European Journal of Operational Research 271 (2018) 40–56

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

A new effective unified model for solving the Pre-marshalling and Block Relocation Problems

Marcos de Melo da Silva*, Sophie Toulouse, Roberto Wolfler Calvo

LIPN (UMR CNRS 7030) – Institut Galilée, Université Paris-Nord, 99, avenue Jean-Baptiste Clément, Villetaneuse 93430, France

A R T I C L E I N F O

Article history:
Received 15 March 2017
Accepted 2 May 2018
Available online 10 May 2018

MSC:
90-02
90C10

Keywords:
Integer programming
Container terminals
Pre-Marshalling Problem
Block Relocation Problem

A B S T R A C T

Container terminals are exchange hubs that interconnect many transportation modes and facilitate the flow of containers. Among other elements, terminals include a yard which serves as temporary storage space. In the yard, containers are piled up by cranes to form blocks of stacks. During the shipment process, containers are carried from the stacks to ships following a given sequence. Hence, if a high priority container is placed below low priority ones, such obstructing containers have to be moved (relocated) to other stacks. Given a set of stacks and a retrieval sequence, the aim in the *Pre-marshalling Problem* (PMP) is to sort the initial configuration according to the retrieval sequence using a minimum number of relocations, so that no new relocations are needed during the shipment. The objective in the *Block Relocation Problem* (BRP) is to retrieve all the containers according to the retrieval sequence by using a minimum number of relocations. This paper presents a new unified integer programming model for solving the PMP, the BRP, and the Restricted BRP (R-BRP) variant. The new formulations are compared with existing mathematical models for these problems, as well as with other exact methods that combines combinatorial lower bounds and the branch-and-bound (B&B) framework, by using a large set of instances available in the literature. The numerical experiments show that the proposed models are able to outperform the approaches based on mathematical programming. Nevertheless, the B&B algorithms achieve the best results both in terms of computation time and number of instances solved to optimality.

© 2018 Elsevier B.V. All rights reserved.

1. Introduction

The container was introduced in the maritime transportation in the mid-1950’s and since then it has become an essential element in the global intermodal freight transportation. After sixty years, the global containerized trade already reached an annual volume of 175 million TEUs (Twenty-foot Equivalent Unit). According to recent United Nations reviews on maritime transport, above 80% of the world merchandise trade is transported by sea vessels and around one sixth of this volume is carried in containers (United Nations, 2015, 2016). The acceptance of containers as a cargo unit is due to its relatively uniform building structure, which results in a reduction in cargo handling, also preventing damages and losses and, additionally, allowing savings in transport time.

A key element in the whole maritime transportation chain, container terminals serve as exchange hubs that interconnect transportation modes and facilitate the flow of containers. Typically, they are located in ports and connect seaside and landside transports, such as sea vessels, barges, trains or trucks. In addition the equipment required for handling the containers, the terminals also include a storage yard, which acts as a temporary storage space and prevents the need of synchronization between the transportation modes. The organization and operation of terminals need to be carefully planned in order to cope with the increasing demand and achieve acceptable customer service levels.

A container terminal can be divided in three main areas depending on the operations performed and the modes of transport being employed, namely seaside, landside, and storage yard. Taking these areas as a reference for the flow direction, containers can be classified as *import* or (*inbound*) containers, *export* (or *outbound*) containers, and *transshipment* containers (Caserta, Schwarze, & Voß, 2011; Kim & Park, 2003). Import containers arrive in the seaside area in container vessels; once unloaded by quay cranes, they are moved to the storage yard by internal vehicles and handled by yard cranes. Trains or external trucks accessing the terminal through the landside area will carry them to their final destination. Export containers follow a similar transportation chain, but in reverse order. Transshipment containers are restricted to the seaside and the storage yard areas; they are unloaded from a

* Corresponding author.
E-mail addresses: marcos.demelodasilva@lipn.univ-paris13.fr (M. de Melo da Silva), toulouse@lipn.univ-paris13.fr (S. Toulouse), wolfler@lipn.univ-paris13.fr (R. Wolfler Calvo).

https://doi.org/10.1016/j.ejor.2018.05.004
0377-2217/© 2018 Elsevier B.V. All rights reserved.

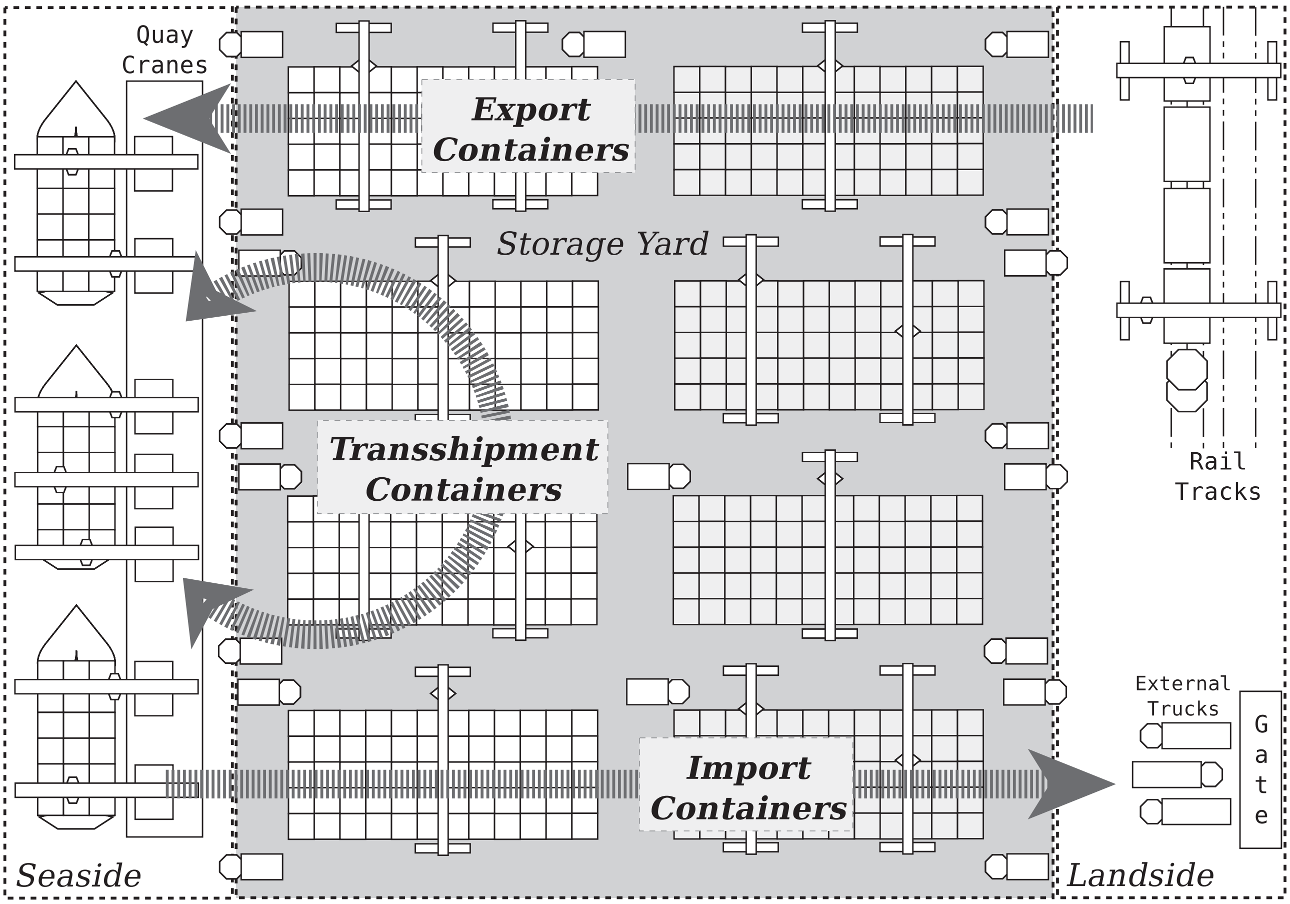

Fig. 1. Container terminal organization and container flows directions.

vessel and stored in the port yard until they are loaded to another cargo boat. Fig. 1 shows the elements that comprise a container terminal together with the directions in which containers can flow.

The dwell times for containers stored in the yard differs from one type of container to another. Trains and external trucks carrying export containers start to arrive in the port weeks before the shipment date; import containers stay in the terminal yard until they are claimed by the customers, normally between 1 and 10 days (Kim & Park, 2003). The dwell time for transshipment containers depends on the arrival and departure times of the intended vessels. The accuracy and amount of available information varies accordingly to the type of container (Caserta, Schwarze, et al., 2011). The loading sequence for export containers is already known when the shipping process starts, whereas the retrieval sequence for import containers is typically revealed during the delivery process. Customers may arrive at the container terminal for the retrieval at any time.

The turnaround time of vessels, trains and external trucks is a key factor to measure the efficiency of container terminals and therefore the customer service levels and port competitiveness (Kim & Kim, 1999; Steenken, Voß, & Stahlbock, 2004). The loading and unloading operations are the most time consuming and involve specific and scarce resources like berth areas, cranes and internal vehicles. The storage yard is another component in container terminals whose utilization has to be well planned. The containers stored in the yard are frequently disposed in such way that the available space is maximized. Savings in storage space is achieved by arranging the containers in stacks instead of spreading them around the yard. Nevertheless, this kind of arrangement demands specific handling equipment and causes many bottlenecks, provided that obstructing or hindering containers positioned in the topmost slots have to be moved while retrieving a requested container or inserting new containers in the bottommost slots.

The storage yard area is mainly occupied by stack blocks organized as depicted in Fig. 2. Containers are piled up to form stacks. Stacks are then aligned to build a bay, and bays are put together to form a block. Inside the stacks, each container is in a determined layer or tier. Thus, the precise location or slot of a container is given by the block, bay, stack and tier number. Moreover, each container can belong to a group *g*, which is defined according to the container weight class, destination port, owner, or any other classification criterion that can be used for identification during the retrieval process. Physically, the maximum width and height of a bay are limited by the handling equipment. We assume that the containers in a bay are reached from the top by yard cranes, e.g., rail mounted gantry cranes (RMG) or rubber tired gantry cranes (RTG).

Considering a bay composed of a given amount of export containers and the respective shipment/loading plan, the question concerning the sequence of operations required for retrieving the containers and which uses the minimum number of relocations is modeled in two problems described in the literature, the Pre-marshalling Problem (PMP) and the Block (Containers) Relocation Problem (BRP). The aim in the Pre-marshalling Problem is to transform the initial bay configuration into another one by using a minimum number of relocations. No retrievals are allowed during the sorting phase, and at the end of the process, all containers can be retrieved in the order indicated by the shipment plan without the need of new relocations. The objective in the Block Relocation Problem is to retrieve all the containers one by one according to

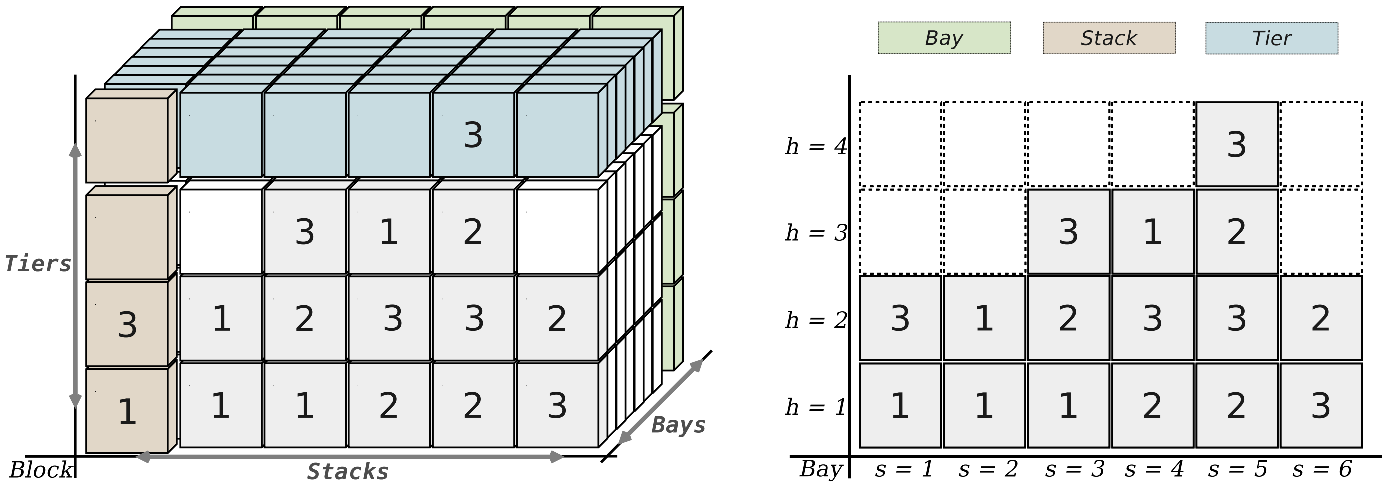

Fig. 2. (a) Container storage block, and corresponding (b) bay representation.

the loading plan using a minimum number of relocations. In both problems it is assumed that no new containers are being inserted in the bay. The PMP and BRP are shown to be $\mathcal{NP}$-Hard (Caserta, Schwarze, et al., 2011; Caserta, Schwarze, & Voß, 2012).

Many authors in the literature include an additional restriction to the BRP concerning which containers can be relocated during the retrieval process. This version of the BRP is called *Restricted* BRP (R-BRP), and it assumes that only the containers positioned above the target containers can be relocated. This assumption reduces the search space of the problem considerably and has allowed the development of efficient branch-and-bound (B&B) based exact methods. Besides that, even if an optimal solution for the R-BRP is sub-optimal for the BRP, it is still a valid solution and can be used as a good quality upper bound in exact methods for BRP.

This work presents a new unified mixed integer programming (MIP) model for solving both problems. The main difference between the PMP and BRP problems is the type of operation being performed. In this sense, the pre-marshalling is the simplest, since it requires the containers to be relocated only. The pre-marshalling formulation is extended into two models for the Block Relocation Problem by incorporating retrieval constraints. In addition, we describe the constraints necessary for our BRP formulations to solve also the R-BRP. The models require an upper bound on the number of relocations necessary to solve the problems. Hence, a simple greedy heuristic was developed for the PMP while a simplified version of a B&B algorithm described in the literature for solving the R-BRP has been implemented and used to provide upper bounds for the BRP. Computational results carried on a large set of instances available in the literature are employed to gauge the effectiveness of the proposed models. The new formulations are compared with most of the exact algorithms described in the literature, which are both methods based on mathematical programming and pure combinatorial approaches. These latter combine problem-specific combinatorial lower bound procedures, the branch-and-bound framework, node exploration dominance rules and primal heuristics. The numerical results show that the proposed models for the PMP and unrestricted BRP are able to outperform other existing formulations on these two problems, both in terms of linear relaxation quality and number of solved instances. Nevertheless, the best performance for the PMP and both BRP variants is achieved by the combinatorial B&B based approaches, and R-BRP tailored formulations. These methods obtain the best results both in terms of number of instances solved to optimality and computational time.

The paper is composed of four sections besides this introduction. Section 2 presents the related literature dealing with the Premarshalling and Block Relocation problems. Preference was given to works describing exact approaches. In Section 3, after describing the notation employed in the paper, we present the new PMP formulation, that is also the starting point for the two BRP formulations described in the subsequent subsections. In addition to the mathematical models, in Section 3 we also describe procedures to compute upper bounds for both problems. Section 4 describes the instance sets employed for benchmarking the formulations and the respective computational results. Concluding remarks are presented in Section 5.

2. Related works

Considering the amount of works published on these problems, the PMP has been less studied than the BRP in the literature. Moreover, most of the solution approaches published for the PMP are heuristics. Among the authors proposing exact approaches, Expósito-Izquierdo, Melián-Batista, and Moreno-Vega (2012) implemented an A* search algorithm capable of solving instances of small size. The authors did not propose further improvements to the basic A* framework, provided their work focuses in the development of a new randomized greedy heuristic relying in a collection of rules and a new set of instances. In a subsequent, more elaborate work, Tierney, Pacino, and Voß (2016) also developed an exact procedure for the PMP by further exploring the A* and IDA* frameworks. In order to reduce the search space of the problem, the authors embedded in their algorithms dominance rules and memory management procedures for preventing the inspection of suboptimal solutions. The instance set of Caserta and Voß (2009) were used for benchmarking their algorithms, and they are able to solve instances of small to medium size. Recently, Tanaka and Tierney (2018), building upon the work of Tierney et al. (2016), proposed an iterative depending B&B algorithm that solves instances of medium and relatively large size. The lower bound procedures and branching rules embedded in their B&B method are similar to those introduced in Tierney et al. (2016). Up to this date, these two algorithms achieve the best results for the PMP in terms of instances solved to optimality. van Brink and van der Zwaan (2014) proved complexity results for two PMP variants in which the final configuration of the stacks is either explicitly defined or sorted accordingly to a given rule, e.g., the containers should be sorted in non-decreasing order with respect to the group they belong to, and the height of the stacks is bounded. For the case in which the final configuration should be sorted accordingly to a given rule, the authors presented a branch-and-price procedure that they benchmark in a large set of random generated instances with grouped container priorities. The paper of Lee and Hsu (2007) is, to the best of our knowledge, the only work, in the pre-marshalling literature, proposing a mathematical programming approach to solve the PMP. The authors developed an integer time-indexed multi-commodity network flow model for the PMP that can solve small instances. In order to deal with larger instances, the authors implemented a heuristic based on their formulation. All the methods proposed by Lee and Hsu (2007), Tierney et al. (2016) and Tanaka and Tierney (2018) solve the PMP by considering unique and grouped container priorities. Among the recent heuristic approaches, Caserta and Voß (2009) proposed a heuristic based on the Corridor Method. Lee and Chao (2009) developed a neighborhood search algorithm, and Bortfeldt and Forster (2012) proposed a heuristic tree search procedure. Jovanovic, Tuba, and Voß (2015) implemented a deterministic version of the greedy algorithm proposed by Expósito-Izquierdo et al. (2012), which is able to find solutions of better quality than the original randomized version.

The BRP literature is concentrated mainly in the R-BRP considering unique container priorities, and only a few papers deal with both problems: the R-BRP and the BRP. The majority of the methods are based on the branch-and-bound framework. The limitations imposed by the R-BRP are the main reason, since they considerably reduce the search space. Kim and Hong (2006), Wu, Ting, and Hernández (2010), Ünlüyurt and Aydin (2012), Expósito-Izquierdo, Melián-Batista, and Moreno-Vega (2015), and Tanaka and Takii (2016) proposed B&B algorithms for the R-BRP. Caserta, Voß, and Sniedovich (2011), Zhu, Qin, Lim, and Zhang (2012), Expósito-Izquierdo, Melián-Batista, and Moreno-Vega (2014), and Ku and Arthanari (2016) proposed Dynamic Programming (DP) and A* based algorithms. The A* algorithms of Zhu et al. (2012) and Expósito-Izquierdo et al. (2014) are also able to solve the BRP. Caserta et al. (2012) developed mathematical formulations and complexity results for both the BRP and R-BRP. Expósito-Izquierdo et al. (2015) and Zehendner, Caserta, Feillet, Schwarze, and Voß (2015) proposed corrections and improvements for the R-BRP model presented by Caserta et al. (2012). Galle, Barnhart, and Jaillet (2018) recently put forward a new MIP formulation for the R-BRP which explores a binary representation that was initially described on the work of Caserta, Schwarze, and Voß (2009). Compared to other formulations for the R-BRP, they are now able to solve a larger number of instances. With respect to the unrestricted version of the BRP, Petering and Hussein (2013) proposed a MIP formulation and a heuristic for the problem; and recently, Tricoire, Scagnetti, and Beham (2018) developed new lower bounding procedures, which were then incorporated within a B&B framework so as to obtain an exact algorithm for solving the BRP. The same authors also modified their exact algorithm in order to have a B&B based heuristic capable of solving bigger BRP instances. Similarly, Tanaka and Mizuno (2018), improved on their previous work for the R-BRP, by developing a new exact B&B algorithm for the BRP. In addition to new lower bounding procedures, the authors also introduce several dominance rules designed to eliminate unnecessary nodes during the tree exploration. They report improved computational results for both the BRP and the R-BRP on instance benchmarks from the literature. Up to this date, the new B&B of Tanaka and Mizuno (2018) is the best exact algorithm for both problems.

Table 1 summarizes the main exact approaches for the BRP and R-BRP in the literature. Most of these papers also describe heuristic algorithms that are used to provide upper bounds and to solve the larger instances. The interested reader on heuristic approaches can refer to Jovanovic and Voß (2014).

A recent survey and classification scheme for loading, unloading, and pre-marshalling operations in stack storage contexts is proposed by Lehnfeld and Knust (2014). For surveys on container terminals and their operations see Steenken et al. (2004), Dekker, Voogd, and van Asperen (2006), Vis and Roodbergen (2009), Carlo, Vis, and Roodbergen (2013), (2014a), (2014b).

3. Mathematical models

To the best of our knowledge none of the exact approaches proposed in the literature solves the BRP in which the precedence among the containers is defined by both unique containers and groups. In addition, the existing PMP formulation relies on a complex combination of flows for representing the stacks and the relocation of containers that seems to reduce its final performance. In this section, we present a PMP formulation that models the stacks and relocations in a straightforward and direct way, which facilitates the incorporation of alternative characteristics and limitations of the PMP or other related problems.

The following notations and definitions will be employed in the remaining of this paper. Let $\mathscr{S} = \{1, \ldots, S\}$ be a bay composed of S stacks. Each stack is defined by a set of position $\mathscr{H} = \{1, \ldots, H\}$ and can hold at most H uniformly shaped containers. The total amount of containers in the bay is defined by N, and a PMP instance always has a feasible solution if $N \leq (S \times H) - H$. Moreover, each container belongs to a group g, where the set of groups $\mathscr{G}$ is defined as $\mathscr{G} = \{1, \ldots, G\}$, $(G \leq N)$. The *target group* refers to containers of the group $t_g \in \{1, \ldots, G\}$ which must be retrieved next. Provided that stacks are the storage structures being used, the Last In First Out (LIFO) policy has to be respected, i.e., only the topmost containers can be directly reached by a crane. Thus, to access a specific container, any hindering container above it has to be moved to other stacks. Obstructing containers on top of target containers are called *mis-overlays* (or *deadlocks*). Two kinds of operations can be performed while moving containers within or outside of the bay, namely relocation and retrieval. A *relocation* occurs when a container has to be moved from one stack to another. It is assumed that containers can be relocated to other stacks only and hence there exists enough space above the stacks to perform the necessary relocations. A *retrieval* is performed when a container of the *target group* is moved outside the storage space.

We now describe the models proposed for solving the PMP and the BRP. They are time-indexed binary formulations which make use of the set of discrete time steps $\mathscr{T} = \{1, \ldots, T\}$ for capturing each bay operation: either relocation or retrieval. Three sets of binary variables are employed to represent the container relocation moves and to track the changes in the bay layout after these operations: the movement variables $y \in \{0, 1\}$ and $z \in \{0, 1\}$ and the bay state variables $x \in \{0, 1\}$. Variables z indicates from which slot a container is picked, and variables y informs where it will be placed. They are defined as follows:

x^t_{gsh}: equals 1 if there is a container of type $g \in \mathscr{G}$ in the slot $h \in \mathscr{H}$ of stack $s \in \mathscr{S}$ at the end of time interval $t \in \mathscr{T} \cup \{0\}$, otherwise it is 0.

y^t_{gsh}: equals 1 if there is a container of type $g \in \mathscr{G}$ going into the slot $h \in \mathscr{H}$ of stack $s \in \mathscr{S}$ during time interval $t \in \mathscr{T}$, and 0 otherwise.

z^t_{gsh}: equals 1 if there is a container of type $g \in \mathscr{G}$ leaving the slot $h \in \mathscr{H}$ of stack $s \in \mathscr{S}$ during time interval $t \in \mathscr{T}$, and 0 otherwise.

The information concerning the initial bay configuration C_{gsh} is given at the beginning of the solving process, Fig. 3 illustrates how the state variables x are initialized and how a relocation

Table 1
Summary of the BRP literature in exact methods.

Author and year	BRP version		Container priorities		Method
	Restricted	Unrestricted	Unique	Groups	
Kim and Hong (2006)	✓		✓	✓	B&B
Wu et al. (2010)	✓		✓		B&B
Caserta, Voß, et al. (2011)	✓		✓		Dynamic program.
Caserta et al. (2012)	✓	✓	✓		MIP model
Ünlüyurt and Aydin (2012)	✓		✓		B&B
Zhu et al. (2012)	✓	✓	✓		IDA* search
Petering and Hussein (2013)		✓	✓		MIP model
Expósito-Izquierdo et al. (2014)	✓	✓	✓		A* search
Expósito-Izquierdo et al. (2015)	✓		✓		MIP model, B&B
Zehendner et al. (2015)	✓		✓		MIP model
Ku and Arthanari (2016)	✓		✓		Abstraction method
Tanaka and Takii (2016)	✓		✓	✓	B&B
Tricoire et al. (2018)		✓	✓		B&B
Galle et al. (2018)	✓		✓		MIP model
Tanaka and Mizuno (2018)	✓	✓	✓		B&B
This work	✓	✓	✓	✓	MIP models

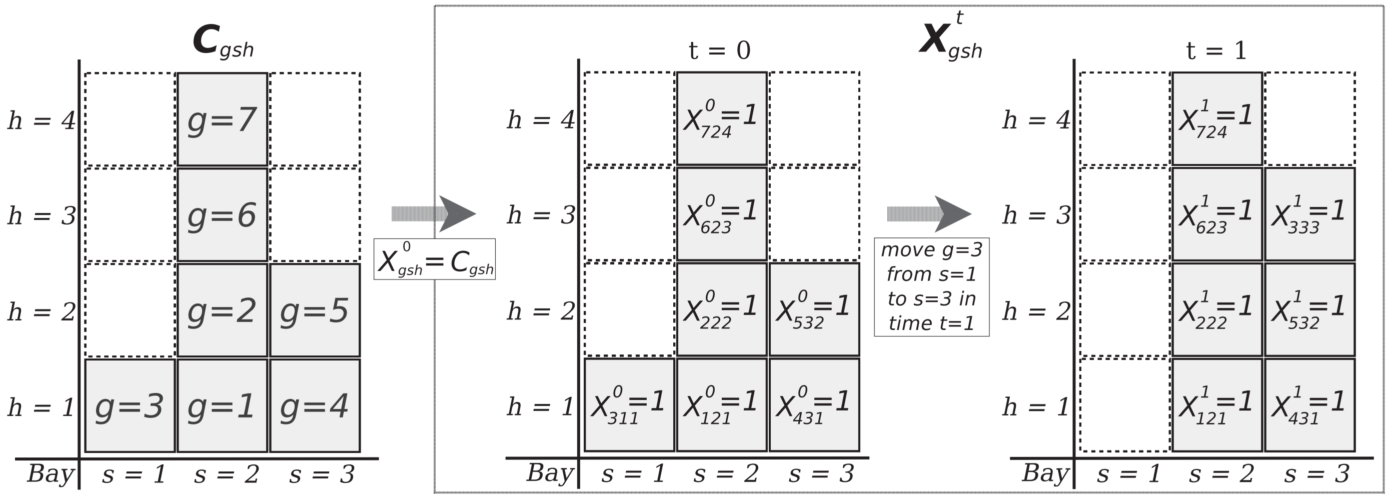

Fig. 3. Bay matrix representation and respective state variables x^t_{gsh} initialization ($t = 0$) and relocation ($t = 1$).

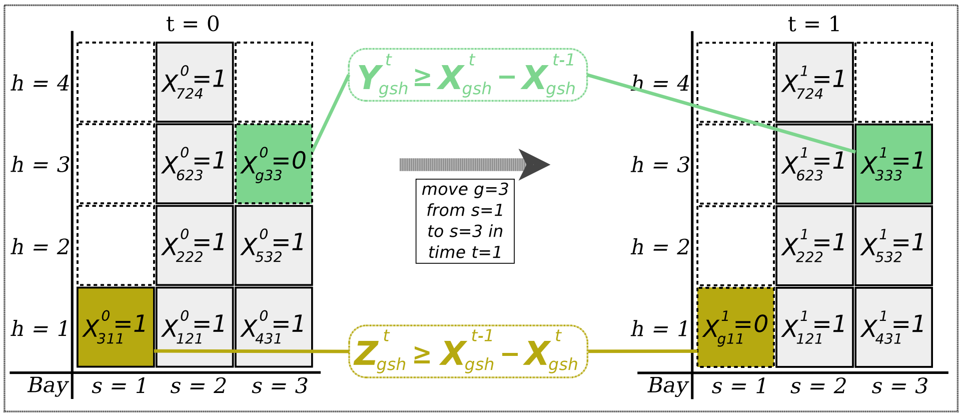

Fig. 4. Bay state variables x^t_{gsh} and respective movement variables z^t_{gsh} and y^t_{gsh} describing a relocation ($t = 1$).

move affects the variables from one time step to another. The values assumed by the movement variables in each time step are defined accordingly to the state variables, $z^t_{gsh} \geq x^{t-1}_{gsh} - x^t_{gsh}$ and $y^t_{gsh} \geq x^t_{gsh} - x^{t-1}_{gsh}$. They represent the positive difference between two consecutive time steps. Fig. 4 shows how these movement variables are linked to the state variables x.

3.1. The Pre-marshalling model (PMP$_{m1}$)

In the PMP, the task is to transform the initial bay configuration into another, usually unknown configuration, in which the containers are sorted accordingly to a given scheme, by using a minimum number of relocations. This objective is addressed by Expression

(1), where the cost of each relocation is constant.

$$(PMP_{m1}) \quad \min \quad z = \sum_{t=1}^{T}\sum_{g=1}^{G}\sum_{s=1}^{S}\sum_{h=1}^{H} y_{gsh}^{t} \tag{1}$$

The constraints are divided in two groups according to the variables involved (bay state variables and/or movement variables). Hence, there are constraints responsible for representing the stacks and keeping their consistency during the entire process, and other constraints responsible for modeling the relocation movements while observing the LIFO principle. The first group is presented thereon.

$$x_{gsh}^{0} = C_{gsh} \qquad g \in \mathscr{G},\ s \in \mathscr{S},\ h \in \mathscr{H} \tag{2}$$

$$\sum_{g=1}^{G} x_{gsh}^{t} \le 1 \qquad t \in \mathscr{T},\ s \in \mathscr{S},\ h \in \mathscr{H} \tag{3}$$

$$\sum_{g=1}^{G} x_{gsh}^{t} \ge \sum_{g=1}^{G} x_{gs(h+1)}^{t} \qquad t \in \mathscr{T},\ s \in \mathscr{S},\ h \in \mathscr{H} \setminus \{H\} \tag{4}$$

$$\sum_{s=1}^{S}\sum_{h=1}^{H} x_{gsh}^{t-1} = \sum_{s=1}^{S}\sum_{h=1}^{H} x_{gsh}^{t} \qquad t \in \mathscr{T},\ g \in \mathscr{G} \tag{5}$$

Constraints (2) initialize the bay state variables in time step 0 with a given bay configuration; Constraints (3) assure that each bay slot is occupied by at most one group, and Constraints (4) prevent the occurrence of empty spaces among two occupied tiers within a stack. Eq. (5) ensures that the amount of containers in each group is kept constant for all time steps, provided relocation is the only operation allowed in the PMP.

Two possibilities can be explored when defining the final configuration of the stacks in the PMP. In the first case, the arrangement of each container inside the bay is explicitly defined. Constraints (6) ensure the containers are organized accordingly to a final configuration F_{gsh} provided as input data. Such modeling constraints can be interesting in a practical scenario where a predefined arrangement need to be achieved. For instance, when the containers have to be located at a specific bay slot in the end of the sorting process due to operational constraints (e.g., cargo incompatibility, connection to power inlets, etc.).

$$x_{gsh}^{T} = F_{gsh} \quad g \in \mathscr{G},\ s \in \mathscr{S},\ h \in \mathscr{H} \tag{6}$$

In the second case, the stacks are sorted accordingly to a given rule, e.g., the containers should be sorted in non-decreasing order (starting from the topmost slot) with respect to the group they belong to. This is the sorting rule usually adopted in the literature. Constraints (7) can be added to achieve this goal.

$$\sum_{q=g}^{G} x_{qs(h-1)}^{T} \ge x_{gsh}^{T} \quad g \in \mathscr{G},\ s \in \mathscr{S},\ h \in \mathscr{H} \setminus \{H\} \tag{7}$$

The second group of constraints is the following.

$$\sum_{g=1}^{G}\sum_{s=1}^{S}\sum_{h=1}^{H} y_{gsh}^{t} \le 1 \qquad t \in \mathscr{T} \tag{8}$$

$$\sum_{g=1}^{G}\sum_{s=1}^{S}\sum_{h=1}^{H} z_{gsh}^{t} \le 1 \qquad t \in \mathscr{T} \tag{9}$$

$$x_{gsh}^{t} + z_{gsh}^{t} = x_{gsh}^{t-1} + y_{gsh}^{t} \qquad t \in \mathscr{T},\ g \in \mathscr{G},\ s \in \mathscr{S},\ h \in \mathscr{H} \tag{10}$$

$$\sum_{g=1}^{G}\sum_{s=1}^{S}\sum_{h=1}^{H} z_{gsh}^{t-1} \ge \sum_{g=1}^{G}\sum_{s=1}^{S}\sum_{h=1}^{H} z_{gsh}^{t} \qquad t \in \mathscr{T} \setminus \{1\} \tag{11}$$

$$x_{gsh}^{t} \in \{0,1\} \qquad t \in \mathscr{T} \cup \{0\},\ g \in \mathscr{G},\ s \in \mathscr{S},\ h \in \mathscr{H} \tag{12}$$

$$y_{gsh}^{t} \in \{0,1\} \qquad t \in \mathscr{T},\ g \in \mathscr{G},\ s \in \mathscr{S},\ h \in \mathscr{H} \tag{13}$$

$$z_{gsh}^{t} \in \{0,1\} \qquad t \in \mathscr{T},\ g \in \mathscr{G},\ s \in \mathscr{S},\ h \in \mathscr{H} \tag{14}$$

Constraints (8) and (9) limit the number of relocations that can take place in each time step to at most one. Eq. (10) assures that a container picked up from the bay is put back. These constraints build the link between the bay state variables and the movement variables. Notice that Eq. (10) alone does not suffice to prevent “empty” relocations, i.e., when, at the same time step, a container is relocated to the stack from which it was picked up. As this type of move can be considered valid relocations, they would make the solution cost increase. Nevertheless, the solutions with such relocations are sub-optimal and thus are avoided by the objective function, which forces the number of relocations to be kept minimal. In addition, such relocations can be prevented by incorporating Constraints (19) in the formulation. The number of time steps T is an upper bound to the optimal number of relocations. Therefore, there will be time steps where no operation is performed. Constraints (11) force these idle time steps to occur in the end of the process. Constraints (12)–(14) define the domain of all variables.

3.2. Strengthening the proposed model

The objective function (1) together with Constraints (2)–(5), one of the final configuration Constraints (6) or (7), Constraints (9)–(11), and the variable domain constraints are enough to define a valid model for the Pre-marshalling Problem. Nevertheless, this model can be strengthened by including constraints to enforce the LIFO policy when relocating blocks from one stack to another. Constraints (15) and (16) assure that blocks will be put in the available top tiers of the stacks only, and Constraints (17) force the topmost containers to be selected during a relocation.

$$\sum_{g=1}^{G} y_{gs1}^{t} \le 1 - \sum_{g=1}^{G} x_{gs1}^{t-1} \qquad t \in \mathscr{T},\ s \in \mathscr{S}, \tag{15}$$

$$\sum_{g=1}^{G} y_{gs(h+1)}^{t} \le \sum_{g=1}^{G} (x_{gsh}^{t-1} - x_{gs(h+1)}^{t-1}) \qquad t \in \mathscr{T},\ s \in \mathscr{S},\ h \in \mathscr{H} \setminus \{H\} \tag{16}$$

$$\sum_{g=1}^{G} z_{gsh}^{t} \le \sum_{g=1}^{G} (x_{gsh}^{t-1} - x_{gs(h+1)}^{t-1}) \qquad t \in \mathscr{T},\ s \in \mathscr{S},\ h \in \mathscr{H} \setminus \{H\} \tag{17}$$

Constraints (18) prevent a container that was relocated in the previous time step to be relocated again in the current time step. Cyclic and other transitive moves are avoided by these constraints.

$$\sum_{h=1}^{H} (y_{gsh}^{t-1} + z_{gsh}^{t}) \le 1 \qquad t \in \mathscr{T} \setminus \{1\},\ g \in \mathscr{G},\ s \in \mathscr{S} \tag{18}$$

To simplify and further strengthen the model a few modifications can be done. Constraints (4) can be suppressed as they are implied by LIFO constraints (15)–(17). Constraints (3) can be replaced by Constraints (19).

$$\sum_{g=1}^{G} (x_{gsh}^{t} + z_{gsh}^{t}) \le 1 \qquad t \in \mathscr{T},\ s \in \mathscr{S},\ h \in \mathscr{H} \tag{19}$$

One of the weaknesses of this extended model, when the linear relaxation is solved, concerns the movement flow variables z and y allowing flow fractions going out from a bottom slot and coming into in a top slot of the same stack. The constraints described below are devised in an attempt to avoid this behavior. Constraints (20) ensure that a container cannot be both picked up and dropped-off from/to the same stack. Similarly, Constraints (21) ensure that a container g being dropped-off at stack s need to be picked up from a stack $r \in S$, $r \neq s$, and conversely, Constraints (22) ensure that a container g being picked up from stack s need to be dropped-off in a stack $r \in S$, $r \neq s$.

$$\sum_{g=1}^{G}\sum_{h=1}^{H}(z_{gsh}^t + y_{gsh}^t) \le 1 \qquad t \in \mathscr{T},\ s \in \mathscr{S} \tag{20}$$

$$\sum_{h=1}^{H} y_{gsh}^t \le \sum_{\substack{r=1\\ r\neq s}}^{S}\sum_{h=1}^{H} z_{grh}^t \qquad t \in \mathscr{T}, g \in \mathscr{G}, s \in \mathscr{S} \tag{21}$$

$$\sum_{h=1}^{H} z_{gsh}^t \le \sum_{\substack{r=1\\ r\neq s}}^{S}\sum_{h=1}^{H} y_{grh}^t \qquad t \in \mathscr{T}, g \in \mathscr{G}, s \in \mathscr{S} \tag{22}$$

The formulation proposed here depends on a parameter T, that is an upper bound to the optimal value of the number of relocations needed to solve the PMP. This is similar to the model presented by Lee and Hsu (2007), but can be seen as a drawback of these models, since their solution time depends on this parameter. During our experiments, we refer to the formulation composed of the objective function (1) and constraints (2), (4), (5), (7), (9)–(19) as PMP model.

3.3. *Estimating parameter T for the PMP model*

In this section, we describe a greedy heuristic employed to compute a valid upper bound on the number of relocations necessary for solving the PMP. The algorithm repeats the following three steps until a valid PMP configuration is reached. Additional details are presented in Algorithm 3 (Appendix A).

- *Step 1 (Stack selection).* If the procedure receives as input a stack, it starts *Step 2*. Otherwise, the stack with the smallest number of containers is selected. Ties are broken by choosing the stack with the highest container index. If the chosen stack is already empty the algorithm proceeds to *Step 3*, otherwise, *Step 2* is executed.
- *Step 2 (Empty).* The chosen stack is emptied by relocating its containers into other stacks with available space. The sorted stacks with available space are considered first, provided that no new mis-overlays will be formed after the relocations. If no sorted stack is available, the unsorted stacks with available space are then taken into account. The priority is given to the stacks whose containers have indexes smaller than the container being relocated. Finally, if none of these conditions is met, a sorted stack with available space is exceptionally considered.
- *Step 3 (Rebuild).* The topmost containers with the highest indexes are relocated back to the emptied stack. The last two tiers are not filled. They are used as temporary space during *Step 2*.

The heuristic is executed S times. Each time it starts with a different stack from $\{1, \dots, S\}$. The execution with the smallest number of relocations is recorded and used as the upper bound for the parameter T.

3.4. *The block relocation model* (BRP_{m1})

The current and the next section present two BRP formulations that are extensions of the PMP model described above. While in the PMP the relocation is the only operation being performed, the BRP also requires containers to be retrieved, i.e., containers are moved out of the bay. The containers leaving the bay are placed in an output queue, which is responsible for dictating the order in which the retrievals are performed. Hence, in addition to the bay state and movement variables defined for the PMP (x, y, and z), this first BRP model employs two further sets of variables: w_{gn}^t represent a queue of size N (i.e., the total amount of containers that will be retrieved); and the k_{gn}^t variables, which represent the movement of containers from the bay to the queue. Therefore,

k_{gn}^t: equals 1 if a container of type $g \in \mathscr{G}$ leaves the bay and is placed in position $n \in \{1, \dots, N\}$ during time interval $t \in \mathscr{T}$, and 0 otherwise.

w_{gn}^t: equals 1 if a container of type $g \in \mathscr{G}$ is in position $n \in \{1, \dots, N\}$ at the end of time interval $t \in \mathscr{T} \cup \{0\}$, and 0 otherwise.

The order in which the containers will be retrieved from the bay is an input parameter provided by the queue configuration Q_{gn}. The most frequent objective function for the BRP found in the literature is also to minimize the number of relocations, as for the PMP.

The PMP objective function (1) together with Constraints (2), (4), (8), (9), (11)–(17) and (19), shown in expressions (23) and (24), are combined with Constraints (25)–(38) to form a valid BRP model.

$$(BRP_{m1}) \quad \min \quad z = \sum_{t=1}^{T}\sum_{g=1}^{G}\sum_{s=1}^{S}\sum_{h=1}^{H} y_{gsh}^t \tag{23}$$

$$s.t. \quad (2), (4), (8), (9), (11) - (14), (15) - (17), (19) \tag{24}$$

$$\sum_{g=1}^{G}\sum_{s=1}^{S}\sum_{h=1}^{H} x_{gsh}^T = 0 \tag{25}$$

$$w_{gn}^0 = 0 \qquad g \in \mathscr{G},\ n \in \{1, \dots, N\} \tag{26}$$

$$w_{gn}^T = Q_{gn} \qquad g \in \mathscr{G},\ n \in \{1, \dots, N\} \tag{27}$$

$$\sum_{g=1}^{G} w_{gn}^t \le 1 \qquad t \in \mathscr{T},\ n \in \{1, \dots, N\} \tag{28}$$

$$\sum_{g=1}^{G}\sum_{n=1}^{N} k_{gn}^t \le 1 \qquad t \in \mathscr{T} \tag{29}$$

$$\sum_{g=1}^{G} w_{gn}^t \ge \sum_{g=1}^{G} w_{g(n+1)}^t \qquad t \in \mathscr{T},\ n \in \{1, \dots, (N-1)\} \tag{30}$$

$$\sum_{n=1}^{N} w_{gn}^t = \sum_{n=1}^{N} (w_{gn}^{t-1} + k_{gn}^t) \qquad t \in \mathscr{T},\ g \in \mathscr{G} \tag{31}$$

$$w_{gn}^t = w_{gn}^{t-1} + k_{gn}^t \qquad t \in \mathscr{T},\ g \in \mathscr{G},\ n \in \{1, \dots, N\} \tag{32}$$

$$\sum_{s=1}^{S}\sum_{h=1}^{H} x_{gsh}^{t-1} = \sum_{n=1}^{N} k_{gn}^t + \sum_{s=1}^{S}\sum_{h=1}^{H} x_{gsh}^t \qquad t \in \mathscr{T},\ g \in \mathscr{G} \tag{33}$$

$$x_{gsh}^t + z_{gsh}^t \ge x_{gsh}^{t-1} + y_{gsh}^t \qquad t \in \mathscr{T},\ g \in \mathscr{G},\ s \in \mathscr{S},\ h \in \mathscr{H} \tag{34}$$

$$\sum_{s=1}^{S}\sum_{h=1}^{H} y_{gsh}^t + \sum_{n=1}^{N} k_{gn}^t = \sum_{s=1}^{S}\sum_{h=1}^{H} z_{gsh}^t \qquad t \in \mathscr{T},\ g \in \mathscr{G} \tag{35}$$

$$\sum_{g=1}^{G}\sum_{s=1}^{S}\sum_{h=1}^{H} z_{gsh}^t = 1 \qquad t \in \{1, \dots, N\} \tag{36}$$

$$k_{gn}^t \in \{0, 1\} \qquad t \in \mathscr{T},\ g \in \mathscr{G},\ n \in \{1, \dots, N\} \tag{37}$$

$$w_{gn}^t \in \{0, 1\} \quad t \in \mathscr{T} \cup \{0\}, \qquad g \in \mathscr{G},\ n \in \{1, \dots, N\} \tag{38}$$

Constraints (25) ensure that the bay will be empty at the end of the retrieving process. Constraints (26) and (27) define the initial and final values for the output queue variables, respectively. These constraints can be suppressed when only a partial amount of the containers in the bay are planned to be retrieved. Constraints (28) allow at most one container in each queue slot, and Constraints (29) limit the number of retrieved containers per time step to at most one. Constraints (30) ensure that the retrievals will be performed according to the order defined by the output queue. Constraints (31) and (32) link the retrieval movement variables and the queue variables, and also ensure the consistency of the queue from one time step to another. The consistency Constraints (33) force the removal of a retrieved container from the bay. The link among bay state variables, relocation variables, and retrieval variables is defined in Constraints (34) and (35). Given that in this model the z variables are employed to describe retrievals, in addition to relocations, and knowing that the N containers in the BRP bay have to be retrieved, Constraints (36) forces that in the first N time steps there is one z variable that is equal one, which remains true even if a relocation is being performed instead of a retrieval. Constraints (37) and (38) specify the variable domains.

In the formulations presented so far at most one operation, either a relocation or a retrieval, can be performed per time step. Consequently, while the parameter T in the PMP model is an upper bound in the number of relocations, for the BRP, T is the sum of an upper bound of the number of relocations (indicated with $\bar{T}$) and N, the number of containers in the bay that will be retrieved, i.e., $T = \bar{T} + N$. This is a modeling drawback present in formulation BRP$_{m1}$, and in similar BRP models in the literature. Therefore, the size of the models grows fast according to the size of the input bays. The next formulation overcomes this behavior by allowing as many retrievals as possible to be performed together with relocations, although the limit of at most one relocation per time step is still applied. Therefore, similarly to the PMP model, the parameter T will be an upper bound in the number of relocations.

3.5. The block relocation model (BRP$_{m2}$)

The second formulation for the BRP requires a different set of variables. They are different from the retrieval and queue variables defined for the previous BRP model. They are similar to the relocation variables z^t_{gsh} in the PMP model, and they can be interpreted as relocations of containers to outside the bay. The retrieval ordering will be achieved by constraints imposed on these variables.

k^t_{gsh}: equals 1 if the container of type $g \in \mathscr{G}$ is being removed from slot $h \in \mathscr{H}$ of stack $s \in \mathscr{S}$ in time $t \in \mathscr{T}$, 0 otherwise;

As for the previous BRP formulation, the PMP objective function (1) and Constraints (2), (4), (8), (9), (12)–(17) and (20) are employed, as shown in expressions (39) and (40), additionally to the new state, movement, and retrieval Constraints (41)–(50) in the new BRP$_{m2}$.

$$(BRP_{m2}) \quad \min \quad z = \sum_{t=1}^{T}\sum_{g=1}^{G}\sum_{s=1}^{S}\sum_{h=1}^{H} y^t_{gsh} \tag{39}$$

$$s.t. \quad (2), (4), (8), (9), (12)-(14), (15)-(17), (20) \tag{40}$$

$$\sum_{g=1}^{G}\sum_{s=1}^{S}\sum_{h=1}^{H} x^T_{gsh} = 0 \tag{41}$$

$$\sum_{g=1}^{G} x^t_{gsh} + k^t_{gsh} \le 1 \qquad t \in \mathscr{T},\ s \in \mathscr{S},\ h \in \mathscr{H} \tag{42}$$

$$\sum_{s=1}^{S}\sum_{h=1}^{H} x^{t-1}_{gsh} = \sum_{s=1}^{S}\sum_{h=1}^{H}(x^t_{gsh} + k^t_{gsh}) \qquad t \in \mathscr{T},\ g \in \mathscr{G} \tag{43}$$

$$z^t_{gsh} + k^t_{gsh} = y^t_{gsh} + x^{t-1}_{gsh} - x^t_{gsh} \qquad t \in \mathscr{T},\ g \in \mathscr{G},\ s \in \mathscr{S},\ h \in \mathscr{H} \tag{44}$$

$$k^t_{gsh} \in \{0, 1\} \qquad t \in \mathscr{T},\ g \in \mathscr{G},\ s \in \mathscr{S},\ h \in \mathscr{H} \tag{45}$$

Constraints (41) ensure that the bay will be empty in the last time step, since all containers should have been retrieved. Constraints (42) limit the amount of containers in each slot and being retrieved to at most one, and consistency Constraints (43) force a container that has been retrieved to be removed from the bay. Constraints (44) build the link among bay variables, relocation variables, and retrieval variables. Constraints (45) specify the variable domain.

The constraints designed to force the containers to be retrieved in a prescribed order, usually from $(1, \ldots, G)$, are divided into two parts accordingly to the occurrence of grouped containers or not. When the containers are not grouped (i.e., $G = N$, which means that each container belongs to a unique group), Constraints (46)–(48) are enough for enforcing the retrieval order.

$$\sum_{s=1}^{S}\sum_{h=1}^{H} k^t_{gsh} \le \sum_{u=1}^{t}\sum_{s=1}^{S}\sum_{h=1}^{H} k^u_{(g-1)sh} \qquad t \in \mathscr{T},\ g \in \mathscr{G} \setminus \{1\} \tag{46}$$

$$k^t_{gsh} + \sum_{l=g+1}^{G} k^t_{ls(h+1)} \le 1 \quad t \in \mathscr{T};\ g \in \mathscr{G} \setminus \{G\},\ s \in \mathscr{S},\ h \in \mathscr{H} \setminus \{H\} \tag{47}$$

$$k^t_{gsh} \le 1 - \sum_{l=1}^{g-1} x^t_{ls(h-1)} \quad t \in \mathscr{T};\ g \in \mathscr{G} \setminus \{1\},\quad s \in \mathscr{S},\ h \in \mathscr{H} \setminus \{1\} \tag{48}$$

In cases where more than one container belongs to the same group (i.e., $G < N$) Constraints (48) are replaced by Constraints (49) and (50).

$$k^t_{gsh} \le 1 - \sum_{l=1}^{g-1} x^t_{lsi} \quad t \in \mathscr{T},\ g \in \mathscr{G} \setminus \{1\},\ s \in \mathscr{S},\ h \in \mathscr{H} \setminus \{1\},\ i \in \{1, \ldots, (h-1)\} \tag{49}$$

$$k^t_{gsh} \le 1 - \sum_{l=1}^{g-1} x^t_{lpi} \quad t \in \mathscr{T},\ g \in \mathscr{G} \setminus \{1\},\ s \in \mathscr{S},\ h \in \mathscr{H},\ p \in \mathscr{S},\ p \neq s,\ i \in \mathscr{H} \tag{50}$$

Constraints (46) prevent containers belonging to higher indexed groups to be retrieved before containers of lower indexed groups. Constraints (47) avoid two containers within the same stack to be retrieved at the same time step if the container in the lower tier belongs to a lower indexed group. Constraints (48) prevent higher indexed containers to be retrieved if they are placed immediately above lower indexed containers. Constraints (49) and (50) expand Constraints (48) and prevent a topmost container to be retrieved if there exists in the bay a container that belongs to a lower indexed group. Note that Constraints (49) and (50) could be used for both cases, grouped and non-grouped instances, though it would result in larger models with many redundant constraints for the latter case.

It is worth noting that despite being more compact, the linear relaxation of this formulation is not as good as the previous BRP model, as show by the computational experiments in Section 4.

3.6. Restricted block relocation models

The algorithms BRP$_{m1}$ and BRP$_{m1}$ can be easly extended to solve the R-BRP as well, which is different from the BRP which allows containers to be relocated among any two stacks. The R-BRP restricts the containers that can be relocated to those that are above the target container that will be retrieved next. Departing from our BRP models, Constraints (51) can be added to BRP$_{m1}$ so that it can solve the R-BRP. The same result can be obtained for BRP$_{m2}$ by incorporating Constraints (52). The restricted version of BRP$_{m1}$ is denoted by R-BRP$_{m1}$, and BRP$_{m2}$ as R-BRP$_{m2}$, respectively.

$$\sum_{j=g+1}^{G}\sum_{h=1}^{H} z_{jsh}^{t} \le w_{gg}^{t-1} + \sum_{i=1}^{g}\sum_{h=1}^{H} x_{ish}^{t} \quad t \in \mathscr{T};\ g \in \mathscr{G}\setminus\{G\},\ s \in \mathscr{S} \tag{51}$$

$$\sum_{j=g+1}^{G}\sum_{h=1}^{H} z_{jsh}^{t} \le \sum_{u=1}^{t}\sum_{r=1}^{S}\sum_{l=1}^{H} k_{grl}^{u} + \sum_{i=1}^{g}\sum_{h=1}^{H} x_{ish}^{t}$$
$$t \in \mathscr{T};\ g \in \mathscr{G}\setminus\{G\},\ s \in \mathscr{S} \tag{52}$$

The relocation movement are divided in two parts: the pickups are modeled by using the z variables and the drop-offs are captured by the y variables, we model the R-BRP by describing constraints that deactivate the z variables for all stacks but the one that contains the target container. Constraints (51) and (52) forbid containers to be picked up from stacks not containing the container g until its retrieval.

3.7. Estimating parameter T for the BRP models

The procedure implemented for providing tight upper bounds on the number of relocations (parameter T) for the BRP models is based on the B&B algorithm for the Restricted BRP of Tanaka and Takii (2016). This algorithm can handle the R-BRP with unique and grouped container priorities. The branch-and-bound implemented differs from the one presented in the paper of Tanaka and Takii (2016) by using a simpler lower bounding method.

The lower bound employed was proposed by Kim and Hong (2006) and is valid for both BRP and R-BRP. It is obtained by counting the number of deadlocks in the bay. More precisely, given a stack $s \in \{1, \dots, S\}$ and a retrieval sequence $\pi = (1, \dots, G)$, if there exists a container j below a container i in s and $\pi(j) < \pi(i)$ (i.e., the priority of j is greater than the priority i), then to retrieve j, the container i will be relocated at least once. Fig. 5 illustrates the lower bound in a bay with containers belonging to $G = 3$ and retrieval priority $\pi = (1, 2, 3)$, all the containers marked in dark gray will be relocated to give access to the containers of higher priority in the lower tiers. Algorithm 1 (Appendix A) specifies the steps for counting the mis-overlays of a bay. The procedure has complexity $\mathcal{O}(N)$, where N is the number of containers in the bay.

Algorithm 1 is used to compute the bay lower bound for the B&B procedure detailed in Algorithm 4 (Appendix A). In our implementation the root node of the B&B tree is initialized with the initial bay configuration and $ub = +\infty$. The tree is explored using the depth first search strategy in order to minimize the use of memory and find feasible solutions fast. During the search, if the number of relocations performed to arrive in the current configuration plus the lower bound is greater than the incumbent solution, the node is fathomed and a new node is selected. The incumbent solution is updated each time a smaller upper bound is found. New subproblems (nodes) are created by selecting a stack $s \in \{1, \dots, S\}$ that contains a target container, if no fathoming condition is met. When the target containers in s are not obstructed, they are retrieved and the search proceeds with this node. Otherwise, if there are containers obstructing all the target containers in the selected stack,

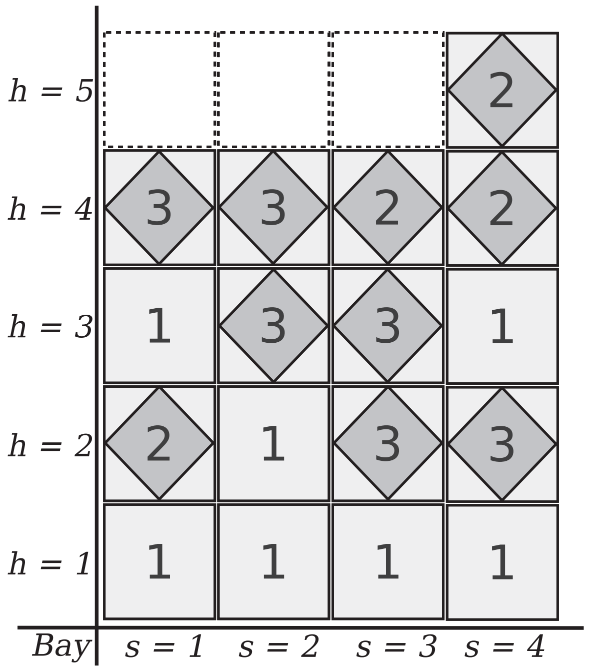

Fig. 5. Obstructing groups 2 and 3 are accounted in the BRP lower bound.

another stack $s' \in \{1, \dots, S\}, s' \neq s$ is selected and the topmost container from s is relocated to s', the lower bound is updated and the search continues with this new node. During the search, when a relocation is performed, instead of updating the lower bound by applying Algorithm 1 in the entire bay, the lower bound is updated by considering only the stack s', thus reducing the computational overhead. This operation can be done in $\mathcal{O}(h)$ comparisons, where h is the number of containers in the stack.

4. Computational results

The proposed algorithms and models were coded in C/C++ (compiled with the g++ 4.7.2 compiler) and executed on an Intel® Core^{TM} i7-2600 3.40 gigahertz CPU, with 8.0 gigabytes of RAM running under GNU/Linux Debian 7.9 (*kernel* 3.2.0-4-amd64). IBM CPLEX® 12.6 was used as LP and MIP solver. A single thread was employed during the experiments.

We evaluate the proposed model for the PMP over the two instances presented in the paper of Lee and Hsu (2007) and the set of instances of Caserta and Voß (2009). This latter has been used to assess the BRP models as well. The original set is composed of 840 instances separated in 21 classes, each one containing 40 instances. The classes are characterized by the dimensions of the bay ($S \times H$). For each bay, the first $H' = (H - 2)$ tiers of each stack are filled with containers, which means a total of $N = (S \times H')$ containers. Each one of these N containers has its position in the bay chosen at random and they are uniquely identified from 1 to N, meaning that each bay is composed of $G = N$ groups. The number of stacks and filled tiers are $S \in \{3, 4, \dots, 10\}$ and $H' \in \{3, 4, 5, 6, 10\}$; not all combinations of these values were used. Caserta and Voß (2009) point out that these bay settings are based on the physical limitations of typically used gantry cranes. The instances used for our experiments with the BRP formulations were selected from a subset of Caserta and Voß (2009) instances in which the R-BRP branch-and-bound (see Algorithm 4 in Appendix A) could solve all the 40 problems in the class to optimality within 10 minutes of computation. A total of 520 instances with $S \in \{3, 4, 5, 6, 7, 8\}$ and $H' \in \{3, 4, 5\}$ were selected. The subset of 400 instances from Caserta and Voß (2009) benchmark, with $S \in \{3, 4, 5, 6, 7, 8\}$

Table 2
Computational results concerning the linear relaxations of the proposed PMP model and the formulation of Lee and Hsu (2007) on the instances of Caserta and Voß (2009) and Lee and Hsu (2007).

Instance				Lee and Hsu			PMP_{m1}		
Name	G	S	H	lb	lb t. (second)	#lb	lb	lb t. (second)	#lb
data3–3	9	3	5	0.10	2.82	40	5.78	0.60	40
data3–4	12	4	5	0.03	4.86	40	6.71	1.72	40
data3–5	15	5	5	0.01	10.85	40	7.81	6.01	40
data3–6	18	6	5	0.00	194.89	40	9.27	5.27	40
data3–7	21	7	5	0.56	487.41	39	10.69	17.48	40
data3–8	24	8	5	1.51	1065.09	37	11.48	18.72	40
data4–4	16	4	6	0.02	28.25	40	10.20	27.86	40
data4–5	20	5	6	0.00	470.78	40	13.07	17.14	40
data4–6	24	6	6	1.41	987.48	39	14.66	48.56	40
data4–7	28	7	6	0.00	1874.44	38	17.12	57.78	40
data_6_4	3	6	4	0.20	1.67	1	6.75	0.18	1
data_12_5	6	12	5	1.61	776.45	1	26.41	10.17	1
Total						395			402

Table 3
Computational results for the proposed PMP model and the formulation of Lee and Hsu (2007) on the instances of Caserta and Voß (2009) and Lee and Hsu (2007).

Instance	Heur.	BKS[a]		Lee and Hsu					PMP_{m1}					
		Opt.	#opt	Feas.	#feas	Opt	Opt t. (second)	#opt	lb*	Feas.	#feas	Opt	Opt t. (second)	#opt
data3–3	10.58	8.78	40	7.86	29	6.63	574.53	19	5.78	8.78	40	8.78	42.79	40
data3–4	11.43	9.03	40	7.15	13	6.63	2039.65	8	6.65	9.03	40	8.90	460.67	39
data3–5	13.38	10.15	40	7.00	4	5.50	1969.31	2	6.89	10.50	40	8.78	386.84	27
data3–6	15.03	11.28	40	4.00	1	–	–	0	8.32	11.97	39	9.24	844.32	17
data3–7	18.13	12.80	40	–	0	–	–	0	8.97	14.18	33	10.00	1299.94	9
data3–8	19.48	13.53	40	–	0	–	–	0	9.17	16.22	27	10.25	1943.94	4
data4–4	20.10	15.82	40	7.00	1	7.00	854.02	1	7.25	16.24	25	9.60	397.77	5
data4–5	24.08	17.85	40	–	0	–	–	0	8.03	17.63	8	11.00	3330.81	1
data4–6	26.90	19.30	40	–	0	–	–	0	–	18.60	5	–	–	0
data4–7	31.20	21.82	40	–	0	–	–	0	–	–	0	–	–	0
data_6_4	9.00	9.00	1	9.00	1	–	–	0	6.75	9.00	1	9.00	17.26	1
data_12_5	45.00	?	?	–	0	–	–	0	–	45.00	1	–	–	0
Total			401		49			30			259			143

(Best Known Solution) BKS[a]: Tanaka and Tierney (2018).

and $H' \in \{3, 4\}$, were employed for our experiments with the PMP formulations.

Moreover, the Caserta and Voß (2009) instances selected above were transformed into grouped BRP instances using the procedure detailed in Algorithm 2 (Appendix A). Each container $i = 1, 2, \ldots, N$ in the bay is reassigned to a group $g \in \{1, \ldots, G\}$ by applying the *modulo* operation. $G \in \{3, 4, 5\}$ were employed during our experiments.

Remark 1. Note that the optimal solution of any problem with groups built as described in Algorithm 2 is a valid lower bound for the same problem with a greater number of groups and therefore also for the original problem without groups. This is true for both the BRP and PMP.

The time horizon upper bounds (parameter T) computed by the PMP heuristic and the R-BRP B&B are applied for both the proposed formulations and the models from the literature during our experiments. We imposed a time limit of one hour (3600s) for each instance. A time limit of 10 minutes (600s) has been imposed to the R-BRP B&B (Algorithm 4) when solving each instance. The other solver (i.e., CPLEX) parameters were left in their default values. The following column headings are employed in Tables 2–6, A.7, and A.8. Average results are reported for each instance class and set.

- *Instance*: specifies the instance set name and dimensions (number of groups (G) and bay size ($S \times H$)), respectively.
- *Heur.*: the average number of relocations provided by the heuristic algorithm.
- *lb*: the average linear relaxation cost of the instances whose the linear relaxation at the root node is solved within the time limit.
- *lb t. (second)*: the average computational time in seconds for solving the linear relaxation.
- *#lb*: the number of instances for which the corresponding formulation was able to solve the linear relaxation within the time limit.
- *lb**: the average linear relaxation cost of instances solved to optimality within the time limit.
- *feas.*: the average best integer solution found. Average over the number of instances for which the respective formulation finds a feasible integer solution within the time limit.
- *#feas*: the number of instances in which the corresponding model was able to find a feasible integer solution within the time limit.
- *opt*: the average optimal integer solution found. Average over the number of instances for which the respective algorithm or formulation finds the optimal solution within the time limit.
- *opt t. (second)*: the average computational time in seconds for solving the respective instance set to optimality.
- *#opt*: the number of instances in which the corresponding model was able to solve to optimality within the time limit.

4.1. Pre-marshalling Problem

The proposed formulation for the PMP has been compared with the model described in (Lee & Hsu, 2007). In order to provide fair comparisons with the work in the literature, we implemented their model and employed the same computational settings during

Table 4
Computational results concerning the linear relaxations of the proposed BRP models and the formulation of Petering and Hussein (2013) on the instances of Caserta and Voß (2009).

Instance				Petering model			BRP$_{m1}$			BRP$_{m2}$		
Name	G	S	H	lb	lb t. (second)	#lb	lb	lb t. (second)	#lb	lb	lb t. (second)	#lb
data3–3	9	3	5	2.74	1.95	40	3.52	1.56	40	2.11	0.85	40
data3–4	12	4	5	2.56	12.11	40	3.60	7.85	40	1.97	2.52	40
data3–5	15	5	5	2.57	65.87	40	3.95	34.66	40	1.89	6.08	40
data3–6	18	6	5	2.63	485.45	40	4.42	204.81	40	1.68	19.90	40
data3–7	21	7	5	2.57	1384.85	40	4.65	725.11	40	1.59	55.76	40
data3–8	24	8	5	2.62	3106.77	38	5.02	1702.46	40	1.17	168.34	40
data4–4	16	4	6	3.23	104.64	40	4.83	110.46	40	1.29	18.98	40
data4–5	20	5	6	3.52	988.68	40	5.54	543.26	40	0.94	129.93	40
data4–6	24	6	6	3.37	3069.55	38	5.81	1308.23	39	0.88	463.96	40
data4–7	28	7	6	3.05	2720.28	40	6.50	2632.07	36	0.68	1482.45	37
data5–4	20	4	7	3.98	1239.02	40	5.93	334.82	40	0.72	243.02	40
data5–5	25	5	7	3.96	3220.56	38	6.82	2001.51	38	0.50	1607.15	37
data5–6	30	6	7	3.86	2383.43	31	7.74	3258.55	30	0.38	2971.11	14
Total						505			503			488

Table 5
Computational results for the proposed BRP models and the formulation of Petering and Hussein (2013) on the instances of Caserta and Voß (2009).

Instance	R-BRP	BKS[a]		Petering model					BRP$_{m1}$						BRP$_{m2}$					
	Opt	Opt	#opt	Feas.	#feas	Opt	opt t. (second)	#opt	lb*	Feas.	#feas	Opt	Opt t. (second)	#opt	lb*	Feas.	#feas	Opt	Opt t. (second)	#opt
data3–3	5.00	4.98	40	4.98	40	4.98	71.56	40	3.52	4.98	40	4.98	6.02	40	2.11	4.98	40	4.98	2.50	40
data3–4	6.18	6.03	40	5.71	34	5.08	1080.55	24	3.60	6.03	40	6.03	221.22	40	1.97	6.03	40	6.03	192.85	40
data3–5	7.03	6.85	40	5.95	20	3.75	944.60	4	3.82	6.74	39	6.30	444.93	33	1.98	6.77	39	6.54	408.47	37
data3–6	8.40	8.28	40	5.00	3	3.00	2064.42	1	3.95	7.43	23	6.20	1151.00	15	2.21	8.05	37	6.58	438.59	19
data3–7	9.28	9.10	40	–	0	–	–	0	4.53	7.50	12	6.50	2023.41	4	2.20	8.45	31	7.21	831.52	14
data3–8	10.65	10.30	40	–	0	–	–	0	–	9.00	3	–	–	0	1.79	9.41	17	7.00	1038.05	1
data4–4	10.20	9.73	40	7.25	4	–	–	0	4.59	9.42	31	8.24	1557.51	17	1.69	9.07	28	8.20	811.63	20
data4–5	12.95	12.25	40	–	0	–	–	0	5.43	10.11	9	9.50	3420.77	2	1.64	9.90	10	8.80	1595.18	5
data4–6	14.03	13.23	40	–	0	–	–	0	–	9.33	3	–	–	0	2.25	8.80	5	7.50	802.82	2
data4–7	16.13	15.38	40	–	0	–	–	0	–	–	0	–	–	0	–	–	0	–	–	0
data5–4	15.43	14.70	40	–	0	–	–	0	4.29	10.33	6	8.00	2068.11	3	1.58	9.50	6	8.00	1301.17	3
data5–5	18.85	17.43	40	–	0	–	–	0	–	–	0	–	–	0	–	–	0	–	–	0
data5–6	22.08	20.80	40	–	0	–	–	0	–	–	0	–	–	0	–	–	0	–	–	0
Total			520		101			69			206			154			253			181

(Best Known Solution) BKS[a]: Tanaka and Mizuno (2018), and Tricoire et al. (2018).

Table 6
Computational results for the proposed R-BRP models on the instances of Caserta and Voß (2009).

Instance	R-BRP	Lit.[a]	R-BRP$_{m1}$						R-BRP$_{m2}$					
	Opt	#opt	lb*	Feas.	#feas	Opt	Opt t. (second)	#opt	lb*	Feas.	#feas	Opt	Opt t. (second)	#opt
data3–3	5.00	40	4.98	5.00	40	5.00	0.50	40	1.79	5.00	40	5.00	2.41	40
data3–4	6.18	40	5.98	6.18	40	6.18	3.41	40	1.77	6.18	40	6.18	27.17	40
data3–5	7.03	40	6.66	7.03	40	7.03	34.98	40	1.92	7.03	40	7.03	123.42	40
data3–6	8.40	40	8.03	8.40	40	8.40	37.32	40	1.91	8.45	40	8.24	457.02	38
data3–7	9.28	40	8.92	9.28	40	9.28	63.41	40	1.92	9.05	38	8.55	652.77	31
data3–8	10.65	40	10.13	10.65	40	10.65	184.14	40	1.37	10.24	34	9.38	1995.07	21
data4–4	10.20	40	8.60	10.20	40	10.20	402.58	40	1.68	10.25	40	9.71	525.19	34
data4–5	12.95	40	11.24	12.87	39	12.87	304.27	39	1.65	11.68	25	10.50	1328.35	16
data4–6	14.03	40	12.20	14.03	39	13.42	566.25	31	1.37	12.71	21	10.00	1563.52	8
data4–7	16.13	40	13.80	15.66	35	15.00	675.90	30	1.03	12.40	5	10.00	2060.36	1
data5–4	15.43	40	11.28	14.94	35	13.62	1138.77	26	1.59	12.83	18	11.18	1254.36	11
data5–5	18.85	40	13.70	18.11	28	16.15	1077.66	13	0.00	15.50	2	0.00	0.00	0
data5–6	22.08	40	16.28	21.12	17	18.33	1510.41	6	0.00	0.00	0	0.00	0.00	0
Total		520			473			425			343			280

Lit.[a]: Galle et al. (2018), Tanaka and Mizuno (2018), Tanaka and Takii (2016), and Zehendner et al. (2015).

the experiments. In their paper, Lee and Hsu (2007) executed experiments with two instances only, the first with small bay dimensions (6×4) holding 14 containers distributed in 3 groups, and the second with big bay dimensions (12×5) holding 45 containers distributed in 6 groups. In our experiments we evaluate their model also on the selected instances of Caserta and Voß (2009). The comparisons between the two formulations are detailed in Tables 2 and 3. In Table 2 we present the results concerning the linear relaxations of the models and in Table 3 we compare the results of the proposed heuristic and the feasible/optimal solutions of the formulations.

In their paper, Lee and Hsu (2007) report that the solver took more than 24 hours to solve the small problem (i.e., 6×4) to optimality. In our experiments, after one hour the process was interrupted and the gap was still above 40%. On the other hand, the proposed model found the optimal solution in less than 20 seconds. Both models fail in solving the big instance within the time limit, despite our model was still capable of finding a feasible solution.

In our experiments with the instances of Caserta and Voß (2009), the proposed model found feasible integer solutions for 259/400 instances, and among those, 143 instances were proved optimal, while the model in literature only found 49/400 feasible solutions and proved the optimality for 30 of them. The new model proved the optimality for the most part of the instances in the (3×3) set from the Caserta and Voß (2009) benchmark in less than one minute (on average). These results are explained by the fact that the proposed formulation has much stronger linear relaxation bounds. Furthermore, while the quality of the linear relaxation of the proposed formulation increases accordingly to the size of the instance, the linear relaxation of the model in the literature decreases to zero, as shown in Table 2. In addition, the time necessary to solve the linear relaxations is smaller for the new model. The experiments also show that the solution provided by the heuristic can have a considerable influence in the solution process. Despite the good overall quality of solutions found by the heuristic, the model shows, as expected, a worse performance for poor quality heuristic solutions, even in the small instances, as the model size increases with the parameter T estimated by the heuristic. The heuristic solved all instances in less than one second. Despite the good results achieved by the new model when compared with other MIP based approaches, we can verify in Table 3, columns three and four, that the results obtained by the combinatorial B&B of Tanaka and Tierney (2018) was capable of solving all instances to optimality within the same time limit of one hour per instance. It is worthy to mention that the success of their method is due to the combination of fast lower bound calculation procedures and the many dominance rules applied for the selection of nodes during the exploration of the B&B tree.

4.2. Block Relocation Problems

Considering that the numerical results in the work of Petering and Hussein (2013) indicates that their BRP model outperforms the formulation presented by Caserta et al. (2012) both in terms of number instances solved and linear relaxation bounds, we decided to compare our BRP formulations with their model.

As for the PMP case, we also implemented this model and employed the same settings during our experiments in order to obtain fair comparisons. The sole modification applied with respect to the original model is the objective function. In our model, we minimize the number of relocations, while in their formulation, they minimize the time step in which the last container leaves the bay. Note that both models allow the objective function to be written in one of these two forms without the need of additional variables. We chose to write the objective function as the minimization of the number of relocations, as is done also by Caserta et al. (2012). We do not consider in our BRP comparisons the formulations for the R-BRP, e.g., Zehendner et al. (2015). As mentioned, the R-BRP is a special case of the BRP that incorporates more restrictive assumptions. These formulations can only be used to provide upper bounds for the BRP.

Tables 4, 5, A.7, and A.8 present the numerical results for the BRP formulations and also for the R-BRP B&B algorithm (Algorithm 4 in Appendix A) on Caserta and Voß (2009) instances. The results in Tables 4 and 5 are for the original non-grouped instances, and the results in Tables A.7 and A.8 (Appendix A) are for the transformed instances. As the models in the literature are not able to handle BRP instances with grouped priorities, Tables A.7 and A.8 show a comparison among the new BRP models and the R-BRP B&B only. As for the PMP experiments, we report the results for the linear relaxations and feasible/optimal solutions in separated tables, Tables 4 and A.7 and Tables 5 and A.8, respectively.

Considering the original instances of Caserta and Voß (2009), Table 5 shows that the new BRP_{m1} formulation and the one of Petering and Hussein (2013) have similar performance regarding the number of instances in which the linear relaxation is solved, nevertheless, BRP_{m1} model achieves consistently better lower bounds in average shorter execution times. BRP_{m2} model is in average faster than the other two formulations, however, it does not present good performances in both linear relaxation bounds and number of instances solved. On the other hand, analyzing the upper bounds results reported in Table 5 we see that BRP_{m2} model, despite having the worse linear relaxation, outperforms the other two formulations in terms of number of instances in which a feasible integer solution is found within the time limit, number of proved optimal solutions and average execution times. This is explained by the fact that it is more compact than the BRP_{m1} model, and thus its linear relaxations are solved faster, which allows the solver to explore more nodes in the B&B tree in less time. In summary, the two new BRP formulations are able to find feasible solutions and prove optimality in twice as many instances than the model in the literature.

Nevertheless, as discussed for the PMP case, also for the BRP the best exact algorithms in the literature are those based on combinatorial lower bounds coupled with the B&B framework, as we can verify in Table 5, columns three and four. The B&B algorithms proposed by Tricoire et al. (2018) and Tanaka and Mizuno (2018) were able to solve all the selected instances sets to optimality within reduce computational times. Among these two works, Tanaka and Mizuno (2018) present the best performance as they can solve all the instances from the even larger Caserta and Voß (2009) benchmarking sets: *data5–7* to *data5–10*, and *data6–6*, while Tricoire et al. (2018) can solve to optimality all the 40 instances of the sets up to *data5–6* only. As for their algorithm for the PMP, the success of the B&B method of Tanaka and Mizuno (2018) for the BRP is due to the combination of fast lower bound calculation procedures and the many dominance rules applied for the selection of nodes during the exploration of the B&B tree.

Tables A.7 and A.8 present the results for the two new BRP formulations and the R-BRP B&B in the instances of Caserta and Voß (2009) that were transformed to include grouped priorities. Differently from the unique priorities case, the models are able to better solve these instances, as indicated by the number of instances in which a feasible solution is found, the number of optimal solutions and the reduced computational effort. This behavior is due to the smaller linear programs being generated when the containers are distributed in fewer groups. At every three rows in Table A.8, we can observe that a decreasing number of instances is solved as the bay size grows and also when the number of container groups varies. Comparing to the R-BRP B&B, as for the unique priorities case, similar comments related to the quality of the solutions can be devised, however the proportion of solved instances by the models is much higher for the grouped priority instances. Similarly, despite BRP_{m1} presents much better linear relaxations, model BRP_{m2} is able to solve more instances to optimality within the specified time limit.

We report in Table 6 the results of the experiments with our R-BRP models and B&B algorithm (Algorithm 4) on the instances of Caserta and Voß (2009). As expected, due to its more restrictive assumptions, both methods were able to solve substantially more instances than our BRP models. The B&B algorithm solved all the selected instances in reduced computational times (less than 5 seconds on average). Similar to its BRP equivalents (BRP_{m1} and BRP_{m2}), R-BRP_{m1} provides better linear relaxation bounds than R-

BRP$_{m2}$. However, different from the BRP case, R-BRP$_{m1}$ was capable of solving almost 25% more instances to optimality than the R-BRP$_{m2}$ model. Furthermore, the lower bounds of R-BRP$_{m1}$ are tighter than those of BRP$_{m1}$. As for the BRP, the quality of these bounds decreases as the height of the stacks increases. When compared to the literature, the best linear programming based formulation for the R-BRP was proposed by Galle et al. (2018), and is followed by Zehendner et al. (2015), which is an improved version of the R-BRP model of Caserta et al. (2012). In both works, their implementation employs preprocessing mechanisms for eliminating variables. Despite we do not apply any type of preprocessing, our best R-BRP model still performs reasonably well, solving more than 75% of the instances. Improvements can be achieved by considering similar preprocessing mechanisms. However, as for the previous problems, up to this date the best exact algorithms for the R-BRP are based on the B&B framework. In fact, the best method for the R-BRP is the one proposed by Tanaka and Mizuno (2018), which is also the best for the BRP. Their algorithm is capable of solving all the instances on Table 6 to optimality in reduced computational times.

5. Concluding remarks

This paper studied the Pre-marshalling Problem (PMP) and the Block Relocation Problem (BRP). We developed a new integer programming model for the PMP in which the container priorities can be defined individually or for groups of containers. Based on the PMP formulation, we derived two new models for the BRP and its variant, the Restricted BRP (R-BRP). We also implemented a new greedy heuristic for the PMP and a branch-and-bound algorithm for the R-BRP. We evaluated the proposed formulations and algorithms on randomly generated instances available in the literature and also compared them with the best existing mathematical models for each problem. The computational experiments showed that our PMP formulation outperforms the mathematical model in the literature both in terms of linear relaxation quality and number of instances solved to optimality. The proposed model was able of solving five times more instances. Similar results were obtained for the two new BRP proposed formulations. Despite the proportion of solved instances was not as expressive as for the PMP case, we were still able to solve twice more instances than the model in the literature. Despite the good computational results obtained when strictly compared with other MIP based formulations, the models introduced in this work are not the best exact methods available for these problems. The best performance on the PMP, and both BRP variants have been achieved by approaches that incorporates fast combinatorial lower bound procedures within the B&B framework, as well as other computational improvements to speed up the search. The success of such methods is due to speed in which new lower bounds can be computed in each node of the B&B, normally they can be re-computed in constant time. Meanwhile, for MIP approaches, the linear relaxation of the models has to be computed in each node, and this can be a very time consuming task. Nevertheless, the performance of methods based on mathematical programming is expected to improve over time according new developments are incorporated into MIP solvers. Future research directions could include other relaxation techniques than the one based on linear programming, for example, column generation approaches. It would also be interesting to investigate these problems with new containers being allowed to enter the bay during the sorting and retrieval processes, or to consider more integrated contexts that combine stack sorting problems and crane scheduling and routing problems.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable and constructive comments that helped us to improve the quality of this paper.

Appendix A. Pseudocodes and additional results

Algorithm 1: Compute BRP lower bound.

```
1  Procedure BRP-LB($bay$)
2  $LB \leftarrow 0$;
3  for ($stack\_id \leftarrow 1, 2, \ldots, S$) do
4      $smaller\_Block \leftarrow bay[stack\_id][1]$;
5      for ($tier\_id \leftarrow 2, \ldots, height(bay[stack\_id])$) do
6          if $bay[stack\_id][tier\_id] \neq \emptyset$ then
7              if $bay[stack\_id][tier\_id] > smaller\_Block$ then
8                  $LB \leftarrow LB + 1$;
9              else
10                 $smaller\_Block \leftarrow bay[stack\_id][tier\_id]$;
11             end if
12         end if
13     end for
14 end for
15 return $LB$;
```

Algorithm 2: Transform (Caserta & Voß, 2009) instances into grouped BRP instances using original container information.

```
1  Procedure Instance_Converter($bay$, $G$)
2  for ($stack\_id \leftarrow 1, 2, \ldots, S$) do
3      for ($tier\_id \leftarrow 1, 2, \ldots, height(bay[stack\_id])$) do
4          if ($bay[stack\_id][tier\_id]$ mod $G = 0$) then
5              $bay[stack\_id][tier\_id] \leftarrow G$;
6          else
7              $bay[stack\_id][tier\_id] \leftarrow$
                   $(bay[stack\_id][tier\_id]$ mod $G)$;
8          end if
9      end for
10 end for
```

Algorithm 3: Heuristic algorithm for the PMP.

```
1  PMP_Heuristic(bay, stack_id)
2  relocations = 0;
3  while (there exists mis-overlays in bay) do
4      if (stack_id = ∅ or there is no mis-overlays in bay[stack_id]) then  // Step 1.
5          min_Amount = H;
6          for (s = 1, ..., S) do
7              if (height(bay[s]) < min_Amount and there is mis-overlays in bay[s]) then
8                  min_Amount = height(bay[s]);
9                  stack_id = s;
10             else
11                 if (bay[s] = min_Amount) then
12                     Select the stack holding the container with highest index;
13                 end if
14             end if
15         end for
16     end if
17     if (height(bay[stack_id]) > 0) then  // Step 2.
18         while (bay[stack_id] is not empty) do
19             Select a stack s' ∈ {1, ..., S}, s' ≠ stack_id, with available space and in which a relocation from stack does not create a deadlock;
20             If such stack does not exist, select s' with available space, unsorted stacks first;
21             Relocate the top container from s' to stack_id;
22             relocations = relocations + 1;
23         end while
24     else  // Step 3.
25         Build a list, sorted in non-increasing order, with all topmost containers;
26         while (height(bay[stack_id]) ≤ H − 2) do
27             Relocate the containers from the list to the emptied bay[stack_id];
28             Add the new topmost container, if it exists, in the list;
29             relocations = relocations + 1;
30         end while
31     end if
32     stack_id = ∅;
33 end while
34 return relocations;
```

Algorithm 4: Branch-and-bound algorithm for the R-BRP.

```
1  R-BRP_branch-and-bound(bay, previous_TargetBlock, blocks_Left, ub, lb, relocations, depth)
2  if (relocations + lb > ub) then
3      return; // Fathoming
4  end if
5  if (relocations < ub) and (blocks_Left = 0) then
6      ub = relocations;                    // Update best bound
7      Store incumbent solution;
8      return; // Fathoming
9  end if
10 if (blocks_Left > 0) then
11     if (there exists a stack s in bay with a target block available for retrieval) then
12         Retrieve the target block from s;
13         if (All blocks belonging to the current target group have been retrieved) then
14             previous_TargetBlock stores the current target group;
15             The target group is updated to the next group;
16         end if
17         R-BRP_BranchBound(bay, previous_TargetBlock, blocks_Left − 1, ub, lb, relocations, depth + 1);
18     else
19         for (s = 1, ..., S) do
               // Find a stack s in bay with an obstructed target block
20             if (stack s has a target block) then
21                 for (s' = 1, ..., S) do
                       // Find a stack s' in bay with available space
22                     if (s ≠ s') then
23                         Move an obstructing block from s to s';
24                         Update lb; // If no new mis-overlay is generated in s': lb = lb − 1.
25                         R-BRP_BranchBound(bay, previous_TargetBlock, blocks_Left, ub, lb, relocations + 1, depth + 1);
26                     end if
27                 end for
28             end if
29         end for
30     end if
31 end if
32 return;
```

Table A.7
Computational results concerning the linear relaxations of the proposed BRP models using converted instances of Caserta and Voß (2009).

Instance				BRP$_{m1}$			BRP$_{m2}$		
Name	G	S	H	lb	lb t. (second)	#lb	lb	lb t. (second)	#lb
data3–3–3	3	3	5	2.29	0.34	40	1.52	0.50	40
data3–3–4	4	3	5	2.50	0.59	40	1.44	0.91	40
data3–3–5	5	3	5	2.86	0.83	40	1.44	1.61	40
data3–4–3	3	4	5	2.48	1.08	40	1.85	1.22	40
data3–4–4	4	4	5	2.74	1.53	40	1.72	2.06	40
data3–4–5	5	4	5	2.80	2.48	40	1.59	3.83	40
data3–5–3	3	5	5	2.59	2.46	40	2.10	2.16	40
data3–5–4	4	5	5	2.72	3.87	40	1.72	5.22	40
data3–5–5	5	5	5	3.07	5.58	40	1.61	10.20	40
data3–6–3	3	6	5	2.58	5.23	40	2.18	4.06	40
data3–6–4	4	6	5	2.84	8.85	40	1.91	10.46	40
data3–6–5	5	6	5	3.24	15.34	40	1.51	24.88	40
data3–7–3	3	7	5	2.93	9.76	40	2.36	10.00	40
data3–7–4	4	7	5	3.07	15.58	40	2.12	21.71	40
data3–7–5	5	7	5	3.33	29.36	40	1.63	54.55	40
data3–8–3	3	8	5	2.92	14.57	40	2.33	14.15	40
data3–8–4	4	8	5	3.54	39.85	40	1.84	65.79	40
data3–8–5	5	8	5	3.55	52.26	40	1.54	105.01	40
data4–4–3	3	4	6	3.01	3.92	40	2.14	4.32	40
data4–4–4	4	4	6	3.47	6.23	40	1.66	9.12	40
data4–4–5	5	4	6	3.53	12.39	40	1.28	22.72	40
data4–5–3	3	5	6	3.33	10.35	40	2.13	10.10	40
data4–5–4	4	5	6	3.74	22.74	40	1.44	30.69	40
data4–5–5	5	5	6	3.91	29.82	40	1.37	60.68	40
data4–6–3	3	6	6	3.54	23.06	40	2.11	29.50	40
data4–6–4	4	6	6	3.99	50.86	40	1.64	78.43	40
data4–6–5	5	6	6	4.10	74.25	40	1.29	171.27	40
data4–7–3	3	7	6	3.74	49.76	40	2.13	53.62	40
data4–7–4	4	7	6	4.35	132.10	40	1.38	308.72	40
data4–7–5	5	7	6	4.42	213.34	40	1.02	627.54	39
data5–4–3	3	4	7	3.72	12.05	40	1.92	12.17	40
data5–4–4	4	4	7	4.03	23.55	40	1.31	31.24	40
data5–4–5	5	4	7	4.68	43.86	40	0.97	85.72	40
data5–5–3	3	5	7	3.98	35.91	40	1.83	34.28	40
data5–5–4	4	5	7	4.32	65.67	40	1.40	93.16	40
data5–5–5	5	5	7	4.88	133.36	40	0.92	317.78	40
data5–6–3	3	6	7	4.38	78.02	40	1.85	97.63	40
data5–6–4	4	6	7	4.80	200.49	40	0.98	400.06	40
data5–6–5	5	6	7	5.40	399.47	40	0.62	1375.28	39
Total						1560			1558

Table A.8
Computational results for the proposed BRP models using converted instances of Caserta and Voß (2009).

Instance	R-BRP	BRP$_{m1}$						BRP$_{m2}$					
	Opt	lb*	Feas.	#feas	Opt	Opt t. (second)	#opt	lb*	Feas.	#feas	Opt	Opt t. (second)	#opt
data3–3–3	3.25	2.29	3.23	40	3.23	1.34	40	1.52	3.23	40	3.23	1.34	40
data3–3–4	3.78	2.50	3.75	40	3.75	1.50	40	1.44	3.75	40	3.75	4.84	40
data3–3–5	4.10	2.86	4.10	40	4.10	2.53	40	1.44	4.10	40	4.10	7.53	40
data3–4–3	3.85	2.48	3.85	40	3.85	6.96	40	1.85	3.85	40	3.85	6.16	40
data3–4–4	4.43	2.74	4.43	40	4.43	9.39	40	1.72	4.43	40	4.43	18.33	40
data3–4–5	4.93	2.80	4.90	40	4.90	51.34	40	1.59	4.90	40	4.90	71.51	40
data3–5–3	4.58	2.59	4.58	40	4.58	50.01	40	2.10	4.58	40	4.58	25.12	40
data3–5–4	4.95	2.63	4.95	40	4.68	244.98	38	1.71	4.95	40	4.59	147.47	37
data3–5–5	5.88	3.03	5.83	40	5.72	255.70	39	1.63	5.85	40	5.63	531.70	38
data3–6–3	5.15	2.44	5.15	40	4.66	250.90	35	2.19	5.15	40	4.92	155.32	38
data3–6–4	6.15	2.67	6.15	40	5.45	507.65	29	1.92	6.15	40	5.89	405.24	37
data3–6–5	7.15	2.97	7.15	40	6.04	1024.83	26	1.73	7.13	40	6.07	578.10	27
data3–7–3	6.55	2.59	6.55	40	5.32	734.80	25	2.40	6.55	40	6.00	400.16	33
data3–7–4	6.93	2.75	6.78	37	5.58	891.08	19	2.25	6.93	40	6.21	492.31	29
data3–7–5	7.63	2.74	7.03	32	5.14	475.74	14	1.90	7.62	39	5.61	615.82	18
data3–8–3	6.55	2.56	6.44	39	4.94	754.12	18	2.29	6.55	40	6.03	376.92	33

(*continued on next page*)

Table A.8 (*continued*)

Instance	R-BRP	BRP$_{m1}$						BRP$_{m2}$					
	Opt	lb*	Feas.	#feas	Opt	Opt *t*. (second)	#opt	lb*	Feas.	#feas	Opt	Opt *t*. (second)	#opt
data3–8–4	8.40	2.75	8.06	35	5.67	889.68	9	1.98	8.50	40	6.68	926.39	19
data3–8–5	8.20	3.14	7.94	34	5.71	998.37	7	1.87	8.15	39	6.56	1149.85	16
data4–4–3	6.45	2.96	6.43	40	6.21	381.84	38	2.17	6.43	40	6.21	223.01	38
data4–4–4	7.65	3.40	7.58	40	7.25	394.66	36	1.71	7.58	40	7.25	613.47	36
data4–4–5	8.73	3.31	8.30	37	7.57	684.82	28	1.49	8.50	38	7.29	567.00	24
data4–5–3	7.83	2.85	7.39	36	6.14	1066.43	21	2.29	7.83	40	6.87	509.06	31
data4–5–4	9.33	3.18	8.00	27	6.88	1097.80	16	1.85	9.48	40	7.10	645.75	20
data4–5–5	9.30	3.38	8.66	29	7.17	1019.81	12	1.65	9.33	39	7.07	1213.38	14
data4–6–3	9.33	2.82	8.56	32	6.17	754.87	6	2.21	9.11	38	7.35	617.46	20
data4–6–4	10.18	3.44	8.89	19	6.50	1312.83	4	2.08	9.89	35	7.25	891.29	8
data4–6–5	10.30	3.24	7.94	16	6.60	2294.44	5	1.77	9.50	30	7.33	1214.30	12
data4–7–3	10.30	2.57	8.74	23	5.00	361.58	2	2.78	10.11	38	7.36	1011.14	11
data4–7–4	12.10	–	11.00	2	–	–	0	–	12.13	32	–	–	0
data4–7–5	12.18	3.63	9.14	7	7.00	3365.66	1	1.48	10.53	19	7.00	350.53	1
data5–4–3	9.65	3.43	9.15	33	8.14	1114.44	21	2.15	9.75	40	8.35	552.88	26
data5–4–4	10.35	3.82	9.11	27	7.86	1070.52	14	1.69	9.53	34	7.82	815.30	17
data5–4–5	11.95	3.74	9.94	18	8.13	1075.42	8	1.91	10.89	27	7.17	611.12	6
data5–5–3	11.05	2.90	9.00	22	6.50	1484.84	4	2.26	10.78	36	8.19	973.33	16
data5–5–4	12.03	2.98	9.17	6	6.00	915.13	1	2.26	11.22	23	8.60	1067.46	5
data5–5–5	13.55	–	11.14	7	–	–	0	1.71	11.31	13	7.00	864.40	1
data5–6–3	12.68	2.96	9.91	11	7.00	2730.50	1	2.72	11.87	31	7.50	535.34	2
data5–6–4	14.55	–	10.50	2	–	–	0	2.74	13.60	15	7.00	1249.35	1
data5–6–5	15.63	–	10.50	2	–	–	0	–	13.20	5	–	–	0
Total				1133			757			1371			894

References

Bortfeldt, A., & Forster, F. (2012). A tree search procedure for the container pre-marshalling problem. *European Journal of Operational Research, 217*(3), 53–540.

van Brink, M., & van der Zwaan, R. (2014). *(September 8–10, 2014). Proceedings of the twenty-second annual European symposium algorithms, ESA 2014, Wroclaw, Poland* (pp. 798–809). Berlin, Heidelberg: Springer Berlin Heidelberg.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014a). Storage yard operations in container terminals: literature overview, trends, and research directions. *European Journal of Operational Research, 235*(2), 412–430.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014b). Transport operations in container terminals: literature overview, trends, research directions and classification scheme. *European Journal of Operational Research, 236*(1), 1–13.

Carlo, H. J., Vis, I. F. A., & Roodbergen, K. J. (2013). Seaside operations in container terminals: literature overview, trends, and research directions. *Flexible Services and Manufacturing Journal, 27*(2), 224–262.

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* (pp. 37–48). Berlin, Heidelberg: Springer Berlin Heidelberg.

Caserta, M., Schwarze, S., & Voß, S. (2011). Container rehandling at maritime container terminals. In J. W. Böse (Ed.), *Handbook of terminal planning* (pp. 247–269). Springer New York (vol. 49). Operations Research/Computer Science Interfaces Series.

Caserta, M., Schwarze, S., & Voß, S. (2012). A mathematical formulation and complexity considerations for the blocks relocation problem. *European Journal of Operational Research, 219*(1), 96–104.

Caserta, M., & Voß, S. (2009). A corridor method-based algorithm for the pre–marshalling problem. In M. Giacobini, A. Brabazon, S. Cagnoni, G. A. Di Caro, A. Ekárt, A. I. Esparcia-Alcázar, M. Farooq, A. Fink, & P. Machado (Eds.), *Applications of evolutionary computing* (pp. 788–797). Springer Berlin Heidelberg. vol. (5484). Lecture Notes in Computer Science.

Caserta, M., Voß, S., & Sniedovich, M. (2011). Applying the corridor method to a blocks relocation problem. *OR Spectrum, 33*(4), 915–929.

Dekker, R., Voogd, P., & van Asperen, E. (2006). Advanced methods for container stacking. *OR Spectrum, 28*(4), 563–586.

Expósito-Izquierdo, C., Melián-Batista, B., & Moreno-Vega, J. M. (2012). Pre-marshalling problem: Heuristic solution method and instances generator. *Expert Systems with Applications, 39*(9), 8337–8349.

Expósito-Izquierdo, C., Melián-Batista, B., & Moreno-Vega, J. M. (2014). A domain-specific knowledge-based heuristic for the Blocks Relocation Problem. *Advanced Engineering Informatics, 28*(4), 327–343.

Expósito-Izquierdo, C., Melián-Batista, B., & Moreno-Vega, J. M. (2015). An exact approach for the blocks relocation problem. *Expert Systems with Applications, 42*(1718), 6408–6422.

Galle, V., Barnhart, C., & Jaillet, P. (2018). A new binary formulation of the restricted container relocation problem based on a binary encoding of configurations. *European Journal of Operational Research, 267*(2), 467–477.

Jovanovic, R., Tuba, M., & Voß, S. (2015). A multi-heuristic approach for solving the pre-marshalling problem. *Central European Journal of Operations Research*, 1–28.

Jovanovic, R., & Voß, S. (2014). A chain heuristic for the blocks relocation problem. *Computers & Industrial Engineering, 75*, 79–86.

Kim, K. H., & Hong, G.-P. (2006). A heuristic rule for relocating blocks. *Computers & Operations Research, 33*(4), 940–954.

Kim, K. H., & Kim, H. B. (1999). Segregating space allocation models for container inventories in port container terminals. *International Journal of Production Economics, 59*(1–3), 415–423.

Kim, K. H., & Park, K. T. (2003). A note on a dynamic space-allocation method for outbound containers. *European Journal of Operational Research, 148*(1), 92–101.

Ku, D., & Arthanari, T. S. (2016). On the abstraction method for the container relocation problem. *Computers & Operations Research, 68*, 110–122.

Lee, Y., & Chao, S.-L. (2009). A neighborhood search heuristic for pre-marshalling export containers. *European Journal of Operational Research, 196*(2), 468–475.

Lee, Y., & Hsu, N.-Y. (2007). An optimization model for the container pre-marshalling problem. *Computers & Operations Research, 34*(11), 3295–3313.

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

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*(1), 120–130.

Steenken, D., Voß, S., & Stahlbock, R. (2004). Container terminal operation and operations research – a classification and literature review. *OR Spectrum, 26*(1), 3–49.

Tanaka, S., & Mizuno, F. (2018). An exact algorithm for the unrestricted block relocation problem. *Computers & Operations Research, 95*, 12–31.

Tanaka, S., & Takii, K. (2016). A Faster Branch-and-Bound Algorithm for the Block Relocation Problem. *IEEE Transactions on Automation Science and Engineering, 13*(1), 181–190.

Tanaka, S., & Tierney, K. (2018). Solving real-world sized container pre-marshalling problems with an iterative deepening branch-and-bound algorithm. *European Journal of Operational Research, 264*(1), 165–180.

Tierney, K., Pacino, D., & Voß, S. (2016). Solving the pre-marshalling problem to optimality with A* and IDA*. *Flexible Services and Manufacturing Journal*, 1–37.

Tricoire, F., Scagnetti, J., & Beham, A. (2018). New insights on the block relocation problem. *Computers & Operations Research, 89*(Supplement C), 127–139.

United Nations (2015). Review of maritime transport 2015. In *Proceedings of the united nations conference on trade and development (UNCTAD)*. New York, Geneva: United Nations Publication.

United Nations (2016). Review of maritime transport 2016. In *Proceedings of the united nations conference on trade and development (UNCTAD)*. New York, Geneva: United Nations Publication.

Ünlüyurt, T., & Aydin, C. (2012). Improved rehandling strategies for the container retrieval process. *Journal of Advanced Transportation, 46*(4), 378–393.

Vis, I. F. A., & Roodbergen, K. J. (2009). Scheduling of container storage and retrieval. *Operations Research, 57*(2), 456–467.

Wu, K.-C., Ting, C.-J., & Hernández, R. (2010). Applying tabu search for minimizing reshuffle operations at container yards. *Journal of the Eastern Asia Society for Transportation Studies, 8*, 2379–2393.

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.

Zhu, W., Qin, H., Lim, A., & Zhang, H. (2012). Iterative deepening A* algorithms for the container relocation problem. *IEEE Transactions on Automation Science and Engineering, 9*(4), 710–722.