow i + 1$
    end while
    $h \leftarrow h + 1$
end while
return $\mathcal{D}_t$
```

- $\mathcal{X}_t^-$ the set of containers relocated from $p_t^b \in \mathcal{D}_t$ to $p_{t+1}^b \notin \mathcal{D}_{t+1}$
  $\mathcal{X}_t^- = \{b | b \in \mathcal{X}_t \wedge p_t^b \in \mathcal{D}_t \wedge p_{t+1}^b \notin \mathcal{D}_{t+1}\}$

Fig. 5a represents the situation of the bay depicted in Fig. 5b at period $t + 1$ after retrieving container $y = b_t$ and relocating containers b_{t_1}, b_{t_2} and b_{t_3} according to the leveling heuristic L. It illustrates sets $\mathcal{X}_t$, $\mathcal{X}_t^\circ$, $\mathcal{X}_t^+$ and $\mathcal{X}_t^-$.

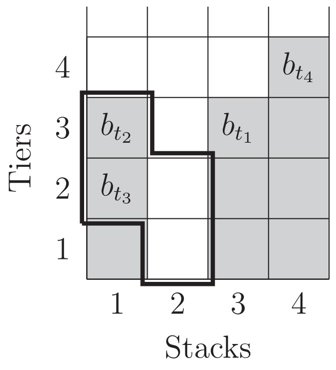
4.3. Proof of the competitiveness ratio of leveling heuristic L

The proof is based on several lemmata (Lemmata 2–6) presented in Appendix A, plus Lemma 1 below.



$$\mathcal{D}_t = \{(2,2),(1,2),(2,3),(1,3)\}$$
$$h(\mathcal{D}_t) = 2+2+3+3 = 10$$
$$B_t^{out} = 2\ (b_{t_3} \text{ and } b_{t_4})$$

(a) Bay at period t with $\mathcal{B}_t = \{b_{t_1}, b_{t_2}, b_{t_3}, b_{t_4}\}$



$$\mathcal{B}_{t+1} = \{b_{t_1}, b_{t_2}, b_{t_3}, b_{t_4}\}$$
$$\mathcal{D}_{t+1} = \{(2,1),(1,2),(2,2),(1,3)\}$$
$$h(\mathcal{D}_{t+1}) = 1+2+2+3 = 8$$
$$B_{t+1}^{out} = 2\ (b_{t_1} \text{ and } b_{t_4})$$
$$\mathcal{X}_t = \{b_{t_1}, b_{t_2}, b_{t_3}\}$$
$$\mathcal{X}_t^+ = \{b_{t_3}\}$$
$$\mathcal{X}_t^\circ = \{b_{t_2}\}$$
$$\mathcal{X}_t^- = \{b_{t_1}\}$$

(b) Bay at period $t+1$ after the retrieval of container y

Fig. 5. Illustration of $\mathcal{D}_t$, $h(\mathcal{D}_t)$, B_t^{out} and $\mathcal{X}_t$, $\mathcal{X}_t^\circ$, $\mathcal{X}_t^+$, $\mathcal{X}_t^-$.

Lemma 2 states that $|\mathcal{X}_t| = |\mathcal{X}_t^-| + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^+|$ for all $t = 1, \dots, N$. This lemma shows that a container relocated in t which did not belong to $\mathcal{D}_t$ (because it is too high, as b_{t_3} in Fig. 5a) necessarily belongs to $\mathcal{D}_{t+1}$. This is illustrated by Fig. 5 b. The size of $\mathcal{D}_t$ is the size of $\mathcal{B}_t$, i.e., 4 here. Because b_{t_3} and b_{t_4} are not in $\mathcal{D}_t$, there are two empty positions in $\mathcal{D}_t$, where b_{t_3} can be relocated to.

Lemmata 3–6 analyze the relation between $\mathcal{D}_t$, B_t^{out} and $\mathcal{X}_t$ for cases where the retrieval container y belongs to the set of potentially blocking containers ($y \in \mathcal{B}_t$) and where it does not ($y \notin \mathcal{B}_t$). For all periods $t = 1, \dots, N$,

- $|\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| \le h(\mathcal{D}_t) - h(\mathcal{D}_{t+1})$ if $y \notin \mathcal{B}_t$ (from Lemma 3);
- $|\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| \le h(\mathcal{D}_t) - h(\mathcal{D}_{t+1}) - 1$ if $y \in \mathcal{B}_t$ (from Lemma 4);
- $|\mathcal{X}_t^\circ| + |\mathcal{X}_t^+| \le h(\mathcal{D}_t) - h(\mathcal{D}_{t+1}) + B_t^{out} - B_{t+1}^{out}$ if $y \notin \mathcal{B}_t$ (from Lemma 5);
- $|\mathcal{X}_t^\circ| + |\mathcal{X}_t^+| \le h(\mathcal{D}_t) - h(\mathcal{D}_{t+1}) + B_t^{out} - B_{t+1}^{out} - 1$ if $y \in \mathcal{B}_t$ (from Lemma 6).

We illustrate these lemmata with the help of Fig. 5a and b. In this example, $y \notin \mathcal{B}_t$, hence only Lemmata 3 and 5 are concerned. Regarding Lemma 3, $|\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = 2$. The lemma thus states that the height of the qualifying position set decreases by at least 2. The reason is as follows. Due to the fact that $|\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = 2$, there are two containers that are located above y in $\mathcal{D}_t$ at period t and therefore, one qualifying position exists at height $h(p_y^t) + 2 = 3$ (occupied by b_{t_2} here). When y is retrieved, a new empty position appears at height 1 in the qualifying position set and replaces one whose height is at least 3 (actually exactly 3 in the example): it permits to decrease by 2 the height of the qualifying position set. Lemma 5 relies on the fact that (in this example) $B_{t+1}^{out} \le B_t^{out} - |\mathcal{X}_t^+| + |\mathcal{X}_t^-|$. Summing this inequality with the inequality from Lemma 3 indeed gives Lemma 5. Inequality $B_{t+1}^{out} \le B_t^{out} - |\mathcal{X}_t^+| + |\mathcal{X}_t^-|$ can be explained as follows. The (blocking) containers that contribute to B_{t+1}^{out} can be split in two subsets: those that contribute also to B_t^{out} and those that do not. The sizes of these two subsets can be upper bounded. Containers b_{t_3} and b_{t_4} contribute to B_t^{out}, but b_{t_3} is not in B_{t+1}^{out} because it belongs to $\mathcal{X}_t^+$: at most $B_t^{out} - |\mathcal{X}_t^+|$ containers are in the first subset (1 here). Containers b_{t_1} and b_{t_2} do not contribute to B_t^{out}, but only b_{t_1} is in B_{t+1}^{out} because it belongs to $\mathcal{X}_t^-$: at most $|\mathcal{X}_t^-|$ containers are in the second subset (1 here).

Lemma 1. *The maximum number of relocations R_L that the leveling heuristic L performs is limited by $R_L \le 2 \cdot h(\mathcal{D}_1) + B_1^{out} - 2B$.*

Proof. By summing the equations given by Lemmata 3–6 over all periods $t = 1, \dots, N$ and taking into account that $y \in \mathcal{B}_t$ occurs exactly B times, we obtain

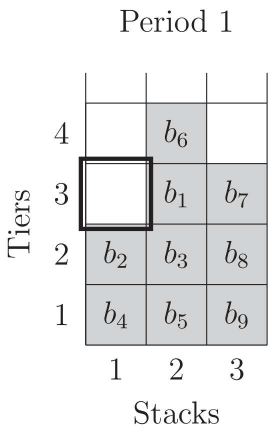
$$\sum_{t=1}^{N} \left(|\mathcal{X}_t^\circ| + |\mathcal{X}_t^-|\right) \le h(\mathcal{D}_1) - h(\mathcal{D}_{N+1}) - B = h(\mathcal{D}_1) - B;$$

$$\sum_{t=1}^{N} \left(|\mathcal{X}_t^\circ| + |\mathcal{X}_t^+|\right) \le h(\mathcal{D}_1) - h(\mathcal{D}_{N+1}) + B_1^{out} - B_{N+1}^{out} - B$$
$$= h(\mathcal{D}_1) + B_1^{out} - B.$$

From Lemma 2 follows that $|\mathcal{X}_t| = |\mathcal{X}_t^-| + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^+|$ for all $t = 1, \dots, N$. Hence,

$$R_L = \sum_{t=1}^{N} |\mathcal{X}_t^-| + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^+|$$
$$\le \sum_{t=1}^{N} \left((|\mathcal{X}_t^\circ| + |\mathcal{X}_t^-|) + (|\mathcal{X}_t^\circ| + |\mathcal{X}_t^+|)\right)$$
$$\le h(\mathcal{D}_1) - B + h(\mathcal{D}_1) + B_1^{out} - B = 2 \cdot h(\mathcal{D}_1) + B_1^{out} - 2B$$

□



$$N = 9,\ W = 3$$
$$\mathcal{B}_1 = \{b_6\},\ B_1^{out} = 1$$
$$\mathcal{D}_1 = \{(1,3)\}$$
$$R^* = 1,\ R_L = 5 \rightarrow \frac{R_L}{R^*} = 5$$
$$c_L(2,4,3) = \frac{2 \cdot h(\mathcal{D}_1) + B_1^{out} - 2B}{B} = \frac{2 \cdot 3 + 1 - 2 \cdot 1}{1} = 5$$
$$c_L = 2 \cdot \left\lceil \frac{N}{W} \right\rceil - 1 = 2 \cdot \left\lceil \frac{9}{3} \right\rceil - 1 = 5$$

Fig. 6. Illustration of the computation of $c_L(s_1, \ldots, s_W)$ and example where $c_L = 2 \cdot \lceil \frac{N}{W} \rceil - 1$ is tight.

Theorem 1. *The competitiveness ratio c_L of the leveling heuristic L is $2\lceil \frac{N}{W} \rceil - 1$ for a bay with W stacks containing N containers.*

Proof. The optimal offline solution performs at least $R^* \geq B$ relocations (Property 4). The leveling heuristic L performs at most $R_L \leq 2 \cdot h(\mathcal{D}_1) + B_1^{out} - 2B$ relocations (Lemma 1). (Remember that $B = |\mathcal{B}|$). Since by definition $B_1^{out} \leq B$,

$$\frac{R_L}{R^*} \leq \frac{2 \cdot h(\mathcal{D}_1) + B_1^{out} - 2B}{B} \leq 2 \cdot \frac{h(\mathcal{D}_1)}{B} - 1$$

We now show that the height h of each position in $\mathcal{D}_1$ satisfies $h \leq \lceil \frac{N}{W} \rceil$. Let us introduce N_{below} the number of containers initially positioned at a height not greater than $\lceil \frac{N}{W} \rceil$, and N_{above} the containers located above. Let us introduce B_{below} and B_{above} the same measures counting only containers in $\mathcal{B}$. Let us finally define E_{below} the number of empty positions under the same threshold $\lceil \frac{N}{W} \rceil$ (included). By definition, $N_{below} + E_{below} = W \times \lceil \frac{N}{W} \rceil$, therefore $N_{below} + E_{below} \geq N$. It implies $E_{below} \geq N - N_{below} = N_{above} \geq B_{above}$. Given that $B_{below} + B_{above} = B$, it shows $B_{below} + E_{below} \geq B$. Seeing the construction procedure of set $\mathcal{D}_1$, it proves that the B positions that constitute $\mathcal{D}_1$ will be found before exceeding tier $\lceil \frac{N}{W} \rceil$. Finally:

$$\frac{R_L}{R^*} \leq 2 \cdot \frac{h(\mathcal{D}_1)}{B} - 1 = 2 \cdot \frac{\sum_{p \in \mathcal{D}_1} h(p)}{B} - 1 \leq 2 \cdot \left\lceil \frac{N}{W} \right\rceil - 1$$

□

The proof permits to derive a more precise competitiveness ratio when the detailed layout of the bay is known. Denoting $s_1, \ldots, s_W$ the initial number of containers in the stacks, we note $c_L(s_1, \ldots, s_W)$ this ratio. $c_L(s_1, \ldots, s_W)$ is determined by computing the maximum value of $\frac{2 \cdot h(\mathcal{D}_1) + B_1^{out} - 2B}{B}$ for the given layout. For a given number of blocking containers B, the maximum value is obtained when these blocking containers are as high as possible. Indeed, it is easy to see that if a non-blocking container is higher than a blocking container, exchanging these two containers cannot decrease $2 \cdot h(\mathcal{D}_1) + B^{out}$. Furthermore, the maximal value for B is given by N minus the number of non-empty stacks: at most all containers are blocking except those at the bottom of the stacks. Hence, $c_L(s_1, \ldots, s_W)$ is easily computed as follows: the worst-case (maximal) value of the numerator is evaluated for each value of B, and the maximal ratio is kept.

Corollary 1. $c_L(s_1, \ldots, s_W)$*, defined as the maximum value of* $\frac{2 \cdot h(\mathcal{D}_1) + B_1^{out} - 2B}{B}$*, is a competitiveness ratio of the leveling heuristic L when the layout of the bay is* $(s_1, \ldots, s_W)$.

Fig. 6 illustrates the computation of the ratio $c_L(2, 4, 3)$, for layout (2, 4, 3). For $B = 1$, the maximal ratio is attained when the blocking container is at the top of the second stack, as in the retrieval sequence depicted on the figure. Then, $\mathcal{D}_1 = \{(1,3)\}$, $h(\mathcal{D}_1) = 3$ and $B_1^{out} = 1$. Hence, $\frac{2 \cdot h(\mathcal{D}_1) + B_1^{out} - 2B}{B} = \frac{2 \cdot 3 + 1 - 2 \cdot 1}{1} = 5$. It is easy to see that ratios obtained for larger values of B are smaller, hence $c_L(2, 4, 3) = 5$.

Corollary 2. *The competitiveness ratio $c_L = 2\lceil \frac{N}{W} \rceil - 1$ for a bay with W stacks and N containers is tight.*

Fig. 6 also shows that the ratio presented in Corollary 2 is tight. Containers have to be retrieved in ascending order from b_1 to b_9. When retrieving container b_1, container b_6 has to be relocated. In the optimal solution container b_6 is relocated to stack 3 on top of container b_7. Containers b_2 to b_9 may then be retrieved without further relocations. Hence, $R^* = 1$. The leveling heuristic L relocates container b_6 to stack 1 on top of container b_2. When retrieving container b_2 container b_6 has to be relocated again and is put back to stack 2 on top of container b_3. Further relocations occur when retrieving containers b_3 (from stack 2 to 1), b_4 (from stack 1 to 2) and b_5 (from stack 2 to 1). Containers b_6 to b_9 may then be retrieved without further relocations. Hence, $R_L = 5$. Consequently, $\frac{R_L}{R^*} = 5 = 2 \cdot \lceil \frac{N}{W} \rceil - 1$.

5. Computational results

This section presents computational results for the leveling heuristic L. Section 5.1 extensively analyzes the average and worst-case performances of heuristic L on small-size layouts. In this first series of experiments, heuristic L is also compared with two other greedy relocation strategies. Section 5.2 evaluates heuristic L on a set of benchmark instances.

5.1. Performance of leveling heuristic L

We evaluate the performance of heuristic L via the average and the maximum gap between the results obtained with heuristic L and the optimal offline solution. We compare heuristic L with two other greedy heuristics. Heuristic R relocates a container to a randomly chosen stack which is different from the retrieval stack. Heuristic M relocates a container to its right neighbor stack (from stack $w = 1, \ldots, W - 1$ to $w + 1$ and from stack W to 1). To analyze the performance of heuristics L, R and M, we run experiments on different layouts with $W = 3$ and $N = 9$. Each instance set is described by a tuple (s_1, s_2, s_3) that indicates the number of containers per stack in the initial layout. For each tuple, experiments are run on all 362 880 (9!) permutations of container positions. We run experiments on 7 different bay configurations where containers are distributed in different patterns among the stacks.

Table 1
Performance of heuristics R, M and L for different layouts with $W = 3$ and $N = 9$.

Inst. set	(1,3,5)	(2,2,5)	(2,4,3)	(3,3,3)	(4,1,4)	(5,4,0)	(9,0,0)
Avg. B	3.88	3.71	3.58	3.5	3.83	4.63	6.17
Avg. R^*	5.40	5.25	5.04	4.92	5.32	5.77	7.44
Avg. R_R	7.79	7.34	7.07	6.72	7.31	7.50	10.0
Avg. Δ_R	1.46	1.40	1.40	1.36	1.37	1.28	1.34
Max Δ_R	4.00	**4.00**	5.00	4.00	**3.00**	2.80	2.87
Avg. R_M	7.99	7.83	7.46	7.23	7.92	8.99	11.9
Avg. Δ_M	1.46	1.47	1.45	1.44	1.46	1.54	1.58
Max Δ_M	4.00	**4.00**	**3.83**	**3.42**	4.20	4.66	4.14
Avg. R_L	6.40	6.28	5.93	5.79	6.21	6.86	9.38
Avg. Δ_L	**1.18**	**1.19**	**1.17**	**1.17**	**1.16**	**1.18**	**1.25**
Max Δ_L	**3.33**	**4.00**	5.00	4.00	**3.00**	**2.75**	**2.42**
$c_L(s_1, s_2, s_3)$	4.00	5.00	5.00	4.00	4.00	3.40	3.14

Table 2
Performance of heuristics R, M and L on the instances of Caserta et al. (2011).

W-S	N	Avg. BKS	Avg. R_R	Avg. Δ_R	Avg. R_M	Avg. Δ_M	Avg. R_L	Avg. Δ_L
3-3	9	5.00	6.78	1.33	7.35	1.45	6.18	1.21
3-4	12	6.18	8.95	1.43	9.90	1.60	7.43	1.20
3-5	15	7.03	11.38	1.64	13.23	1.86	8.68	1.24
3-6	18	8.40	13.70	1.59	15.10	1.79	10.58	1.26
3-7	21	9.28	15.53	1.67	16.53	1.77	11.75	1.27
3-8	24	10.65	18.78	1.77	20.53	1.92	13.03	1.22
4-4	16	10.20	16.70	1.63	18.73	1.83	13.75	1.34
4-5	20	12.95	20.73	1.59	25.45	1.98	17.93	1.38
4-6	24	14.03	25.00	1.79	28.55	2.04	18.98	1.35
4-7	28	16.13	29.23	1.81	34.63	2.15	22.38	1.40
5-4	20	15.43	24.75	1.59	30.40	1.96	21.88	1.43
5-5	25	18.85	35.33	1.88	40.93	2.17	27.98	1.48
5-6	30	22.08	42.48	1.92	51.70	2.33	33.13	1.50
5-7	35	24.30	47.55	1.95	61.43	2.53	37.53	1.54
5-8	40	27.85	55.25	1.99	71.43	2.57	42.60	1.53
5-9	45	30.68	62.40	2.04	76.73	2.51	47.48	1.55
5-10	50	33.63	69.75	2.08	84.80	2.53	52.68	1.57
6-6	36	31.08	61.95	2.00	76.90	2.47	48.70	1.58
6-10	60	47.20	100.83	2.14	126.23	2.68	78.35	1.66
10-6	60	84.98	170.58	2.02	187.05	2.22	149.38	1.77
10-10	100	126.33	290.35	2.30	316.45	2.51	230.28	1.83

Table 1 displays the average results for each instance set. It indicates the average number of blocking containers (B) and the average number of relocations for the optimal offline solution (R^*). The offline solution was computed with CPLEX 12.5 using the formulation presented in Zehendner et al. (2015). The solution time per instance was less than 1 second. It also reports the results for heuristics R (random), M (right neighbor stack) and L (leveling). For each heuristic $h \in \{R, M, L\}$, the average and the maximum number of executed relocations (R_h) and the average and maximum gap with regard to the optimal solution ($\Delta_h = R_h/R^*$) are indicated. The smallest average and maximum gaps for each instance set (best values for Avg Δ_h and Max Δ_h in each column) are marked in bold. For heuristic L, the competitiveness ratio $c_L(s_1, s_2, s_3)$ is also reported. Note that ratio $c_L = 2 \cdot \lceil \frac{N}{W} \rceil - 1 = 5$ is constant for all layouts.

For all instance sets, the average gap of heuristic L is 10 to 35% percentage points smaller than that of heuristics R and M. For unleveled layouts (1,3,5), (2,2,5), (4,1,4), (5,4,0) and (9,0,0), heuristic L obtains the smallest maximum gap. For leveled layouts ((2,4,3) and (3,3,3)) heuristic M obtains the smallest maximum gap. Competitiveness ratio c_L is reached for instance set (2, 4, 3). For other layouts, c_L overestimates the maximum gap. Competitiveness ratio $c_L(s_1, s_2, s_3)$ gives a better estimate in general. The ratio is reached both for instance sets (2, 4, 3) and (3, 3, 3).

5.2. Computational experiments on benchmark instances

We run experiments on the instances introduced by Caserta et al. (2011) that are commonly used to compare different solution methods for the offline container relocation problem. Each instance set $W - S$ is defined by the number of stacks W and the number of containers per stack S (the same for all stacks). Each instance gives the initial positions of N ($N = W \cdot S$) containers. The two topmost positions of every stack are empty ($H = S + 2$). They provide 21 instance sets $W - S$ with 40 instances per set.

Table 2 reports the average of the best known solution (BKS) obtained by Zhang et al. (2010) and the results of the random heuristic R, the right neighbor stack heuristic M and the leveling heuristic L. It displays the average number of relocations (R_h) and the average gap towards the BKS offline solution (Δ_h) for $h \in \{R, M, L\}$. Computation times for all heuristics are $<$ 0.01s for each instance with all its relocations.

Heuristic L outperforms the random heuristic R and the right neighbor stack heuristic M on all instance sets. However, heuristic L performs on average 20% (for small instances) to 85% (for big instances) more relocations than the best known solution for the case with complete information on the retrieval sequence. The maximum gap Δ_L is 2.2 for an instance of set 10–6.

6. Conclusion and outlook

In this paper, we introduced the Online Container Relocation Problem with incomplete information where the knowledge of the retrieval order is limited to a given look-ahead horizon. This problem is a variant of the standard CRP where complete information is available. We focused on the case with no advance information, that we coined as OCRP_0. We analyzed the worst case and average performance of the leveling heuristic L that relocates containers to the lowest empty position (giving priority to the leftmost position in the same tier). The theoretical performance analysis of heuristic L showed that it guarantees a competitiveness ratio that is equal to $2\lceil \frac{N}{W} \rceil - 1$ for a bay with W stacks and N containers. Computational experiments complete the paper and give some insights on the actual performance of heuristic L.

Perspectives for future work are to analyze the theoretical performance of heuristic L and others for larger look-ahead horizons. Further analysis on the value of information and the impact of increasing look-ahead horizons would be interesting. Finally, several variants of the problem deserve being investigated: problems including storage operations (incoming containers), problems with relocations not limited to containers above the target container, problems where the retrieval order is defined for groups of containers, problems where some freedom exists to change retrieval orders...All these issues are left for future works. Further research could also tackle the problem of defining a lower bound on the competitive ratio of any heuristic for the OCRP.

Appendix A. Maximum number of relocations of heuristic *L*

The appendix presents the proofs for Lemmata 2–6. We first recall and extend the notation introduced in Section 4.2.

- p: a position in the bay, defined by stack w and tier h
- $h(p)$: the height of position p; $w(p)$: stack of position p
- y_t: the container to be retrieved in period t; p_t^y: position of container y_t at period t (notation y_t and p_t^y are used instead of b_t and $p_t^{y_t}$ for the sake of a better readability).
- $\mathcal{B}_t$: the set of potentially blocking containers at period t
- $\mathcal{D}_t$: the set of $|\mathcal{B}_t|$ qualifying positions at period t
- $h(\mathcal{D}_t)$: the sum of the heights of the positions in $\mathcal{D}_t$
- $h_{\mathcal{D}_t}^{\max}$: the maximum height of a position in $\mathcal{D}_t$
- $\mathcal{B}_t^{out}$: the set of blocking containers not located in positions belonging to $\mathcal{D}_t$ at period t, and B_t^{out} is its cardinality
- $\mathcal{X}_t \subseteq \mathcal{B}_t$: the set of containers to be relocated at period t (does not include the retrieval container y_t)
- $\mathcal{X}_t^{\circ}$: the set of containers relocated from $p_t^b \in \mathcal{D}_t$ to $p_{t+1}^b \in \mathcal{D}_{t+1}$ at period t
- $\mathcal{X}_t^{+}$: the set of containers relocated from $p_t^b \notin \mathcal{D}_t$ to $p_{t+1}^b \in \mathcal{D}_{t+1}$ at period t
- $\mathcal{X}_t^{-}$: the set of containers relocated from $p_t^b \in \mathcal{D}_t$ to $p_{t+1}^b \notin \mathcal{D}_{t+1}$ at period t
- $\mathcal{D}_{N+1} = \mathcal{B}_{N+1} = \emptyset$, $B_{N+1}^{out} = 0$

For all the subsequent results, t is a given period with $1 \le t \le N$.

We start by demonstrating some properties. From the definition of $\mathcal{X}_t^{\circ}$ and $\mathcal{X}_t^{-}$, it follows that each container $b \in \mathcal{X}_t^{\circ} \cup \mathcal{X}_t^{-}$ is relocated at period t out of a position $p_t^b \in \mathcal{D}_t$ above container y_t and with $h(p_t^b) \le h_{\mathcal{D}_t}^{\max}$. Properties 5 and 6 trivially result from this definition. They are used within the following proofs.

Property 5. *If* $|\mathcal{X}_t^{\circ}| + |\mathcal{X}_t^{-}| > 0$, $h(p_t^y) + |\mathcal{X}_t^{\circ}| + |\mathcal{X}_t^{-}| \le h_{\mathcal{D}_t}^{\max}$.

Property 6. *If* $h(p_t^y) \ge h_{\mathcal{D}_t}^{\max}$, $|\mathcal{X}_t^{\circ}| + |\mathcal{X}_t^{-}| = 0$.

Also, it is important to highlight the following property of sets $\mathcal{D}_t$. Because these sets are composed of positions either empty or containing blocking containers, and because no containers other than blocking containers can be relocated, the set of positions that are candidates to belong to $\mathcal{D}_t$ only increases (in the sense of inclusion) with t: a position that is empty or that contains a blocking container will always either be empty or contain a blocking container at further iterations. Indeed, relocated (blocking) containers move to empty positions and leave positions empty. Furthermore, from iteration t to $t+1$, the only new position that could become candidate to enter $\mathcal{D}_{t+1}$ is p_t^y, i.e., the position left empty by the retrieval container. The following property relies on this observation.

Property 7. *The difference between sets $\mathcal{D}_t$ and $\mathcal{D}_{t+1}$ falls into one of the following cases:*

1. *If $y_t \notin \mathcal{B}$ and $h(p_t^y) < h_{\mathcal{D}_t}^{\max}$, $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{p_t^y\} \setminus \{p\}$ where p is a position in $\mathcal{D}_t$ with $h(p) = h_{\mathcal{D}_t}^{\max}$.*
2. *If $y_t \notin \mathcal{B}$ and $h(p_t^y) > h_{\mathcal{D}_t}^{\max}$, $\mathcal{D}_{t+1} = \mathcal{D}_t$.*
3. *If $y_t \notin \mathcal{B}$ and $h(p_t^y) = h_{\mathcal{D}_t}^{\max}$, either $\mathcal{D}_{t+1} = \mathcal{D}_t$ or $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{p_t^y\} \setminus \{p\}$ where p is a position in $\mathcal{D}_t$ with $h(p) = h_{\mathcal{D}_t}^{\max}$.*
4. *If $y_t \in \mathcal{B}$, $\mathcal{D}_{t+1} = \mathcal{D}_t \setminus \{p\}$ where p is a position in $\mathcal{D}_t$ with $h(p) = h_{\mathcal{D}_t}^{\max}$.*

Proof. Let us start with the cases 1 to 3 where $y_t \notin \mathcal{B}$. If $y_t \notin \mathcal{B}$, $\mathcal{B}_{t+1} = \mathcal{B}_t$ and therefore $|\mathcal{D}_{t+1}| = |\mathcal{D}_t|$. In this case, because of the priority given to lower tiers in the construction of qualifying sets, position p_t^y enters $\mathcal{D}_{t+1}$ when $h(p_t^y) < h_{\mathcal{D}_t}^{\max}$ (case 1), it does not enter $\mathcal{D}_{t+1}$ when $h(p_t^y) > h_{\mathcal{D}_t}^{\max}$ (case 2) and it might or might not enter $\mathcal{D}_{t+1}$ when $h(p_t^y) = h_{\mathcal{D}_t}^{\max}$ (case 3). When p_t^y enters $\mathcal{D}_{t+1}$ (cases 1 and 3), one position of $\mathcal{D}_t$ at tier $h_{\mathcal{D}_t}^{\max}$ is not maintained in $\mathcal{D}_{t+1}$. All other positions in $\mathcal{D}_t$ are also included in $\mathcal{D}_{t+1}$. While this last claim is clear for positions at tier $h < h_{\mathcal{D}_t}^{\max}$, it requires some explanation for positions at tier $h = h_{\mathcal{D}_t}^{\max}$. Indeed, it is possible that the state of some positions change from occupied by a blocking container to empty, or the opposite. However: (i) a higher priority is given to positions already containing blocking containers at period t, (ii) the positions of the relocated containers have a higher priority than empty positions and the leftmost-priority policy for heuristic L is consistent with the policy used when constructing qualifying sets. For these two reasons, the claim holds. When p_t^y does not enter $\mathcal{D}_{t+1}$ (cases 2 and 3), the sets $\mathcal{D}_t$ and $\mathcal{D}_{t+1}$ are identical.

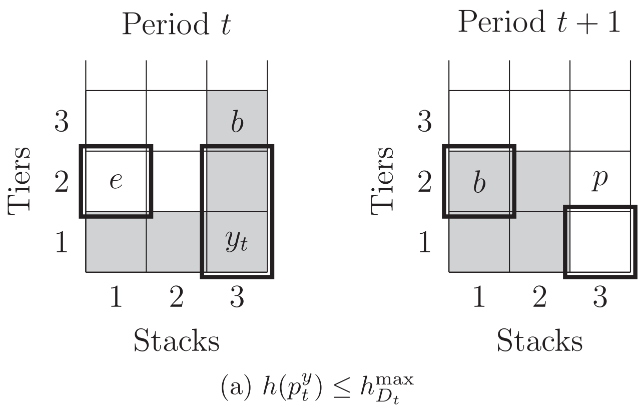
When $y_t \in \mathcal{B}$ (case 4), $\mathcal{B}_{t+1} = \mathcal{B}_t \setminus \{y_t\}$ and therefore $|\mathcal{D}_{t+1}| = |\mathcal{D}_t| - 1$. Furthermore, the set of candidate positions are identical for $\mathcal{D}_{t+1}$ and $\mathcal{D}_t$: p_t^y contains a blocking container at period t and is thus already candidate for $\mathcal{D}_t$. Following the arguments developed above, all positions in $\mathcal{D}_t$ at tiers $h < h_{\mathcal{D}_t}^{\max}$ are included in $\mathcal{D}_{t+1}$ and all positions except one that were in $\mathcal{D}_t$ with $h(p) = h_{\mathcal{D}_t}^{\max}$ are also in $\mathcal{D}_{t+1}$. □

We now prove Lemmata 2–6.

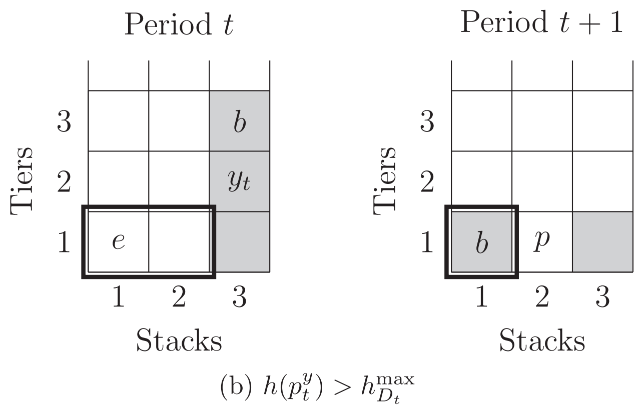
Lemma 2. $\mathcal{X}_t = \mathcal{X}_t^{+} \cup \mathcal{X}_t^{\circ} \cup \mathcal{X}_t^{-}$.

Proof. We prove that the leveling heuristic L never relocates a container $b \in \mathcal{B}_t$ from position $p_t^b \notin \mathcal{D}_t$ to position $p_{t+1}^b \notin \mathcal{D}_{t+1}$. The proof is illustrated with Fig. 7. Consider a container $b \in \mathcal{B}_t$ relocated from position $p_t^b \notin \mathcal{D}_t$ at period t. From the definition of $\mathcal{D}_t$ (remember that $|\mathcal{D}_t| = |\mathcal{B}_t|$), it follows that it is relocated to an empty position $e \in \mathcal{D}_t$. We show that $e \in \mathcal{D}_{t+1}$ is always true. Let us assume $e \notin \mathcal{D}_{t+1}$.

Property 7 states that either $\mathcal{D}_{t+1} = \mathcal{D}_t$, $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{p_t^y\} \setminus \{p\}$ or $\mathcal{D}_{t+1} = \mathcal{D}_t \setminus \{p\}$, where p is a position in $\mathcal{D}_t$ with $h(p) = h_{\mathcal{D}_t}^{\max}$. Hypothesis $e \notin \mathcal{D}_{t+1}$ implies $\mathcal{D}_{t+1} \neq \mathcal{D}_t$ and $e = p$. Furthermore, because $b \in \mathcal{B}$ and $p_t^b \notin \mathcal{D}_t$, $h(p_t^b) \ge h_{\mathcal{D}_t}^{\max}$. Let us now consider two cases:



(a) $h(p_t^y) \le h_{D_t}^{\max}$



(b) $h(p_t^y) > h_{D_t}^{\max}$

Fig. 7. Illustration of Lemma 2.

- $h(p_t^y) \le h_{D_t}^{\max}$: Position $(w(p_t^y), h_{D_t}^{\max})$ contains a container at period t – which can be y_t, b or any container located between y_t and b – and is empty at period $t+1$. Furthermore, seeing that Algorithm 1 gives priority to positions containing blocking containers and seeing that e is empty at period t, that $e \in \mathcal{D}_t$ and that $h(e) = h(p) = h_{D_t}^{\max}$, it implies $(w(p_t^y), h_{D_t}^{\max}) \in \mathcal{D}_t$. Equivalently, it is not possible to have $e \notin \mathcal{D}_{t+1}$ and $(w(p_t^y), h_{D_t}^{\max}) \in \mathcal{D}_{t+1}$. Finally, it is not possible to have $e \notin \mathcal{D}_{t+1}$ and $(w(p_t^b), h_{D_t}^{\max}) \notin \mathcal{D}_{t+1}$ unless $e = (w(p_t^b), h_{D_t}^{\max})$, which is absurd and permits to conclude.
- $h(p_t^y) > h_{D_t}^{\max}$: In this case, $y_t \in \mathcal{B}$ and $\mathcal{D}_{t+1} = \mathcal{D}_t \setminus \{p\}$. At period t, $\mathcal{D}_t$ contains as many empty positions as the number of blocking containers located outside $\mathcal{D}_t$, that is, at least the number of relocated containers (they are all above y_t and outside $\mathcal{D}_t$) plus one (for container y_t). Once the containers relocated, it remains at least one empty position in $\mathcal{D}_t$ at period $t+1$. Hence, the position p excluded from $\mathcal{D}_t$ when constructing $\mathcal{D}_{t+1}$ is empty, which is impossible and permits to conclude.

□

Lemma 3. *If container $y_t \notin \mathcal{B}_t$, the following inequality holds: $h(\mathcal{D}_{t+1}) \le h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^-|$.*

Proof. Following Property 7, we distinguish three cases:

1. If $h(p_t^y) < h_{D_t}^{\max}$, p_t^y enters $\mathcal{D}_{t+1}$ and a position $p \in \mathcal{D}_t$ with $h(p) = h_{D_t}^{\max}$ exits (Fig. 8a): $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{p_t^y\} \setminus \{p\}$.
   $\rightarrow h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t) + h(p_t^y) - h_{D_t}^{\max}$
2. If $h(p_t^y) > h_{D_t}^{\max}$, p_t^y does not enter $\mathcal{D}_{t+1}$ and $\mathcal{D}_{t+1} = \mathcal{D}_t$ (Fig. 8b).
   $\rightarrow h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t)$
3. If $h(p_t^y) = h_{D_t}^{\max}$, p_t^y may or may not enter $\mathcal{D}_{t+1}$ depending on the bay configuration (Fig. 8c and d): either $\mathcal{D}_{t+1} = \mathcal{D}_t$ or there exists a position p with $h(p) = h_{D_t}^{\max}$ such that $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{p_t^y\} \setminus \{p\}$.
   $\rightarrow h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t)$ in both cases

The following equation summarizes the three cases:

$$h(\mathcal{D}_{t+1}) = \begin{cases} h(\mathcal{D}_t) + h(p_t^y) - h_{D_t}^{\max} & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ h(\mathcal{D}_t) & \text{if } h(p_t^y) \ge h_{D_t}^{\max} \end{cases}$$

Since either $|\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = 0$ or by Property 5, if $h(p_t^y) < h_{D_t}^{\max}$, $h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| \le h_{D_t}^{\max}$.Then:

$$h(\mathcal{D}_{t+1}) \le \begin{cases} h(\mathcal{D}_t) + h(p_t^y) - (h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-|) & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ h(\mathcal{D}_t) & \text{if } h(p_t^y) \ge h_{D_t}^{\max} \end{cases}$$

and equivalently,

$$h(\mathcal{D}_{t+1}) \le \begin{cases} h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^-| & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ h(\mathcal{D}_t) & \text{if } h(p_t^y) \ge h_{D_t}^{\max} \end{cases} \quad (4)$$

Using Property 6, it follows that $h(\mathcal{D}_{t+1}) \le h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^-|$. □

Lemma 4. *If container $y_t \in \mathcal{B}_t$, the following inequality holds: $h(\mathcal{D}_{t+1}) \le h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^-| - 1$.*

Proof. From Property 7, a position p with $h(p) = h_{D_t}^{\max}$ exits $\mathcal{D}_t$: $\mathcal{D}_{t+1} = \mathcal{D}_t \setminus \{p\}$ (note that p_t^y might or not belong to $\mathcal{D}_t$ or $\mathcal{D}_{t+1}$ - see Fig. 9a–c):
$\rightarrow h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t) - h_{D_t}^{\max}$
Since either $|\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = 0$ or by Property 5, if $h(p_t^y) < h_{D_t}^{\max}$, $h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| \le h_{D_t}^{\max}$.Then:

$$h(\mathcal{D}_{t+1}) \le \begin{cases} h(\mathcal{D}_t) - (h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-|) & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ h(\mathcal{D}_t) - h_{D_t}^{\max} & \text{if } h(p_t^y) \ge h_{D_t}^{\max} \end{cases}$$

Because $y_t \in \mathcal{B}_t$, $\mathcal{D}_t$ is not empty and $h_{D_t}^{\max} \ge 1$. Also, $h(p_t^y) \ge 1$. Thus:

$$h(\mathcal{D}_{t+1}) \le \begin{cases} h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^-| - 1 & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ h(\mathcal{D}_t) - 1 & \text{if } h(p_t^y) \ge h_{D_t}^{\max} \end{cases} \quad (A.5)$$

Because of Property 6, it shows $h(\mathcal{D}_{t+1}) \le h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^-| - 1$. □

Lemma 5. *If container $y_t \notin \mathcal{B}_t$, the following inequality holds: $h(\mathcal{D}_{t+1}) + B_{t+1}^{out} \le h(\mathcal{D}_t) + B_t^{out} - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^+|$.*

Proof. The proof consists of three steps:

1. Relation between $h(\mathcal{D}_{t+1})$ and $h(\mathcal{D}_t)$

We first refine the relation (4) established in the proof of Lemma 3. We consider the case when $h(p_t^y) < h_{D_t}^{\max}$. We showed in the proof of Lemma 3 that $h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t) + h(p_t^y) - h_{D_t}^{\max}$ and that $h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| \le h_{D_t}^{\max}$. Then, if $h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| < h_{D_t}^{\max}$, $h(\mathcal{D}_{t+1}) \le h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^-| - 1$.

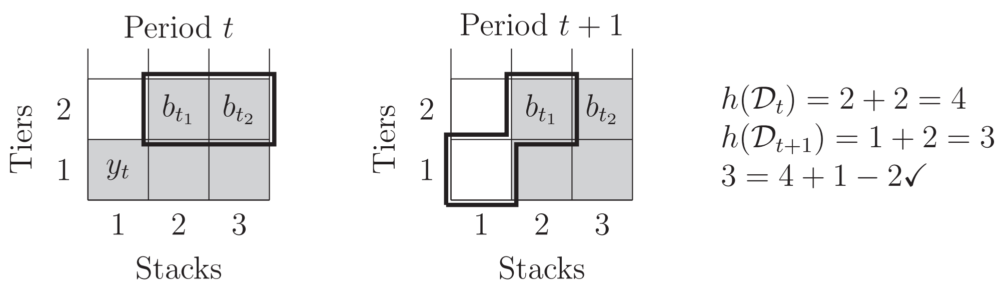
It permits to develop (4) and obtain:

$$h(\mathcal{D}_{t+1}) \le \begin{cases} h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^-| - 1 & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ & \text{and } h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| < h_{D_t}^{\max} \\ h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^-| & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ & \text{and } h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = h_{D_t}^{\max} \\ h(\mathcal{D}_t) & \text{if } h(p_t^y) \ge h_{D_t}^{\max} \end{cases}$$

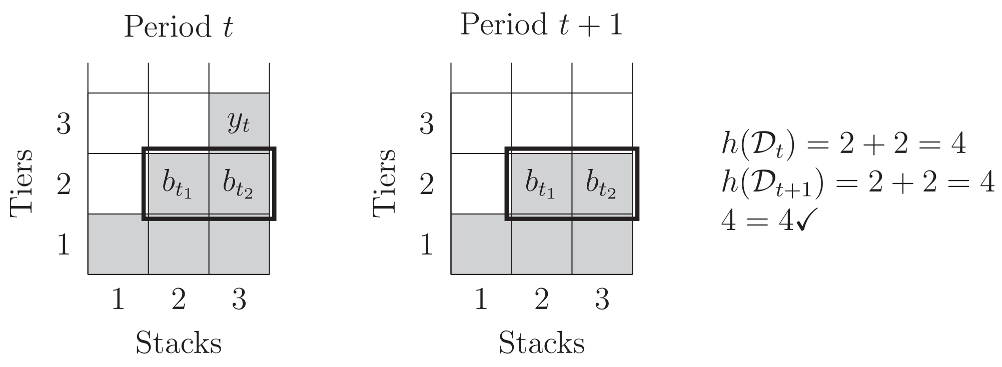
2. Relation between B_{t+1}^{out} and B_t^{out}

Let us recall that $\mathcal{B}_t^{out}$ is the set of blocking containers $b \in \mathcal{B}_t$ that are located at positions $p_t^b \notin \mathcal{D}_t$ at period t, and that B_t^{out} is its cardinality. We partition set $\mathcal{B}_{t+1}$ $(= \mathcal{B}_t)$ in two subsets:

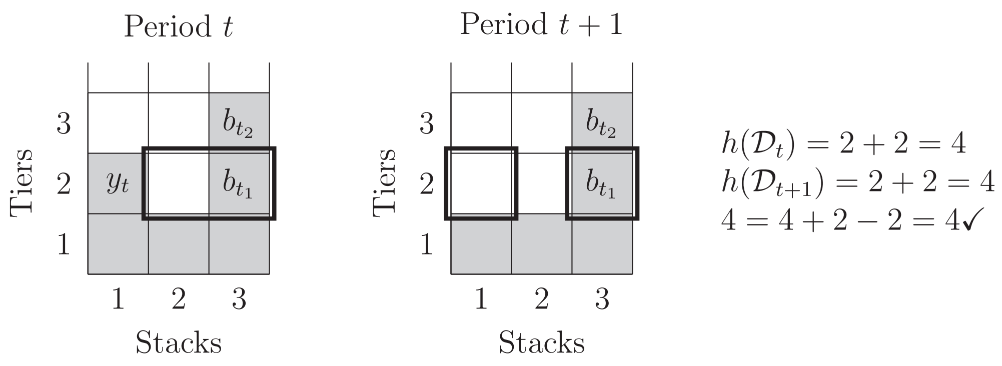
- $\mathcal{B}_t^{out}$
- $\mathcal{B}_t \setminus \mathcal{B}_t^{out}$.



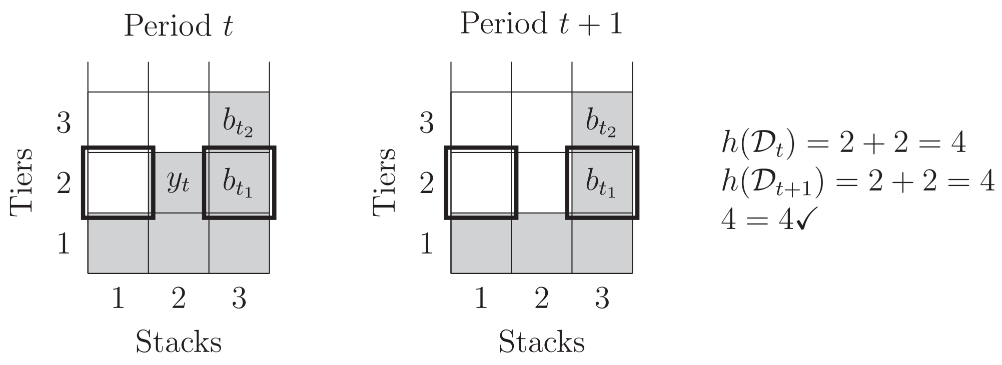
(a) $h(p_t^y) < h_{D_t}^{\max}$: $h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t) + h(p_t^y) - h_{D_t}^{\max}$


Period t
Period $t+1$
Tiers
Stacks
y_t
b_{t_1}
b_{t_2}
$h(\mathcal{D}_t) = 2 + 2 = 4$
$h(\mathcal{D}_{t+1}) = 2 + 2 = 4$
$4 = 4$✓

(b) $h(p_t^y) > h_{D_t}^{\max}$: $h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t)$

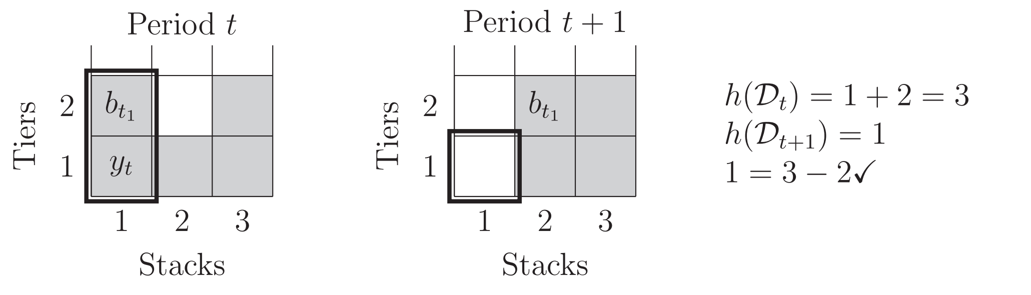


(c) $h(p_t^y) = h_{D_t}^{\max}$: Case 1: $h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t) + h(p_t^y) - h_{D_t}^{\max} = h(\mathcal{D}_t)$

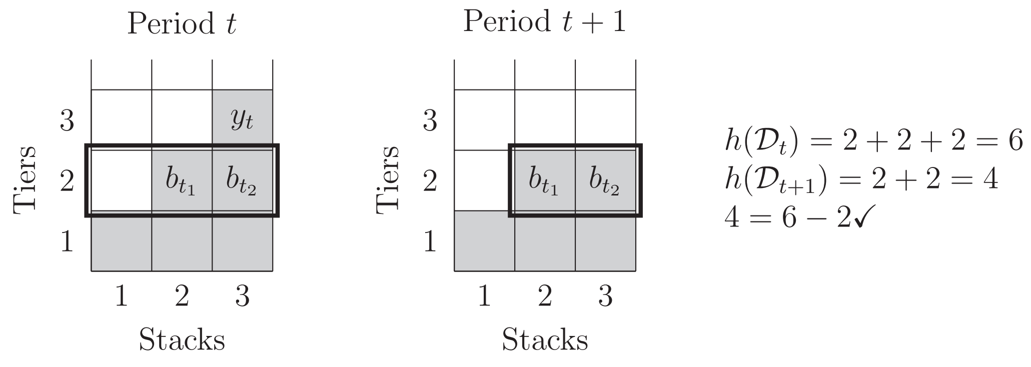

Period t
Period $t+1$
Tiers
Stacks
y_t
b_{t_1}
b_{t_2}
$h(\mathcal{D}_t) = 2 + 2 = 4$
$h(\mathcal{D}_{t+1}) = 2 + 2 = 4$
$4 = 4$✓

(d) $h(p_t^y) = h_{D_t}^{\max}$: Case 2: $h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t)$

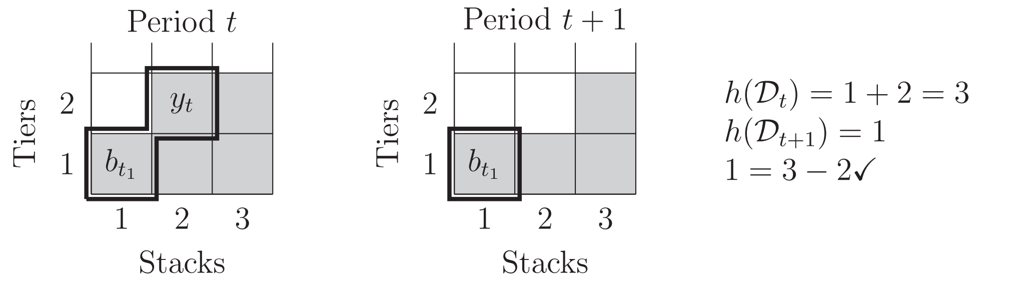
Fig. 8. Impact of the retrieval of $y_t \notin \mathcal{B}_t$ on $h(\mathcal{D}_{t+1})$.



(a) $h(p_t^y) < h_{D_t}^{\max}$: $h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t) - h_{D_t}^{\max}$


Period t
Period $t+1$
Tiers
Stacks
y_t
b_{t_1}
b_{t_2}
$h(\mathcal{D}_t) = 2 + 2 + 2 = 6$
$h(\mathcal{D}_{t+1}) = 2 + 2 = 4$
$4 = 6 - 2$✓

(b) $h(p_t^y) > h_{D_t}^{\max}$: $h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t) - h_{D_t}^{\max}$



(c) $h(p_t^y) = h_{D_t}^{\max}$: $h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t) - h_{D_t}^{\max}$

Fig. 9. Impact of the retrieval of $y_t \in \mathcal{B}_t$ on $h(\mathcal{D}_{t+1})$.

For each subset we evaluate the maximum number of containers that can belong to $\mathcal{B}_{t+1}^{out}$.

At most, all containers in $\mathcal{B}_t^{out}$ belong to $\mathcal{B}_{t+1}^{out}$ except those that are relocated at positions in $\mathcal{D}_{t+1}$, that is, those in $\mathcal{X}_t^+$. The maximal number of containers from $\mathcal{B}_t^{out}$ that belong to $\mathcal{B}_{t+1}^{out}$ is thus $B_t^{out} - |\mathcal{X}_t^+|$.

Let us now evaluate this number for containers in $\mathcal{B}_t \setminus \mathcal{B}_t^{out}$. These containers are in $\mathcal{D}_t$ at period t. Again we split the containers of $\mathcal{B}_t \setminus \mathcal{B}_t^{out}$ in two subsets: those that are relocated at period t, and those that are not. Regarding the first subset, $\mathcal{X}_t^-$ represents exactly those that belong to $\mathcal{B}_{t+1}^{out}$. Their number is $|\mathcal{X}_t^-|$. Regarding those that are not relocated, we consider three cases (Fig. 10):

1. If $h(p_t^y) < h_{D_t}^{\max}$, $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{p_t^y\} \setminus \{p\}$, where p is a position in $\mathcal{D}_t$ (see Property 7). A non-relocated container that would be located in a position in $\mathcal{D}_t$ but not in $\mathcal{D}_{t+1}$, is necessarily located in position p.
   (a) In general, it can thus happen that a non-relocated container belonging to $\mathcal{B}_t \setminus \mathcal{B}_t^{out}$ belongs to $\mathcal{B}_{t+1}^{out}$. It concerns at most 1 container.
   (b) Let us show that in the special case when $h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = h_{D_t}^{\max}$, it cannot happen. Assume that position p contains a blocking container (non-relocated at period t). Position $(w(p_t^y), h_{D_t}^{\max})$ is in $\mathcal{D}_t$ and is empty at period $t+1$. Seeing that Algorithm 1 gives priority to position containing blocking containers, it is not possible to have $p \notin \mathcal{D}_{t+1}$ and $(w(p_t^y), h_{D_t}^{\max}) \in \mathcal{D}_{t+1}$. Furthermore, it is not possible to have $p \notin \mathcal{D}_{t+1}$ and $(w(p_t^y), h_{D_t}^{\max}) \notin \mathcal{D}_{t+1}$ unless $p = (w(p_t^y), h_{D_t}^{\max})$ because only one position is in $\mathcal{D}_t \setminus \mathcal{D}_{t+1}$, which is absurd and permits to conclude.
2. If $h(p_t^y) \geq h_{D_t}^{\max}$, either $\mathcal{D}_{t+1} = \mathcal{D}_t$ or $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{p_t^y\} \setminus \{p\}$, where p is a position in $\mathcal{D}_t$ (see Property 7). In the first case, trivially, no non-relocated container can be in $\mathcal{D}_t \setminus \mathcal{D}_{t+1}$. In the second case, the empty position p_t^y enters $\mathcal{D}_{t+1}$ hence meaning (seeing the priorities of Algorithm 1) that all positions with non-relocated blocking containers at tier $h_{D_t}^{\max}$ also belong to $\mathcal{D}_{t+1}$. Again, no non-relocated container can be in $\mathcal{D}_t \setminus \mathcal{D}_{t+1}$.

Noting that $\mathcal{X}_t^- = \emptyset$ when $h(p_t^y) \geq h_{D_t}^{\max}$, the following equation summarizes the different cases.

$$B_{t+1}^{out} \leq \begin{cases} B_t^{out} - |\mathcal{X}_t^+| + |\mathcal{X}_t^-| + 1 & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ & \quad \text{and } h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| < h_{D_t}^{\max} \\ B_t^{out} - |\mathcal{X}_t^+| + |\mathcal{X}_t^-| & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ & \quad \text{and } h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = h_{D_t}^{\max} \\ B_t^{out} - |\mathcal{X}_t^+| & \text{if } h(p_t^y) \geq h_{D_t}^{\max} \end{cases}$$

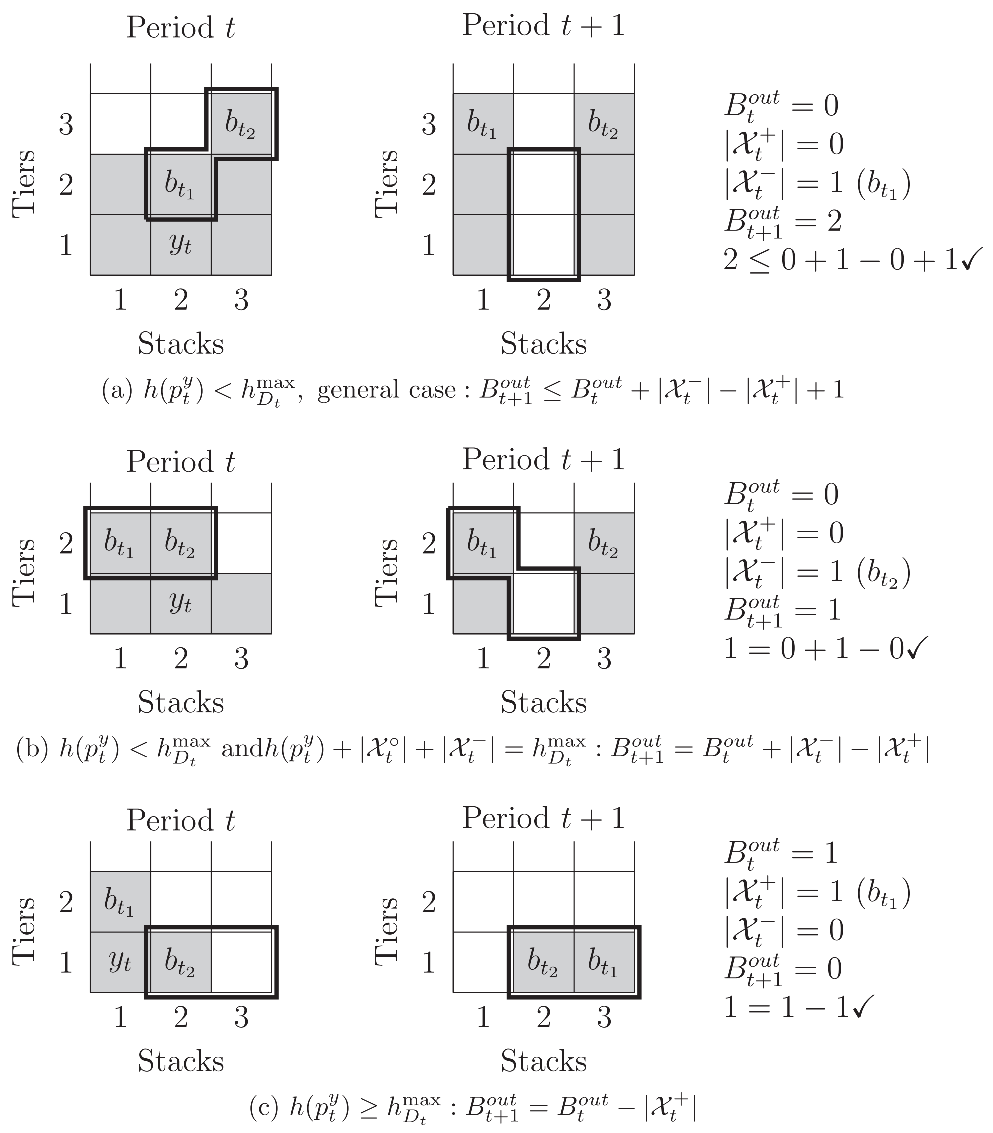

Period t
Period $t+1$
Tiers
Stacks
b_{t_1}
b_{t_2}
y_t
$B_t^{out} = 0$
$|\mathcal{X}_t^+| = 0$
$|\mathcal{X}_t^-| = 1\ (b_{t_1})$
$B_{t+1}^{out} = 2$
$2 \leq 0 + 1 - 0 + 1 ✓$
(a) $h(p_t^y) < h_{D_t}^{\max}$, general case : $B_{t+1}^{out} \leq B_t^{out} + |\mathcal{X}_t^-| - |\mathcal{X}_t^+| + 1$
$B_t^{out} = 0$
$|\mathcal{X}_t^+| = 0$
$|\mathcal{X}_t^-| = 1\ (b_{t_2})$
$B_{t+1}^{out} = 1$
$1 = 0 + 1 - 0 ✓$
(b) $h(p_t^y) < h_{D_t}^{\max}$ and$h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = h_{D_t}^{\max}$: $B_{t+1}^{out} = B_t^{out} + |\mathcal{X}_t^-| - |\mathcal{X}_t^+|$
$B_t^{out} = 1$
$|\mathcal{X}_t^+| = 1\ (b_{t_1})$
$|\mathcal{X}_t^-| = 0$
$B_{t+1}^{out} = 0$
$1 = 1 - 1 ✓$
(c) $h(p_t^y) \geq h_{D_t}^{\max}$: $B_{t+1}^{out} = B_t^{out} - |\mathcal{X}_t^+|$

Fig. 10. Impact of the retrieval of $y_t \notin \mathcal{B}_t$ on B_{t+1}^{out}.

3. Relation between $h(\mathcal{D}_{t+1}) + B_{t+1}^{out}$ and $h(\mathcal{D}_t) + B_t^{out}$
We combine the inequalities on $h(\mathcal{D}_{t+1})$ and B_{t+1}^{out}.

$$h(\mathcal{D}_{t+1}) + B_{t+1}^{out} \leq \begin{cases} h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| + B_t^{out} - |\mathcal{X}_t^+| & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ & \text{and } h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| < h_{D_t}^{\max} \\ h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| + B_t^{out} - |\mathcal{X}_t^+| & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ & \text{and } h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = h_{D_t}^{\max} \\ h(\mathcal{D}_t) + B_t^{out} - |\mathcal{X}_t^+| & \text{if } h(p_t^y) \geq h_{D_t}^{\max} \end{cases}$$

Because of Property 6, it shows: $h(\mathcal{D}_{t+1}) + B_{t+1}^{out} \leq h(\mathcal{D}_t) + B_t^{out} - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^+|$. □

Lemma 6. *If container $y_t \in \mathcal{B}_t$, the following inequality holds: $h(\mathcal{D}_{t+1}) + B_{t+1}^{out} \leq h(\mathcal{D}_t) + B_t^{out} - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^+| - 1$.*

Proof. As before, the proof consists of three steps.

1. Relation between $h(\mathcal{D}_{t+1})$ and $h(\mathcal{D}_t)$
We first refine the relation (A.5) established in the proof of Lemma 4. We consider the case when $h(p_t^y) < h_{D_t}^{\max}$. We showed in the proof of Lemma 4 that $h(\mathcal{D}_{t+1}) = h(\mathcal{D}_t) - h_{D_t}^{\max}$ and that $1 + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| \leq h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| \leq h_{D_t}^{\max}$. Then, if $h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| < h_{D_t}^{\max}$, $h(\mathcal{D}_{t+1}) \leq h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^-| - 1 - 1$.

It permits to develop (A.5) and obtain:

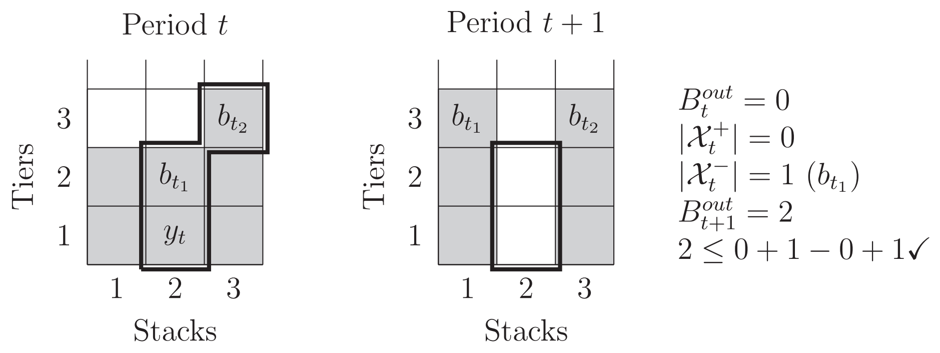
$$h(\mathcal{D}_{t+1}) \leq \begin{cases} h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^-| - 2 & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ & \text{and } h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| < h_{D_t}^{\max} \\ h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^-| - 1 & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ & \text{and } h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = h_{D_t}^{\max} \\ h(\mathcal{D}_t) - 1 & \text{if } h(p_t^y) \geq h_{D_t}^{\max} \end{cases}$$

2. Relation between B_{t+1}^{out} and B_t^{out}
Following the proof of Step 2 of Lemma 5, we partition set $\mathcal{B}_t$ in three subsets:

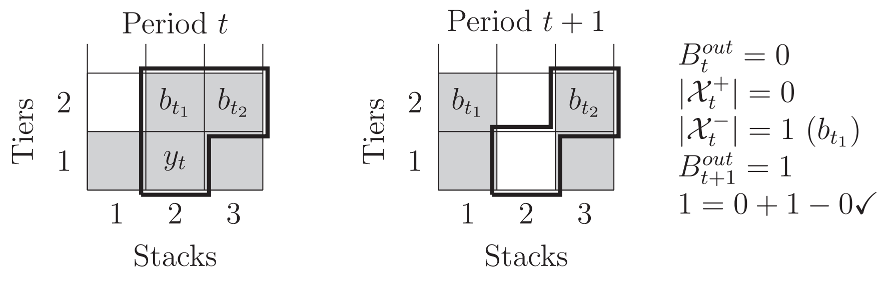
- $\mathcal{B}_t^{out}$
- containers from $\mathcal{B}_t \setminus \mathcal{B}_t^{out}$ relocated at period t
- containers from $\mathcal{B}_t \setminus \mathcal{B}_t^{out}$ not relocated at period t

The maximal number of containers from $\mathcal{B}_t^{out}$ that belong to $\mathcal{B}_{t+1}^{out}$ is $B_t^{out} - |\mathcal{X}_t^+|$ if $y \in \mathcal{D}_t$ and $B_t^{out} - |\mathcal{X}_t^+| - 1$ if $y \notin \mathcal{D}_t$. The number of containers from $\mathcal{B}_t \setminus \mathcal{B}_t^{out}$ relocated at period t is $|\mathcal{X}_t^-|$.

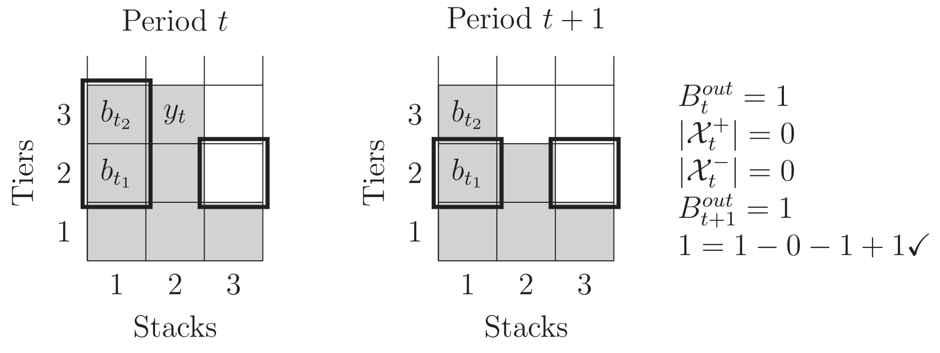
Regarding the third subset, Property 7 states that $\mathcal{D}_{t+1} = \mathcal{D}_t \setminus \{p\}$ with p being a position in $\mathcal{D}_t$ at tier $h_{D_t}^{\max}$. At most one non-relocated container could belong to $\mathcal{D}_t \setminus \mathcal{D}_{t+1}$ (i.e., $\mathcal{B}_{t+1}^{out}$), and it has to be located in position p. Let us precise



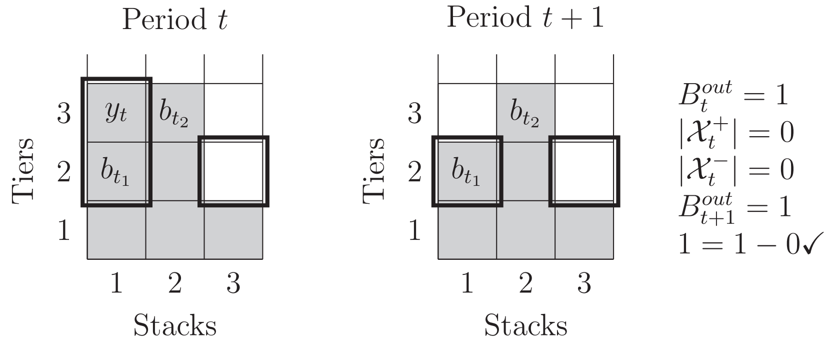
(a) $h(p_t^y) < h_{D_t}^{\max}$, general case: $B_{t+1}^{out} \leq B_t^{out} + |\mathcal{X}_t^-| - |\mathcal{X}_t^+| + 1$



(b) $h(p_t^y) < h_{D_t}^{\max}$ and $h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = h_{D_t}^{\max} : B_{t+1}^{out} = B_t^{out} + |\mathcal{X}_t^-| - |\mathcal{X}_t^+|$



(c) $h(p_t^y) \geq h_{D_t}^{\max}, p_t^y \notin \mathcal{D}_t : B_{t+1}^{out} \leq B_t^{out} - |\mathcal{X}_t^+| - 1 + 1$



(d) $h(p_t^y) \geq h_{D_t}^{\max}, p_t^y \in \mathcal{D}_t : B_{t+1}^{out} \leq B_t^{out} - |\mathcal{X}_t^+|$

Fig. 11. Impact of the retrieval of $y_t \in \mathcal{B}_t$ on B_{t+1}^{out}.

situations where it is not possible that p contains a non-relocated container and is not in $\mathcal{D}_{t+1}$ simultaneously:

1. If $h(p_t^y) < h_{D_t}^{\max}$ and $h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = h_{D_t}^{\max}$: the retrieval of y_t and the associated relocations leave an empty position at tier $h_{D_t}^{\max}$; this empty position is in $\mathcal{D}_t$ at period t; following the proof of Lemma 5, $p \in \mathcal{D}_{t+1}$.
2. If $h(p_t^y) \geq h_{D_t}^{\max}$ and $y_t \in \mathcal{D}_t$: $p \in \mathcal{D}_{t+1}$ for the same reason.

As a conclusion, given that $\mathcal{B}_{t+1} \subset \mathcal{B}_t$, the equation given in Lemma 5 still holds (see also Fig. 11a–d):

$$B_{t+1}^{out} \leq \begin{cases} B_t^{out} - |\mathcal{X}_t^+| + |\mathcal{X}_t^-| + 1 & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ & \text{and } h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| < h_{D_t}^{\max} \\ B_t^{out} - |\mathcal{X}_t^+| + |\mathcal{X}_t^-| & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ & \text{and } h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = h_{D_t}^{\max} \\ B_t^{out} - |\mathcal{X}_t^+| & \text{if } h(p_t^y) \geq h_{D_t}^{\max} \end{cases}$$

3. Relation between $h(\mathcal{D}_{t+1}) + B_{t+1}^{out}$ and $h(\mathcal{D}_t) + B_t^{out}$
We combine the inequalities on $h(\mathcal{D}_{t+1})$ and B_{t+1}^{out}.

$$h(\mathcal{D}_{t+1}) + B_{t+1}^{out} \leq \begin{cases} h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| + B_t^{out} - |\mathcal{X}_t^+| - 1 & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ & \text{and } h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| < h_{D_t}^{\max} \\ h(\mathcal{D}_t) - |\mathcal{X}_t^\circ| + B_t^{out} - |\mathcal{X}_t^+| - 1 & \text{if } h(p_t^y) < h_{D_t}^{\max} \\ & \text{and } h(p_t^y) + |\mathcal{X}_t^\circ| + |\mathcal{X}_t^-| = h_{D_t}^{\max} \\ h(\mathcal{D}_t) + B_t^{out} - |\mathcal{X}_t^+| - 1 & \text{if } h(p_t^y) \geq h_{D_t}^{\max} \end{cases}$$

Because of Property 6, it shows: $h(\mathcal{D}_{t+1}) + B_{t+1}^{out} \leq h(\mathcal{D}_t) + B_t^{out} - |\mathcal{X}_t^\circ| - |\mathcal{X}_t^+| - 1$. □

References

Akyüz, M. H., & Lee, C.-Y. (2014). A mathematical formulation and efficient heuristics for the dynamic container relocation problem. *Naval Research Logistics, 61*, 101–118.

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

Borjian, S., Galle, V., Manshadi, V. H., Barnhart, C., & Jaillet, P. (2015). *Container relocation problem: Approximation, asymptotic, and incomplete information*. arXiv preprint 1505.04229.

Borjian, S., Manshadi, V. H., Barnhart, C., & Jaillet, P. (2015). *Managing relocation and delay in container terminals with flexible service policies*. arXiv preprint 1503.01535.

Borodin, A., & El-Yaniv, R. (1998). *Online computation and competitive analysis*. Cambridge University Press.

Caserta, M., Schwarze, S., & Voß, S. (2009). A new binary description of the blocks relocation problem and benefits in a look ahead heuristic. In C. Cotta, & P. Cowling (Eds.), *Evolutionary computation in combinatorial optimization, lecture notes in computer science: 5482* (pp. 37–48). Springer Berlin Heidelberg.

Caserta, M., Schwarze, S., & Voß, S. (2011). Container rehandling at maritime container terminals. In J. Böse (Ed.), *Handbook of terminal planning operations*. In *chapter 13: 49* (pp. 247–269). Springer New York.

Caserta, M., Schwarze, S., & Voß, S. (2012). A mathematical formulation and complexity considerations for the blocks relocation problem. *European Journal of Operational Research, 219*, 96–104.

Caserta, M., & Voß, S. (2009). Corridor selection and fine tuning for the corridor method. In T. Stützle (Ed.), *Learning and intelligent optimization, lecture notes in computer science: 5851* (pp. 163–175). Springer Berlin Heidelberg.

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

Dekker, R., Voogd, P., & van Asperen, E. (2006). Advanced methods for container stacking. *OR Spectrum, 28*, 563–586.

Duinkerken, M. B., Evers, J. J., & Ottjes, J. A. (2001). A simulation model for integrating quay transport and stacking policies on automated container terminals. In *Proceedings of the 15th European simulation multiconference (ESM2001). SCS, Prague*.

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

Grötschel, M., Krumke, S. O., Rambau, J., Winter, T., & Zimmermann, U. T. (2001). Combinatorial online optimization in real time. In M. Grötschel, S. O. Krumke, & J. Rambau (Eds.), *Online optimization of large scale systems* (pp. 679–704). Springer.

Jaillet, P., & Wagner, M. R. (2010). Online optimization. *International series in operations research and management science*. Springer-Verlag.

Jang, D.-W., Kim, S. W., & Kim, K. H. (2013). The optimization of mixed block stacking requiring relocations. *International Journal of Production Economics, 143*, 256–262.

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.

Lee, Y., & Hsu, N. (2007). An optimization model for the container pre-marshalling problem. *Computers & Operations Research, 34*, 3295–3313.

Lehnfeld, J., & Knust, S. (2014). Loading, unloading and premarshalling of stacks in storage areas: Survey andclassification. *European Journal of Operational Research, 239*(2), 297–312.

Murty, K. G., Wan, Y.-W., Liu, J., Tseng, M. M., Leung, E., Lai, K.-K., & Chiu, H. W. C. (2005). Hongkong international terminals gains elastic capacity using a data-intensive decision-support system. *Interfaces, 35*, 61–75.

Petering, M. E., & Hussein, M. I. (2013). A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem. *European Journal of Operational Research, 231*, 120–130.

Rei, R., & Pedroso, J. a. P. (2012). Tree search for the stacking problem. *Annals of Operations Research, 203*, 371–388.

Tang, L., Zhao, R., & Liu, J. (2012). Models and algorithms for shuffling problems in steel plants. *Naval Research Logistics, 59*, 502–524.

Ünlüyurt, T., & Aydin, C. (2012). Improved rehandling strategies for the container retrieval process. *Journal of Advanced Transportation, 46*, 378–393.

van Asperen, E., Borgman, B., & Dekker, R. (2013). Evaluating impact of truck announcements on container stacking efficiency. *Flexible Services and Manufacturing Journal, 25*, 543–556.

Wan, Y.-w., Liu, J., & Tsai, P.-c. (2009). The assignment of storage locations to containers for a container stack. *Naval Research Logistics, 56*, 699–713.

Wu, K.-C., Hernández, R., & Ting, C.-J. (2009). Applying tabu search for minimizing reshuffle operations at container yards. *Journal of the Eastern Asia Society for Transportation Studies, 8*.

Wu, K.-C., & Ting, C.-J. (2010). A beam search algorithm for minimizing reshuffle operations at container yards. In *Proceedings of the international conference on logistics and maritime systems, September 15 - 17, Busan, Korea* (pp. 703–710).

Yang, J. H., & Kim, K. H. (2006). A grouped storage method for minimizing relocations in block stacking systems. *Journal of Intelligent Manufacturing, 17*, 453–463.

Yu, M., & Qi, X. (2013). Storage space allocation models for inbound containers in an automatic container terminal. *European Journal of Operational Research, 226*, 32–45.

Zehendner, E., Caserta, M., Feillet, D., Schwarze, S., & Voß, S. (2015). An improved mathematical formulation for the blocks relocation problem. *European Journal of Operational Research, 245*(2), 415–422.

Zehendner, E., & Feillet, D. (2014). A branch and price approach for the container relocation problem. *International Journal of Production Research, 52*(24), 7159–7176.

Zhang, H., Guo, S., Zhu, W., Lim, A., & Cheang, B. (2010). An investigation of IDA* algorithms for the container relocation problem. In N. García-Pedrajas, F. Herrera, C. Fyfe, J. M. Benítez, & M. Ali (Eds.), *Trends in applied intelligent systems, lecture notes in computer science: 6096* (pp. 31–40). Springer Berlin Heidelberg.

Zhao, W., & Goodchild, A. V. (2010). The impact of truck arrival information on container terminal rehandling. *Transportation Research Part E, 46*, 327–343.

Zhu, W., Qin, H., Lim, A., & Zhang, H. (2012). Iterative deepening A* algorithms for the container relocation problem. *IEEE Transactions on Automation Science and Engineering, 9*, 710–722.