A Survey of Optimization Methods for Block Relocation and PreMarshalling Problems

Abstract

Problems of stacking items or goods occur in a plenty of applications, such as container terminals, warehouses, and steel plants. A lot of research has been done in this area in recent years. In this survey, we provide an overview of the Block Relocation Problem (BRP) and the PreMarshalling Problem (PMP) and review optimization methods for these two problems. Due to common properties between these problems, solutions and methods applicable to one problem can often be applied to the other, and vice-versa. We distinguish four categories of optimization methods and propose future directions for each of them.

Keywords: optimization methods, block relocation, premarshalling, container stacking

1. Introduction

The storage of items in a limited area arises in many applications, such as container terminals, warehouses, and steel plants. For instance, container terminals handle the storage of containers when transferring them among different types of ships, trains, and trucks. With growing vessel sizes, shipping companies are more demanding than ever. In 2018, world seaborne trade volumes rose to 11 billion tons [107]. Warehouses temporarily store items such as parcels or steel plates before their retrieval. Estimated 87 billion parcels were shipped worldwide in 2018 according to Bowes [12]. In 2018, the world crude steel production reached 1.8 billion tons [114]. According to Świeboda and Zając [89], avoiding unproductive moves of containers can reduce fuel consumption by 40 %. Therefore, with the growing demand of the last decades, improving logistics performance in the industry has become particularly important and challenging. This paper provides a comprehensive review of the recent methods found in the literature for tackling two popular stacking problems, the *Block Relocation Problem* and the *PreMarshalling Problem*.

For the sake of clarity, we specify the terms employed in this paper. A *storage area* can be a container ship, a warehouse, a yard, or a train depot. A storage area is assumed to be arranged as *stacks* (or columns). Stacks contain *items* piled up on top of each other. By default, we assume that a *crane* operates the items, so each stack is accessed from the top only. A stack may have a *maximum height* or *capacity*, i.e. the maximum number of *slots* in the stack. An arrangement of items in the storage area is called a *layout* (also called configuration in the literature). Items are cuboids such as containers, blocks, steel plates, or parcels. This paper does not consider round items such as rolls or coils. Items that arrive at a storage area are called *incoming* items, whereas the ones that leave the storage area are *outgoing* items. Items may have a known *arrival time* and *departure time* (also called due time, retrieval time or priority). Let us define three categories of *moves*. When an item arrives at the storage area, we call it a *placement*. When an item leaves the storage area, we call it a *retrieval*. A *relocation* happens when we move an item from a stack to another. In the literature, a relocation might also be called reshuffling or rehandling. In problems where items must be retrieved in a specific order, we call the current item to retrieve the *target item*. When the target item is not located at the top of a stack, all the items above it must be relocated. Such relocations are called *forced relocations* because they cannot be avoided. Otherwise, *voluntary moves* (also called cleaning or anticipatory moves) consist in relocating arbitrary items.

Table 1: Abbreviations for solution methods

Mathematical formulations	
(M)IP	(Mixed) Integer Programming
CP	Constraint Programming
DP	Dynamic Programming
SDP	Stochastic Dynamic Programming
Metaheuristics	
ACO	Ant Colony Optimization
CM	Corridor method
GA	Genetic algorithm
GRASP	Greedy randomized adaptive search procedure
PM	Pilot Method
SA	Simulated annealing
Tree Search based	
A^*	A^* search
BS	Beam Search
B&B	Branch & Bound
B&P	Branch & Price
B&C	Branch & Cut
DT	Decision Trees
ID-A^*	Iterative Deepening A^*
ID-B&B	Iterative Deepening B&B
RS	Rake Search
TS	Tree Search

In many applications, departure times of items are unknown when items are loaded into the storage area. In this case, operators cannot guarantee that items are arranged in their order of departure, so relocations may be necessary afterward. When the departure times are revealed, two approaches are typically considered. Premarshalling arises when items can be rearranged before unloading. In this approach, items are relocated inside the storage area in such a way that items can be retrieved afterward without additional relocations. The *PreMarshalling Problem* (PMP) aims at finding a shortest sequence of moves achieving this goal. When items cannot be rearranged before retrievals, operators have to consider relocations and retrievals simultaneously. The *Block Relocation Problem* (BRP) consists in finding a shortest sequence of moves to retrieve items in the given departure order.

As suggested in a comprehensive survey [69], stacking problems can be distinguished into three classes: loading, premarshalling, and unloading problems. Problems can also belong to a combination of these classes, e.g. loading/unloading problems. A closely related survey [25] (based on [24]), classifies stacking problems into *storage problems*, *re-handling problems*, and *retrieval problems*. Storage problems aim at choosing a best placement for incoming items. Re-handling problems aim at continuously handling both incoming and outgoing items. Finally, the pure BRP and PMP fall into the last class, i.e. retrieval problems, where incoming items are not allowed. In the literature, we observed that authors often applied similar methods to distinct classes of stacking problems. This is possible because these problems often share common structures and common decision types. We recall that the BRP consists in unloading items from the storage area in a given order with a shortest sequence of moves. During the retrieval process, the decision-maker may need to relocate items blocking a target item. This type of decision is also in the core of the PMP. For example, Caserta and Voß [21] adapt to the PMP a method developed in [22] for the BRP. It can also be observed that numerous heuristics can be applied each time an item needs to be placed or relocated, regardless of the type of problem. Our paper gives a method-centric view of the literature and aims at helping researchers and engineers in the development of advanced methods for solving a wide range of stacking problems. For this purpose, we provide a classification of optimization methods in four distinct categories:

- **Mathematical formulations**, described in Section 3, including Integer Programming and Constraint Programming models, as well as Dynamic Programming.
- **Heuristics**, described in Section 4.
- **Metaheuristics**, described in Section 5, including methods such as Genetic Algorithms and Simulated Annealing.
- **Tree search-based methods**, described in Section 6, including methods based on the exploration of a tree, such as Branch & Bound, Beam Search and A*.

Table 1 summarizes the abbreviations used to name the optimization methods in this paper.

Various papers present overviews of related problems and topics, e.g. container rehandling [20], container terminals [88, 73], container stowage metrics [46], container loading [11], crane scheduling [14, 13, 61], container ship stowage planning [126], ship loading problem [50], train shunting [43], storage yard operations [16], transport operations [17].

The rest of this paper is structured as follows. In Section 2, we review stacking problems studied in the literature related to the Block Relocation Problem and the PreMarshalling

Problem. In Sections 3 to 6, we review methods developed during recent years. Finally, Section 7 provides concluding remarks and discusses related future directions.

2. Problem definitions

According to a previous comprehensive survey [69], stacking problems can be classified into three main categories. These categories correspond to three stacking processes. In *loading problems*, one has to store items arriving in the storage area. In *premarshalling problems*, one has to rearrange items already placed in the storage area to satisfy an objective such as being able to retrieve all the items without relocations. In *unloading problems*, one has to retrieve outgoing items from the storage area, e.g. by determining a sequence of moves that minimizes the number of relocations. Finally, these categories may be combined to describe problems covering multiple stacking processes. For example, in a *combined loading/unloading problem*, one has to store incoming items and retrieve outgoing items, simultaneously.

Lehnfeld and Knust [69] propose a three-field notation to identify problems and their characteristics. Although this notation has several advantages, problems encountered in this paper can also be considered as variants of the Block Relocation Problem or the Pre-Marshalling Problem. For the sake of brevity, this survey does not cover other stacking problems, such as loading and combined loading/unloading problems. We refer to [69] for an overview of those.

2.1. Block Relocation Problem

The **Block Relocation Problem (BRP)** [59], also called **Container Relocation Problem (CRP)**, is certainly the most studied problem presented in this paper. Items, already located in the storage area, have predefined priorities or departure times. The objective of the classic BRP is to retrieve all the items with respect to their departure times, with a minimum number of relocations.

Figure 1 illustrates an example of a layout with 9 items placed in 3 stacks without height limit. Items are indexed by retrieval time and must be retrieved in the order 1, 2, . . . , 9. In the first turn, item 4 must be relocated to access the (shaded) target item 1. After items 1 and 2 have been retrieved, items 8 and 6 need to be relocated to access item 3. Afterward, items 3, 4, 5, and 6 can be retrieved. Then, a relocation of item 9 is necessary to remove item 7. Finally, a total of four relocations are required to empty the layout.

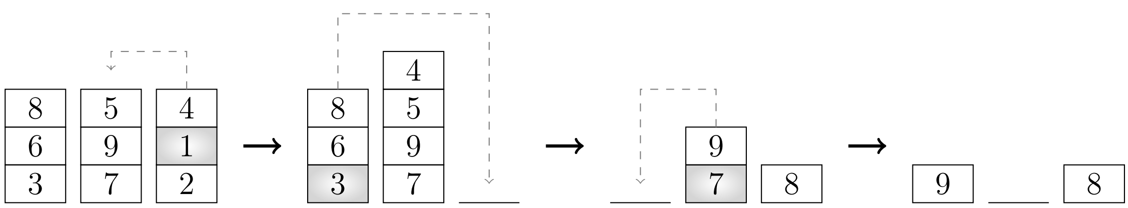

Figure 1: Example of solution for the Block Relocation Problem

Let us discuss several common variants of the BRP. Under the **restricted BRP (rBRP)**, only items blocking the current target item can be relocated. On the other hand, the **unrestricted BRP (uBRP)** allows relocating any item, including voluntary moves. Note that the unrestricted BRP can lead to fewer relocations, at the cost of a significantly larger search space. Another characteristic of the BRPs to consider is the retrieval order of items. Under the BRP with **distinct priorities**, the retrieval order is described as a sequence and cannot be changed. In contrast, under the BRP with **duplicate priorities**, items are partitioned into groups in which all items have the same priority. Thus, items belonging to the same group can be retrieved in an arbitrary order.

Some authors optimize alternative objectives, such as the *crane working time/distance* [108, 85, 86], or *waiting times* [9, 71, 35]. The retrieval order of items may be partially unknown and revealed during the unloading process, as in the **Stochastic BRP** [42] where the objective is to minimize the *expected number of relocations* [83, 128, 35]. Departure times of items may lie within a time window [63]. The **Slab Stack Shuffling (SSS)** problem assumes that items belong to predefined families with given priorities. These families may be disjoint [99], or may overlap [100]. In the SSS, one item has to be retrieved per family with respect to the family priorities. In the **BRP with Stowage Plan (BRP-SP)** [51, 54], items must be put in a destination storage area with designated slots in which items cannot be relocated. In the **Block Retrieval Problem (BRTP)** [83], only a subset of the items must be retrieved, in any order. The BRP usually assumes that only retrievals and relocations occur. The dynamic version of the BRP, the **Dynamic Container Relocation Problem (DCRP)** [1], considers the arrival of items during the unloading process.

Both the restricted BRP and the unrestricted BRP have been proven NP-hard [19]. Naturally, all the problems having the BRP as a particular case, such as the BRP-SP, are also NP-hard. When the objective is to minimize the crane working time, the BRP remains NP-hard [110]. Papers related to the BRP are listed in Table 2.

Table 2: References for the Block Relocation Problem

Reference	Restricted BRP	Unrestricted BRP	Duplicate priorities	Alternative objectives	Uncertainty	Methods	Notes
Azab and Morita [2]		✓	✓			IP	with appointment schedul.
Bacci et al. [3]	✓					-	
Bacci et al. [4]	✓					BS	
Bacci et al. [5]	✓					B&C IP	

Continued on next page

Table 2 (continued)

Reference	Restricted BRP	Unrestricted BRP	Duplicate priorities	Alternative objectives	Uncertainty	Methods	Notes
Borjian et al. [8]	✓	✓			✓	A*	
Borjian et al. [9]		✓	✓	✓		IP	with service times
Caserta and Voß [22]	✓					CM	
Caserta et al. [18]	✓					H	
Caserta et al. [23]	✓					CM	
Caserta et al. [19]	✓	✓				H IP	
de Melo da Silva et al. [84]	✓	✓	✓			IP	
ElWakil et al. [29]	✓					SA	
Eskandari and Azari [30]	✓					IP	
Expósito-Izquierdo et al. [31]	✓	✓				A* H	
Expósito-Izquierdo et al. [32]	✓					B&B IP	
Feillet et al. [34]		✓				H	
Feng et al. [35]	✓			✓	✓	DT H SDP	with service times
da Silva Firmino et al. [85]	✓			✓		A*	min crane workload
da Silva Firmino et al. [86]	✓			✓		GRASP	min crane workload
Forster and Bortfeldt [38]		✓	✓			H TS	
Galle et al. [41]	✓	✓				H	
Galle et al. [42]	✓				✓	DT H	
Galle et al. [40]	✓					IP	
Ji et al. [51]	✓		✓			GA H IP	multi-crane with stowage plan
Jin et al. [53]	✓		✓			H	
Jovanovic and Voß [57]	✓					H	
Jovanovic et al. [54]	✓		✓			GRASP H	with stowage plan
Jovanovic et al. [56]	✓	✓		✓		ACO	min crane workload with stowage plan
Kim and Hong [59]	✓		✓			B&B H	
Kim et al. [60]		✓				H	
Ku and Arthanari [64]	✓					TS	
Ku and Arthanari [63]	✓		✓			H SDP TS	with time windows

Continued on next page

Table 2 (continued)

Reference	Restricted BRP	Unrestricted BRP	Duplicate priorities	Alternative objectives	Uncertainty	Methods	Notes
Lee and Lee [68]	✓			✓		H IP	min crane workload
Lin et al. [70]		✓	✓	✓		H	min crane workload , multi-lift crane
López-Plata et al. [71]		✓		✓		H IP	with waiting times
Lu et al. [72]	✓	✓	✓	✓		IP	min crane workload with batch moves
Olsen and Gross [75]	✓					H	
Petering and Hussein [78]		✓				H IP	
Quispe et al. [80]	✓					B&B ID-A*	
de Melo da Silva et al. [83]	✓		✓	✓	✓	B&B BS	block retrieval problem
Tanaka and Takii [93]	✓		✓			B&B	
Tanaka and Mizuno [92]	✓	✓				B&B	
Tanaka and Voß [96]	✓	✓	✓			ID-B&B	with stowage plan
Tanaka and Voß [97]	✓					IP	
Tang et al. [99]	✓		✓			GA IP	slab stack shuffling
Tang and Ren [100]	✓		✓			CP DP H	slab stack shuffling
Tang et al. [101]	✓			✓		H IP Tabu	min crane workload
Tang et al. [98]	✓					H IP	
Ting and Wu [104]	✓					BS H	
Tricoire et al. [105]		✓				B&B H PM	
Ünlüyurt and Aydın [108]	✓			✓		B&B H	min crane workload
Voß and Schwarze [110]	✓			✓		IP	min crane workload
Wan et al. [111]	✓					H IP	
Wu and Ting [116]	✓					BS H	
Zehendner and Feillet [118]	✓		✓			B&P IP	
Zehendner and Feillet [119]	✓		✓			B&P IP	
Zehendner et al. [117]	✓					IP	
Zehendner et al. [120]	✓					H	
Zeng et al. [121]	✓		✓			H IP	
Zhang et al. [125]		✓				H TS	with batch moves

Continued on next page

Table 2 (continued)

Reference	Restricted BRP	Unrestricted BRP	Duplicate priorities	Alternative objectives	Uncertainty	Methods	Notes
Zhang et al. [123]	✓					B&B BS	
Zhu et al. [127]	✓	✓				H ID-A*	
Zweers et al. [128]	✓		✓		✓	B&B H	with time windows

Computational experiments in the literature for the BRP can be conducted on the following datasets, available online.

- CVS [21, 19]: 840 instances with 3 to 10 stacks, filled with 9 to 100 items. Available in [45, 91].
- ZQLZ [127]: 12,500 instances with 6 to 10 stacks, filled with 15 to 69 items. Available in [91].
- UA [108]: 9,600 instances with 3 to 7 stacks, filled with 6 to 39 items. Available in [26].
- GBMBJ [42]: instances for the stochastic BRP. Available in [39].
- KA [63]: instances for the BRP with time windows. Available in [62].
- JTNV [54]: instances for the BRP with stowage plan. Available in [58].

2.2. PreMarshalling Problem

Another way of reducing the time required for unloading a storage area is to rearrange the layout before the first retrieval. We call an item *misplaced* if it is located above an item of higher priority or another misplaced item. The goal of the **PreMarshalling Problem (PMP)** [67] is to "clean" a given layout, i.e. to reorganize its items in such a way that no item is misplaced, while retrievals are forbidden. Unless specified, we assume that the storage area has a maximum stack height and no dummy stack. The most common objective function is to minimize the number of relocations.

Figure 2 illustrates an example of a solution for the PMP on a layout with 9 items placed in 3 stacks without height limit. Items are indexed according to their retrieval order. One has to relocate items in such a way that they can be retrieved in the given order 1, 2, . . . , 9, without additional relocation. In the first step, items 4 and 8 are misplaced, because they are blocking the way of items that need to be retrieved earlier. We can decide to relocate items 4 and 8 above item 1, so we can put item 2 on top of item 3. Finally, items 8 and 4 can be put above item 9. This example requires five moves to clean the layout.

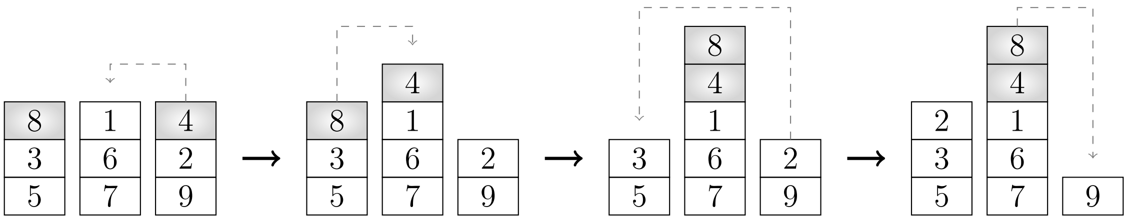

Figure 2: Example of solution for the PreMarshalling Problem

In the following, we discuss variants of the PMP. The **PMP with target layout** [67, 84] requires that every item is located to a predefined slot. Other variants impose that items are moved to specified stacks instead of slots [27], or that each stack contains only one item type [67]. In a problem tackled in [49], items belong to groups that must be relocated to dedicated locations. Rendl and Prandtstetter [82], Boge et al. [6] study the PMP where the retrieval order is uncertain. Minimizing the crane working time has been considered as an alternative objective function [77]. The **2-Dimensional PMP (2D-PMP)** [106] aims at making items retrievable using a reach stacker, i.e. items must be well ordered vertically as well as horizontally.

The pure PMP has been proven NP-hard, even when the maximum stack height b is fixed, for any $b \geq 6$ [15]. However, the question of deciding whether the pure PMP has a feasible solution can be solved in $\mathcal{O}(n)$ [6]. Table 3 lists the references related to the PMP.

Numerous datasets are available for computational experiments on PMP instances, which are available in [65, 90].

- CV [21, 19]: 840 instances with 3 to 10 stacks, filled with 9 to 100 items.
- BF [10]: 640 instances with 16 to 20 stacks, filled with 48 to 128 items.
- BZ [15]: 960 instances with 3 to 9 stacks, filled with 6 to 37 items.
- EMM [33]: two sets of respectively 1200 instances [65] and 950 instances [102], with 4 to 10 stacks, filled with 8 to 40 items.
- LC [66]: 41 instances, with 10 to 12 stacks, filled with 35 to 54 items.
- ZJY [124]: 100 instances with 6 to 9 stacks, filled with 17 to 25 items.

3. Mathematical formulations

Mathematical formulations are a convenient way to formally describe problems and to compute optimal solutions. Although *Integer Programming* (IP) models are not yet suitable

Table 3: References for the PreMarshalling Problem

Reference	Methods	Notes
Boge et al. [6]	IP	uncertain priority classes
Bortfeldt and Forster [10]	TS	
Caserta and Voß [21]	CM	
de Melo da Silva et al. [84]	IP	
Expósito-Izquierdo et al. [33]	H	
Gheith et al. [44]	GA	
Hottung and Tierney [48]	GA	
Hottung et al. [47]	TS	
Huang and Lin [49]	H	with target locations
Jin and Yu [52]	ID-B&B	
Jovanovic et al. [55]	H	
Lee and Hsu [67]	H IP	with target locations
Lee and Chao [66]	H	
Parreño-Torres et al. [76]	IP	
Parreño-Torres et al. [77]	B&B IP	minimize crane time
Prandtstetter [79]	B&B DP H	
Rendl and Prandtstetter [82]	CP H	uncertain retrieval times
Tanaka and Tierney [94]	ID-B&B	
Tanaka et al. [95]	B&B	
Tierney et al. [102]	A* ID-A*	
Tierney and Voß [103]	ID-A*	uncertain retrieval times
Tus et al. [106]	ACO PM	2D-PMP
van Brink and van der Zwaan [15]	B&P IP	
Voß [109]	-	
Wang et al. [112]	BS H	with/without buffer stack
Wang et al. [113]	H	
Zhang et al. [124]	B&B H	

for solving large instances, there have been significant improvements in recent years. Some authors also designed *Constraint Programming* (CP) models. Both IP and CP models can be handled by free and commercial solvers whose performance improves over time. We also include *Dynamic Programming* (DP) models in this section. The following review of models is split into four parts, each focusing on one class of stacking problems: loading, premarshalling, unloading, and combined problems.

3.1. Formulations for the Block Relocation Problem

Wan et al. [111] introduce the first IP-based method, named MRIP, for the restricted BRP. Their formulation assumes that at each retrieval, relocated items can be moved at most once. Due to a large number of variables and long computational times, they describe a heuristic based on MRIP. Instead of considering the retrieval of all items in a single model, they solve a series of reduced MRIP_K models considering the retrieval of the next K items only. After the retrieval of a single item, a new MRIP_K model is considered, and the process goes on. The MRIP-based heuristic reduced the number of relocations compared to the LS [122], RI [74] and ENAR [59] heuristics (described in Section 4), but computational times were longer. Wan et al. [111] also applied the MRIP-based method when new items arrive at the storage area during the unloading process. Tang et al. [101] follow the structure of MRIP to formulate an IP model optimizing a weighted sum of the number of relocations and the crane traveling distance. They also model the scenario where the shape of items is round. Tang et al. [98] improve the MRIP model (we call it MRIP2) by removing variables and adding constraints. MRIP2 took shorter computational times (one-third of MRIP for the largest instances) and could solve more large instances than MRIP.

Two of the most cited IP models are in [19] for the BRP. The first model, BRP-I, solves the unrestricted BRP. The time horizon is discretized into periods during which a single move (relocation or retrieval) can be performed. The variables are distinguished into two sets, one for defining feasible layouts, and one for defining feasible moves. A drawback of the BRP-I model is that the user has to provide an upper bound of the number of moves. The second model, named as BRP-II, forbids voluntary moves. Thus, it does not require an upper bound of the number of moves and significantly reduces the feasible region, thus can solve larger instances.

Several authors [32, 30, 117] found issues in the BRP-II model for the restricted BRP such as erroneous constraints. Expósito-Izquierdo et al. [32] introduce a corrected model BRP-II*. Also, Eskandari and Azari [30] add several valid inequalities to their corrected model BRP2ci, resulting in computational times decreased by a factor of 25. Zehendner et al. [117] also describe an improved model called BRP-II-A, where some variables and parameters are removed, and constraints are tightened. Besides, they introduce a new upper bound, and a preprocessing step fixing some variables. The preprocessing step removed 65 to 89 % of the variables, so they could solve instances with more than 25 items. Voß and Schwarze [110] give an in-depth analysis of different objectives for BRP-II-A, including the number of relocations and the crane working time.

Galle et al. [40] enhance the binary encoding introduced in [18] and derive a new IP model (CRP-I) from it. Previous models for the restricted BRP described layouts with variables describing item-stack assignments. In contrast, CRP-I uses variables to determine whether an item is below another item. This leads to much fewer variables. Within a time limit, CRP-I could solve significantly more instances compared to BRP-II-A. In further experiments, CRP-I outperformed BRP-II* and BRP2ci in terms of computational time. CRP-I solved with the commercial solver Gurobi has also shown good performance compared to B&B methods from [32, 93] and the method from [64].

Zehendner and Feillet [119] present the first *Column Generation* approach for the restricted BRP. They decompose BRP-II into a master problem and a pricing subproblem, both formulated as IP models. In the master problem, variables represent sequences of moves, each for retrieving a single item. They present three ways to solve the pricing subproblem: one binary IP model and two enumeration-based approaches.

Bacci et al. [5] introduce a formulation for the restricted BRP in which the number of time periods is equal to the number of items to be retrieved. To cope with the exponential number of constraints, the model is solved by a Branch & Cut (B&C) algorithm with a custom cutting strategy. In their results, B&C was found more efficient than CRP-I, and even the best-so-far B&B method from [92].

For solving the unrestricted BRP, Petering and Hussein [78] improve the BRP-I model by removing unnecessary variables. Results show that with the new model called BRP-III, CPLEX achieves significantly shorter computational times, by a factor of 100 on some instances. The LP relaxation also offers a tighter upper bound compared to BRP-I.

de Melo da Silva et al. [84] present two models, BRP-m1 and BRP-m2 for the unrestricted BRP with duplicate priorities. The main difference between the two is that BRP-m1 allows only one move per time step, while BRP-m2 allows one retrieval and one relocation at the same time step. Thus, the size of BRP-m2 can be reduced compared to BRP-m1. Whereas BRP-m1 obtained better linear relaxations, BRP-m2 was on average faster than BRP-III and BRP-m1. Note that BRP-m1 and BRP-m2 can also solve the restricted BRP in their respective variants, R-BRP-m1 and R-BRP-m2.

Whereas most of the BRP formulations define binary variables to determine whether an item is located at a given slot at a given time, Lu et al. [72] propose a strong formulation (BRP-m3) with a different approach. They instead use variables that describe the adjacency relationship between pairs of items. Their formulation can be easily modified to solve eight variants of the BRP, including the restricted/unrestricted BRP, with/without duplicate priorities, and with/without complete retrieval. Besides, the authors introduce a MIP relaxation-based iterative procedure that solves a relaxed version of BRP-m3 and strengthens the latter until the optimality criterion regarding BRP-m3 is met. BRP-m3 achieved shorter computational times than BRP-m2.

Tanaka and Voß [97] reformulate the restricted BRP as the problem of finding an optimal combination of relocation sequences. Thus, the authors introduce an IP model in which each variable encodes a relocation sequence of a single item, and constraints avoid selecting conflicting sequences. Since the number of variables and constraints is large in practice, they use an iterative approach. The algorithm starts with a limited number of truncated relocation sequences. Then the algorithm repeatedly solves the model with the sequences being extended and coupling constraints being added on-the-fly until the optimality gap becomes zero. This approach obtained a better performance than an enhanced version of the *Branch & Bound* from [93] and the *Branch & Cut* from [5]. Moreover, this IP-based approach could solve all the instances with up to 100 items to optimality within one hour.

Models for the pure BRP are summarized in Table 4. The following works tackle variants of the BRP.

Galle et al. [40] show that CRP-I can be easily extended to solve the three following BRP variants: (1) with non-uniform relocation costs, (2) minimizing the crane travel distance, and (3) with one voluntary move allowed per retrieval. Lu et al. [72] modify BRP-m3 to tackle the following variants: (1) BRP with penalty coefficients, (2) BRP considering energy consumptions, (3) BRP subject to stacking restrictions, (4) BRP considering retrieval pace.

Table 4: Mathematical formulations for the pure BRP

Model	Restricted	Unrestricted	Duplicate priorities	References
BRP-II	✓			Caserta et al. [19]
BRP-II*	✓			Expósito-Izquierdo et al. [32]
BRP2ci	✓			Eskandari and Azari [30]
BRP-II-A	✓			Zehendner et al. [117]
CRP-I	✓			Galle et al. [40]
BC-RBRP	✓			Bacci et al. [5]
Relocation sequences	✓			Tanaka and Voß [97]
R-BRP-m1/2	✓		✓	de Melo da Silva et al. [84]
BRP-I		✓		Caserta et al. [19]
BRP-III		✓		Petering and Hussein [78]
BRP-m1/2		✓	✓	de Melo da Silva et al. [84]
BRP-m3	✓	✓	✓	Lu et al. [72]

Ji et al. [51] propose an IP model for a variant of the *BRP with Stowage Plan*, where items of a source storage area have to be retrieved to fill multiple destination storage areas with designated slots. The goal is to determine a loading sequence minimizing the number of relocations. da Silva Firmino et al. [86] use the BRP-II* and BRP-II-A as a base of a new IP model (we call it BRP-F) optimizing the crane working time. López-Plata et al. [71] aim at solving the BRP with waiting times. Their IP model minimizes the differences between the actual and the expected retrieval times. Tang et al. [99], Tang and Ren [100] tackle the *Slab Stack Shuffling* problem. The main difference with the pure BRP is that items belong to families. Families are given as a sequence and one item per family must be chosen for retrieval. Tang et al. [99] model the SSS minimizing the number of relocations where relocated items are pushed back. Tang and Ren [100] formulate an IP model for minimizing the total crane workload. Zeng et al. [121] formulate an IP model based on MRIP and MRIP2, to solve a restricted BRP in which items are split into groups to be picked up in a certain order. However, the pickup order of items within a same group is unknown. The IP model integrates additional variables to decide the pickup order within groups, to minimize the number of relocations. Feng et al. [35] propose a stochastic *Dynamic Programming* model for the stochastic BRP with flexible service policies to minimize the expected number of relocations. Each item is associated with a time window, during which a truck arrives for pickup. An optional second objective aims at minimizing the truck waiting times. Inspired by BRP-I, Azab and Morita [2] propose two IP models that allow rescheduling the retrieval times of items within time windows.

3.2. Formulations for the PreMarshalling Problem

Lee and Hsu [67] introduce the first IP model for the PMP, composed of a multi-commodity flow problem and a set of side constraints. Three extensions are proposed: (1) one can impose a target final layout, (2) one imposes that each stack contains only one item type, and (3) one allows items to leave the storage area during the premarshalling process. de Melo da Silva et al. [84] present a model (PMP-m1) that decreased significantly the computational times compared to [67]. Note that their model can solve the PMP with a target final layout. Parreño-Torres et al. [76] designed four models named IP3 to IP6. All the models use two groups of variables, x representing the layout at a given time, and y representing the moves. The main difference between the IP3 to IP6 models is that a set y is indexed by 3 to 6 values, respectively. Also, the authors provide alternative formulations IPS3 to IPS6 that split the set y into two sets of variables, one indicating the origin and one indicating the destination of the move. Experiments showed that splitting the variables led to shorter computational times, IPS6 being the fastest. Compared to [67] and PMP-m1, IPS6 was also significantly faster. The authors have extended the model for the following goals: (1) limiting the height difference between adjacent stacks in the final layout, (2) avoiding empty or full stacks, (3) imposing mimimum and maximum stack heights, (4) favoring that same-priority items share the same stack. Parreño-Torres et al. [77] use IPS6 as the base for proposing the IPCT model, to solve the PMP minimizing the crane working time. van Brink and van der Zwaan [15] decompose the PMP as a master problem and a pricing subproblem, suitable for *Column Generation* algorithms. They formulate the master problem as an IP model, in which each variable corresponds to a stack and a sequence of moves. The pricing subproblem is similar to finding a maximum weight independent set in a circle graph. As an alternative to IP models, Rendl and Prandtstetter [82] formulate the PMP using *Constraint Programming*. Besides the classical PMP, they propose a model for the robust PMP where retrieval times are uncertain.

Boge et al. [6] tackle a PMP where the retrieval order of items is uncertain. They decompose the main problem into a master problem and an adversary subproblem. The master problem, formulated as two IP models, maximizes the maximum number of misplaced items over all scenarios. The adversary subproblem, also formulated as an IP model, aims at finding worst-case scenarios. An iterative method starts with an arbitrary scenario. Solving the master problem gives a candidate solution that is then evaluated by solving the adversary subproblem. If both objectives are equal, the solution is optimal. Otherwise, the scenario generated by the adversary subproblem is added to the master problem, and the process is repeated. Dayama et al. [27] formulate several IP models for the *Container Stack Rearrangement Problem* (CSRP), where each item has to be moved to a specified destination stack, but no vertical order of items is required. During the process, blocking items are placed in a temporary staging area and later pushed back to their original stack.

3.3. Future directions

IP models have been dramatically improved during the last years. For the BRP, the formulation from [97] and BRP-m3 [72] achieve the best computational times. The former can solve the restricted BRP instances with 10 stacks and 6 items per stack within seconds, that are typical layouts in container terminals. BRP-m3 achieves also competitive results, instances of the unrestricted BRP with 5 stacks and 5 items per stack being solved within one hour. For the PMP, IPS6 [76] may be the actual best model, solving instances having 6 stacks and 4 tiers within one hour. Nevertheless, the promising decomposition from [15] has not been compared yet. To the best of our knowledge, the IP-based iterative method from [97] and *Branch & Bound* methods [92, 95] are the current fastest exact methods for the BRP and the PMP. Note that IP models have the advantage of following the performance improvements of IP solvers. Therefore, their computational times should reduce over time with no change required.

In the meantime, further research might consider the following aspects. The vast majority of the models maintain variables that assign items to slots. Using variables determining whether items are stacked on top of other items as suggested in [72] can significantly reduce the number of variables. To the best of our knowledge, such an approach has not been proposed for the PMP. Besides, tighter upper bounds may be found to discretize the time horizon in fewer time periods. Besides, BRP-m3 has not been compared with some recent IP models for the BRP such as BC-RBRP. de Melo da Silva et al. [84] note that their model R-BRP-m1 gave the most promising results for solving the restricted BRP, but did not outperform CRP-I. Including a preprocessing step as in [40] to remove variables may reduce the computational times of R-BRP-m1. Galle et al. [40] suggest a way to increase the efficiency of CRP-I by including the combinatorial lower bounds from [127, 93].

Due to common characteristics of stacking problems, ideas that work for the BRP could work with the PMP. A trending topic is the incorporation of uncertainty, leading to robust models. A direction to investigate may be to extend existing formulations for different types of uncertainty. In this survey, we listed one *Constraint Programming* (CP) approach and a few *Column Generation*-based (CG) formulations. Further investigation is required to check whether CP is an efficient approach. Since they require only a small subset of variables, CG algorithms have been used in other domains and have been successful for solving real-world instances. Moreover, to the best of our knowledge, the approaches from [15] and [119] have not been compared with the recent IP models. Another direction is to investigate whether the successful iterative approach from [97] can be adapted to the unrestricted BRP or the PMP. Finally, researchers may consider formulating more time-based stacking problems such as [9] to fit more realistic constraints such as service times.

4. Heuristics

Due to their similar structure, the BRP and the PMP often share common types of decisions. We distinguish two types of decisions that characterize most stacking problems.

- **Item selection**. Choosing the next item to operate occurs in particular when several items are simultaneously available for placement or relocation. This is a decision that permanently occurs in the PMP. In the unrestricted BRP, one may have to decide whether to move a blocking item or perform a voluntary move. The restricted BRP may encounter this type of decision when multiple items have the same departure time (duplicate priorities).

- **Stack selection**. When an item of interest has been chosen, one has to determine its destination stack. This type of decision is considered in both the PMP and the BRP, at each relocation.

We classify heuristics in three categories: *item selection heuristics*, *stack selection heuristics*, and *complex heuristics*. The two former categories cover heuristics that focus on the previous decision types. Heuristics that do not fall into the two categories, use advanced techniques, or take simultaneous decisions, are considered complex heuristics. A list of references and heuristics can be found in Table 5.

Table 5: Summary of heuristic methods

Reference	Methods	Problems: rBRP	uBRP	PMP	Notes
Caserta et al. [18]	Matrix	✓			
Caserta et al. [19]	MinMax	✓	✓		
Expósito-Izquierdo et al. [33]	LPFH			✓	
Expósito-Izquierdo et al. [31]	DSKB	✓	✓		
Feillet et al. [34]	Local search		✓		
Feng et al. [35]	Expected MinMax	✓			stochastic BRP with service times
Forster and Bortfeldt [38]	SGreedy		✓		
Galle et al. [41]	MinMax	✓	✓		
Galle et al. [42]	MinMax, EGA	✓			
Huang and Lin [49]	Labeling			✓	
Ji et al. [51]	Rule-based	✓			multi-crane with stowage plan
Jin et al. [53]	GLAH	✓			
Jovanovic and Voß [57]	MinMax, Chain	✓			
Jovanovic et al. [55]	LPI, Multi			✓	
Jovanovic et al. [54]	MinMax, MinB	✓			with stowage plan
Kim and Hong [59]	ENAR	✓			
Kim et al. [60]	Rule-based		✓		
Ku and Arthanari [63]	ERI	✓			with time windows
Lee and Hsu [67]	IP-based			✓	
Lee and Chao [66]	IP-based			✓	
Lee and Lee [68]	LL	✓			minimize crane time
Lin et al. [70]	Rule-based		✓		minimize crane time
López-Plata et al. [71]	Look-ahead		✓		with waiting times
Olsen and Gross [75]	Extended MinMax	✓			
Petering and Hussein [78]	LA-N		✓		

Continued on next page

Table 5 (continued)

		Problems			
Reference	Methods	rBRP	uBRP	PMP	Notes
Prandtstetter [79]	DP-based			✓	
Rendl and Prandtstetter [82]	Specialized search			✓	
Tang and Ren [100]		✓			slab stack shuffling
Tang et al. [101]	Rule-based	✓			
Tang et al. [98]	H1-H5, IP-based	✓			
Ting and Wu [104]	VRH	✓			
Tricoire et al. [105]	SM-2, SmSEQ-2		✓		
Ünlüyurt and Aydın [108]	EAR, Difference	✓			
Wan et al. [111]	MRIP	✓			
Wang et al. [112]	Target-guided			✓	
Wang et al. [113]	Feasibility-based			✓	
Wu and Ting [116]	RIL	✓			
Zehendner et al. [120]	Leveling	✓			
Zeng et al. [121]	H1-H6, IP-based	✓			
Zhang et al. [124]	α/β, E/H			✓	
Zhang et al. [125]	Greedy		✓		with batch moves
Zhu et al. [127]	PR1-4, PU1-4	✓	✓		
Zweers et al. [128]	Local-search	✓			stochastic BRP with time windows

4.1. Item selection heuristics

The main challenge of the PMP is to find a finite sequence of moves that leads to a layout without misplaced items. Without a temporary storage area, the order in which a heuristic rearranges items may impact whether it finds a feasible solution or not. Methods proposed such as the *Corridor Method* (described in Section 5) from [21] or the *Tree Search Procedure* from [10] do not guarantee that a final solution is found. To deal with this issue, *LPFH* [33] handles misplaced items from the latest to earliest departure time. The idea is that once an item with the latest departure time is well-placed, it will no longer interfere with the rearrangement of the rest of the items. To choose among items having the same departure time, *LPFH* calculates the number of required moves for each of them, and randomly selects one from a restricted candidate list. *Target-guided* algorithms fix items at appropriately chosen locations and avoid further moves afterward, although Wang et al. [112] still treat misplaced items by decreasing departure times. In contrast, the *Feasibility-based* heuristic [113], which is also *Target-guided*, does not impose a predetermined order of item rearrangement. Instead, the heuristic detects and prunes decisions that lead to undesirable states before their application.

Item selection decisions also occur in the unrestricted BRP (since the latter allows voluntary moves) and the BRP with duplicate priorities. In the BRP with duplicate priorities, one has to choose between items of the same priority. Kim and Hong [59] apply the *ENAR* heuristic for each candidate target item and select the action obtaining the minimum expected number of additional relocations (plus realized relocations). The **PU1** heuristic [127] considers voluntary moves only when a relocated item is about to remain misplaced. If a topmost item is the earliest of its stack, and a voluntary move does not make it misplaced, then *PU1* considers it as a better candidate. Forster and Bortfeldt [38] evaluate forced relocations as well as voluntary moves with a generalization of *MinMax* from [19] (**S_{GREEDY}**) to determine the next action. The former approach reduced the average number of moves by 8.7 % with shorter computational times compared to LL, an IP-based heuristic described in Section 4.3.2 [37]. The *LA-N* heuristic [78] allows voluntary moves for items that belong to a stack containing one of the N next items to retrieve. To choose among target items of the same priority, Jin et al. [53], Lin et al. [70] first retrieve items having fewer items above. For the unrestricted BRP, Jovanovic et al. [56] define **Gre-C** and **Gre-N**, heuristics based on *MinMax*. *Gre-C* considers relocating misplaced items, including items not above the target. *Gre-N* considers relocating any item, including well-placed ones. Well-placed items are a candidate for relocation only in certain conditions. Tanaka and Voß [96] improve the greedy heuristic used in [54] for the BRP with storage plan allowing voluntary moves.

4.2. Stack selection heuristics

Numerous heuristics from the literature are based on *decision indexes* for choosing destination stacks for incoming or relocated items [86]. The basic idea is to compute a desirability score for every possible choice and to select a stack having the best score or randomly among a restricted candidate list of elite stacks. The **Leveling** heuristic (also called *Lowest Slot*) selects a stack containing the minimum number of items. For breaking ties, one may choose a stack randomly [111], or select the first stack of the list [7, 120]. For the online BRP, *Leveling* guarantees a competitiveness ratio of $2\lceil \frac{n}{m} \rceil - 1$, for a storage area with n items and m stacks [120]. A drawback of *Leveling* is that it does not take advantage of information such as the due times of items. To overcome this issue, the **Reshuffle Index (RI)** heuristic [74] assigns the current item to a stack containing the minimum number of items departing earlier. An extension of *RI* called **Reshuffle Index with Look-ahead (RIL)** [116] breaks ties by choosing a stack in which the earliest departure time is the latest. *RI* has also been adapted for the BRP with time windows [63] and named as **Expected Reshuffle Index (ERI)**. The **Expected Number of Additional Relocations (ENAR)** [59] estimates the number of relocations to be added if items from other stacks are relocated to the candidate stack. To do so, *ENAR* recursively computes probabilities and derive an expectancy. Wan et al. [111] improve *ENAR*, *Leveling*, and *RI*, by computing decision indices based on the resulting layout after move instead of the current layout. Ünlüyurt and Aydın [108] adapt *ENAR* to take into account the crane's horizontal travel. The **Average Time Index-Based (ATIB)** heuristic [1] selects a stack with the latest average departure time.

When relocations cannot be avoided, a good strategy is to postpone them as much as possible. To do so, Ünlüyurt and Aydın [108] propose the **Difference** heuristic for the BRP. *Difference* first considers stacks in which all items have later departure times and selects one containing the earliest time. If no such a stack exists, among stacks having a topmost item departing later, one having the earliest topmost item is chosen. Otherwise, *Difference* selects a stack in which the topmost item has the latest departure time.

A similar and very popular greedy heuristic for the BRP is **MinMax** from [19]. Like *Difference*, *MinMax* first considers stacks in which all items have later departure times and selects one containing the earliest one. If all stacks contain at least one item departing earlier, *MinMax* selects a stack having the latest departure time. Zhu et al. [127] design a variant of *MinMax* called **PR4**, that introduces an additional rule when every choice leads to additional relocations. Jovanovic and Voß [57] improve *MinMax* by avoiding the creation of new deadlocks when a stack is going to reach the maximum height. Jovanovic et al. [54] and Tanaka and Voß [96] extend *MinMax* for the restricted and unrestricted BRP with stowage plan, respectively. The **Expected MinMax** [42] adapts *MinMax* for the Stochastic BRP, in which the retrieval order is not fully known in advance. The latter heuristic is extended in [35] for the Stochastic BRP with flexible service policies, in which items are associated with time windows during which a truck arrives for pickup.

A way for improvement is to look ahead within the heuristic search. Caserta et al. [18] introduce the **Matrix** heuristic for the BRP, a look-ahead algorithm that encodes layouts as binary matrices. The **LA-N** heuristic [78] extends *MinMax* by considering the N next retrievals to determine eligible relocations. In the **Chain** heuristic, Jovanovic and Voß [57] redefine *MinMax* to take into account the next item to relocate. The **Virtual Relocation Heuristic (VRH)** [104] for the restricted BRP, inspired by *Chain*, determines simultaneously destination stacks for all items blocking the current target. The **SmSEQ-2** heuristic [105] includes a rule detecting decreasing sequences of departure times of consecutive items. The idea is that if the topmost item can be relocated without causing future relocations, the whole sequence can be relocated in the same destination stack. Otherwise, the heuristic falls back to a simpler one, **SM-2**, that attempts to avoid conflicts by performing *safe relocations*.

The **Lowest Priority First Heuristic (LPFH)** [33] for the PMP gives a score to each candidate stack, and randomly selects one among a restricted list of stacks having the best scores. LPFH chooses a destination stack for the target item, then for the items above it. For the target item, LPFH favors stacks having the minimum number of items to be removed to make it well-placed. For items above it, LPFH prefers stacks having no misplaced items or stacks in which misplaced items have the latest departure times. The latter rule is called **Lowest Priority Index (LPI)** [55].

We remind that the PMP does not always admit a feasible solution. To our knowledge, no heuristic explicitly allows exceeding the maximum stack height when no feasible solution can be constructed. When the maximum stack height cannot be increased, a workaround is to allow using a dummy stack, as in [112].

4.3. Complex heuristics

4.3.1. Rule-based heuristics

Rule-based algorithms may have a complex structure, although easy to implement. The *Blocking Index* (BI) of a stack is the number of items blocking the earliest item after putting the current item. Based on the RI and the BI, Tang et al. [98] describe five rule-based heuristics, **H1-H5**, for the restricted BRP. These heuristics were also applied to the DCRP, in which incoming items arrive during the unloading process. Lin et al. [70] describe a rule-based algorithm for the restricted BRP that takes into account the crane movement time and features specific to container terminals, such as inter-bay relocations. The latter algorithm is challenged by another rule-based algorithm [60] that applies for the unrestricted BRP.

For the PMP, the strategy in [49] is to label stacks as clean (i.e. without misplaced items) or dirty (i.e. containing misplaced items), and to fill up clean stacks to make some stacks empty. Afterward, items from dirty stacks are moved to empty stacks while avoiding unordered stacks. For the PMP where items have target cells, the authors propose to label each stack with the number of items that need to be removed and the number of other items. The algorithm finds moving paths while scanning stacks one after another until the number of items that need to be removed is reduced to zero. Zhang et al. [124] propose one heuristic (α/β) that also distinguishes clean and dirty stacks, and another (**E/H**) that distinguishes "easy" and "hard" stacks. Hard stacks are dirty stacks in which items are in the reverse order of priority. α/β rearranges dirty stacks in an arbitrary order. *E/H* first rearranges easy stacks, then hard stacks. Jovanovic et al. [55] extend *LPFH* for the PMP into a **Multi heuristic** framework to incorporate arbitrary rules. *Multi* requires four heuristic components: (1) to select the target item, (2) to select a destination stack for the target item, (3) to move items above it, (4) to fill the destination stack with well-placed items. In particular, the authors test *Leveling*, *LPI*, and *MinMax* as components for (3).

The **Domain-Specific Knowledge-Based** (DSKB) heuristic [31] for the restricted BRP constructs new solutions until no improvement is observed for a certain number of iterations. At each step of the construction of a single solution, the heuristic attempts to reach a state in which the current item is retrieved and the number of future relocations is minimized. For the BRP, Jin et al. [53] propose a **Greedy Look-Ahead Heuristic (GLAH)** split into three levels. The top level is a greedy mechanism that executes one relocation at each stage. The middle level runs a tree search of limited depth (3 or 4) to guide the greedy mechanism at the first level. The bottom level applies a set of heuristic rules extended from [127] to evaluate leaf nodes explored in the second level. Also, the authors apply a solution condensation approach, improved in [105]. It transforms two relocations into one when some conditions are satisfied, without sacrificing feasibility. *GLAH* has been adapted for a *Steel Stacking Problem* [81].

4.3.2. IP-based heuristics

Some works exploit the idea of embedding exact methods such as IP solvers in heuristic methods. Wan et al. [111] introduce an IP-based look-ahead heuristic, **MRIP**, for the BRP. At each step, the algorithm determines the destination stack of blocking items by solving an IP model (MRIP_K) that considers the retrieval of the next K items. *MRIP* is further improved and extended in [98, 121]. A similar idea has been adapted for the DCRP [1]. The idea of using IP models in heuristic methods is further developed in [68]. **LL** is a three-phase method for the restricted BRP. The initial phase builds a complete solution in a greedy fashion by relocating blocking items in the nearest available stack. A movement reduction phase and a time reduction phase attempt to reduce, respectively, the number of moves and the crane working time of the initial solution. During these two phases, the algorithm finds alternate paths for items and assembles them into a super-sequence of moves. After identifying all possible conflicts of the super-sequence, an IP determines a best feasible combination of alternate paths.

For the PMP, Lee and Hsu [67] solve iteratively a relaxed IP model. At each iteration, constraint violations are detected, and new constraints are added to the model until a set of movements satisfying conditions is obtained. However, the obtained sequence of movements may contain cycles. In the second phase, the algorithm attempts to break these cycles in the movements by introducing new movements. Lee and Chao [66] describe an algorithm for the PMP, starting from an initial sequence of movements, and iteratively running two major subroutines. First, a neighborhood search builds a new feasible sequence of moves by randomly modifying the current one. Second, a binary IP is solved to shorten the sequence of moves while keeping the final layout. Besides, three minor subroutines further improve the solution.

4.3.3. Post-processing heuristics

Existing heuristics can be improved by post-processing techniques. Previously discussed IP-based heuristics [67, 66] commonly use post-procesing techniques. For the unrestricted BRP, Feillet et al. [34] develop a local search heuristic based on dynamic programming. Their method reached improvements of up to 50% on solutions built by *LA-N*, *SM-2*, *SmSEQ-2* and *GLAH*. Another local search heuristic is described in [128] for the stochastic BRP. The heuristic attempts to convert moves that keep items misplaced to moves that make them well-placed, therefore decreasing the number of forced moves. Zweers et al. [128] develop a local search heuristic for the stochastic BRP inspired by the *Expected MinMax* [42] and *LPFH* [33].

4.4. Future directions

For the BRP, the most competitive heuristics are *GLAH* [53], *SM-2*, and *SmSEQ-2* [105]. Whereas *SmSEQ-2* obtains the fewest relocations, *GLAH* and *SM-2* achieve shorter computational times. Experiments from [105] suggest that as instance size grows, state-of-the-art heuristics are still far from optimal solutions. Indeed, a local search heuristic can significantly improve the solutions obtained by these heuristics [34]. Future research may consider the design of more efficient improvement methods.

Due to the larger solution space, the unrestricted BRP yields more opportunities for improvement than the restricted BRP. Existing heuristics for the unrestricted BRP could be accelerated by intelligently limiting the search space explored [34]. Better solution quality could be obtained by using look-ahead mechanisms, especially in local search methods.

Some heuristics designed for the BRP have been successfully adapted to variants of the BRP, such as the BRP with stowage plan. A research direction can be to extend existing methods to cover a wider range of stacking problems, and to tackle more realistic constraints.

Fewer heuristics have been developed for the PMP, suggesting research opportunities. The *Multi* heuristic [55] and the *Feasibility-based* heuristic [113] show the best results for the PMP.

Finally, a few works have analyzed heuristics theoretically. For the BRP, Olsen and Gross [75] perform a probabilistic analysis of a heuristic that is closely related to *MinMax*. *MinMax* is on average at most 1.25 away from the optimal solution [41]. More theoretical analysis could be made for various heuristics. Moreover, the study of the online BRP [120] can be extended to the assumption where the retrieval order is defined for groups of items rather than single items.

5. Metaheuristics

Due to their flexibility, metaheuristics are often used for solving problems involving realistic constraints. Metaheuristics include Ant Colony Optimization (ACO), Corridor Method (CM), Genetic Algorithms (GA), Greedy Randomized Adaptive Search Procedure (GRASP), Pilot Method (PM), Simulated Annealing (SA) algorithms. This section describes these metaheuristics along with their characteristics. Table 6 summarizes references using metaheuristics and which problems they tackle.

Table 6: Summary of metaheuristics

		Problems			
Reference	Methods	rBRP	uBRP	PMP	Notes
Jovanovic et al. [56]	ACO	✓	✓		minimize crane time with stowage plan
Tus et al. [106]	ACO PM			✓	2D-PMP
Caserta and Voß [22]	CM	✓			
Caserta and Voß [21]	CM			✓	
Caserta et al. [23]	CM	✓			
Gheith et al. [44]	GA			✓	variable-length GA
Hottung and Tierney [48]	GA			✓	
Ji et al. [51]	GA	✓			multi-crane with stowage plan
Tang et al. [99]	GA	✓			slab stack shuffling
Jovanovic et al. [54]	GRASP	✓			with stowage plan
da Silva Firmino et al. [86]	GRASP	✓			minimize crane time , reactive GRASP
Tricoire et al. [105]	PM		✓		
ElWakil et al. [29]	SA	✓			

Continued on next page

Table 6 (continued)

Reference	Methods	Problems rBRP	uBRP	PMP	Notes
Tang et al. [101]	Tabu	✓			

5.1. Ant Colony Optimization

Tus et al. [106] introduce an *Ant Colony Optimization* (ACO) algorithm for the Two-Dimensional PMP (2D-PMP), where reach stackers can access items from the left and the right of the storage area. The idea of ACO is to store a pheromone matrix that remembers experience gained by previously produced solutions. At each iteration, a colony of artificial *ants* generates solutions guided by the pheromone trails, the latter being updated according to the generated solutions. The authors adopted the *Max-Min Ant System* (MMAS) approach, where only the best ant updates the pheromone trails, and pheromone values are bounded. From initial experiments, they found that a state-based pheromone matrix (in which each value represents a layout) does not perform well. Instead, they use a move-based pheromone matrix, that associates its values to state-move pairs. Since the number of possible states is extremely large, pheromone values are created on the fly in a hash table. After finding no improvement for a certain number of iterations, the algorithm restarts to avoid stagnations. In terms of the number of relocations, MMAS outperformed LPFH and the Pilot Method.

Jovanovic et al. [56] present an ACO algorithm for both the restricted and unrestricted BRP, as well as the objective of reducing the crane working time. In their approach, they associate the pheromone values to (i, d, p, t) tuples, where i is the item to relocate, d is the minimal retrieval time of items in the destination stack (which is set to a large value if the stack is empty), p is the number of times item i was moved, and t is the target item. The construction algorithms, *Gre-C* and *Gre-N* are discussed in Section 4. ACO outperformed CM, LA-N, the Domain-Specific Knowledge-Based Heuristic from [31] and the Heuristic Tree Search Procedure from [38].

5.2. Corridor Method

The Corridor Method (CM) is a method-based iterated local search inspired by dynamic programming [87]. Since complex methods can solve efficiently small instances, the idea is to use the same methods on a restricted portion of the solution space for large instances. The restricted solution space is called a *corridor*. All the CM variants presented below restrict the number of candidate stacks for relocation with a user-defined parameter. For the BRP, Caserta and Voß [22] use an algorithm inspired by GRASP to build a pool of elite solutions. Then a roulette-type scheme randomly selects a solution from the elite set. Caserta et al. [23] also define a vertical corridor, that is a maximum height for stacks. For the PMP, [21] select the next target item with a roulette-wheel mechanism that favors items located in stacks involving less forced relocations. Then the choice of destination stacks for the items blocking the target is restricted in a similar way as with the BRP. Besides, the authors apply a move-based local search to improve the current layout.

5.3. Genetic Algorithms

Genetic algorithms (GA) are a class of metaheuristics inspired by the evolution theory [28].

Tang et al. [99] tackle the *Slab Stack Shuffling* problem, where items belong to families. Families are given as a sequence and one item per family must be chosen for retrieval. A chromosome is encoded as a sequence of integers, where each value represents an item selected for a given family. The authors apply three crossover operators: one-point, two-point, and one that exchanges genes one by one with a uniform probability. In addition, they test two mutation operators exchanging genes while ensuring feasibility.

Ji et al. [51] design a GA for a variant of the *BRP with Stowage Plan*. In this problem, items of a storage area have to be retrieved to fill multiple destination storage areas with designated slots. The retrieval order is determined by the genetic algorithm, GA-ILSRS. Thus, GA-ILSRS encodes solutions as a vector representing a loading sequence. GA-ILSRS uses a two cross-bit method for both the crossover and the mutation operators. Finally, three heuristics determine the destination stacks of relocated items: nearest stack strategy, lowest stack strategy, and a strategy that avoids putting relocated items above the next target item (called optimization strategy). The optimization strategy was found to be the best, followed by the lowest stack strategy.

Gheith et al. [44] apply the idea of variable chromosome lengths for the PMP. Chromosomes encode solutions as a sequence of moves, where each gene is a pair origin-destination. In addition of a single-point crossover, they apply four mutation operators. A growth and a shrink mutations modify the length of the chromosome, whereas a swap and a replace mutations improve the solutions while keeping their original length. The fitness function is the number of blocking items.

Hottung and Tierney [48] introduce a *Biased Random-Key Genetic Algorithm* (BRKGA) for the PMP. The *random-key* GA is a variant of GA where genes consist on a sequence of floating point numbers between 0 and 1. The solutions are then produced from genes by interpreting them using a non-deterministic decoder. BRKGA is *biased* since it applies each crossover on one random elite and one random non-elite solution. Their original idea is to incorporate an online learning mechanism to the construction method. They define a class of moves named as *excellent moves*, that are nearly always present in optimal solutions. When these moves are available, they are automatically applied, otherwise a move is selected in a heuristic manner. BRKGA outperformed the Heuristic Tree Search Procedure [38], whereas it obtained contrasted results compared to the Target-Guided algorithm [112]. BRKGA reduced the number of relocations compared to the *Corridor Method* [21] and LPFH [33] (described in Section 4), however, took longer computational times.

5.4. GRASP

The *Greedy Randomized Adaptive Search Procedure* [36] is a parameterizable algorithm combining a constructive phase and an improvement phase. The construction phase incrementally builds a solution. At each step, candidate decisions are evaluated by a *utility function*, and added to a *Restricted Candidate List* (RCL), if their utility is greater than a threshold. The threshold is computed according to a user-defined parameter α to make the algorithm more deterministic or more randomized. One decision is then chosen randomly from the RCL and added to the partial solution. The improvement phase, typically a local search, attempts to find better solutions by applying simple modifications. In stacking problems, decisions are typically choosing a destination stack for an item, or choosing the next item to move.

For reducing the crane working time in the BRP, da Silva Firmino et al. [86] propose a *Reactive GRASP* approach. Reactive GRASP is an extension of GRASP that self-adjusts the α parameter during the execution. The authors tested six different utility functions found in the literature: LS, RI, RIL, ENAR, LADI [115] and MNI [57]. During the improvement phase, the algorithm attempts to find better solutions by replacing moves in the current one, until reaching a local optimum. From the experiments, the authors found that the best utility function was MNI and adopted it as the scoring function of the Reactive GRASP. The average percentage optimality gap of Reactive GRASP with MNI was 1.15%.

Another GRASP approach is proposed in [54] for solving a restricted version of the *BRP with Stowage Plan* (BRP-SP). The utility function is based on a modified MinMax function. Also, their algorithm maintains a precedence graph exploiting the structure of the problem. For the improvement phase, they apply a correction procedure to delete moves having undesirable properties. GRASP significantly outperformed the GA from [51] by reducing the number of relocations by approximatively 30%.

5.5. Pilot Method

The *Pilot Method* (PM) is a look-ahead metaheuristic that takes a construction algorithm as an input. At each iteration, every possible decision (e.g. relocation) is evaluated by the construction algorithm. The best candidate is selected as the new current partial solution, and the process is repeated until the current solution is complete.

For solving the unrestricted BRP, Tricoire et al. [105] use a *Rake Search* as the construction algorithm. Rake Search consists of a breadth-first tree search where the tree is generated level by level until the number of nodes reaches a user-defined limit. Afterward, the tree search is stopped, and each partial solution is used as a starting point of four given construction heuristics. In their experiments, Rake Search was the fastest. Whereas PM was significantly slower, results show that it seems more scalable than GLAH from [53]. A variant of the Pilot Method has been applied for a stacking problem for the steel industry, involving stacking and time constraints, as well as buffer stacks [81]. The algorithm incorporates a Rake Search as in [105], and a greedy look-ahead heuristic.

Tus et al. [106] tackle the 2D-PMP, a variant of the PMP where items are relocated by a crane but will be retrieved by reach stackers. As a construction algorithm, the authors adapt

LPFH from [33] for the 2D-PMP. The latter heuristic is described in Section 4. Results show that PM is significantly faster than ACO algorithms at a cost of solution quality.

5.6. Simulated Annealing

The *Simulated Annealing* (SA) is a stochastic method that mimics the cooling of metals. SA starts from an arbitrary solution, and generates a new potential solution by altering the current one. If the new solution is better than the current one, it is accepted. Otherwise, SA accepts the new solution with a probability that varies according to a decreasing temperature T.

The SA in [29] for the restricted BRP constructs iteratively a sequence of moves. The cost function of a partial solution sums up the number of realized moves and the number of items blocking the target. When a partial solution is accepted, the algorithm determines randomly the destination stacks of the blocking items. SA showed better results than *Tabu search* [116] and the *Corridor Method* [23] on small instances, but not on large ones.

5.7. Future directions

Metaheuristics are one of the most promising approaches for solving realistic and complex problems, due to their flexibility and their short computational times. For the BRP, Rake Search, Pilot Method [105], Reactive GRASP [86], and ACO [56] have shown very competitive results. To the best of our knowledge, no full comparison has been made between these methods. Different metaheuristics may be investigated for solving problems involving side constraints such as the 2D-PMP [106].

Nevertheless, existing methods for classic problems may still be improved for solving larger instances. For the BRP, Tricoire et al. [105] observe that metaheuristics may still be far from optimal solutions. The settings of the SA from [29] may be tuned to lead to better solutions, e.g. by changing the cooling behavior. Besides, metaheuristics could be combined with the efficient local search procedure from [34]. For the PMP, Hottung and Tierney [48] suggest developing a stochastic version of their GA to obtain more robustness than the deterministic version.

6. Tree search-based methods

Tree search-based (abbreviated as TS-based) methods include exact methods such as Branch & Bound (B&B) as well as approximate methods such as Beam Search (BS). For a review of Integer Programming formulations, we refer the reader to Section 3.

TS-based methods attempt to find a solution by traversing a tree structure. In every method presented in this section, a node represents a layout, the root node being the initial layout. Child nodes represent layouts in which moves have been performed. The process of exploring child nodes is called *branching*. *Bounding* refers to the process of eliminating nodes that are guaranteed to not contain any better solution than the current best solution. To this end, TS-based methods may maintain the global lower and upper bounds of the objective function. Each node can be evaluated by computing a lower bound in the case of a minimization problem. When a complete and feasible solution is met, its objective value can be used as an upper bound to prune nodes having a greater lower bound. Heuristic methods are usually employed to quickly compute such bound. We refer to Section 4 for a review of heuristic methods. *Pruning* can also be performed by applying dominance rules, e.g. to avoid symmetry or unproductive moves.

TS-based methods may differ in the way they explore the solution space. In *Branch & Bound* (B&B) methods, the whole solution space is considered and enumerated, discarding subtrees that are guaranteed to not contain optimal solutions unless an optimal solution has been previously discovered. In contrast, *Beam Search* (BS) explores only a predetermined number of best partial solutions at each level of the tree. *Rake Search*, inspired by BS, performs a breadth-first tree search where the tree is generated level by level. Once the number of nodes reaches a user-defined limit, the tree search is stopped, and each partial solution is used as a starting point of fast construction heuristics. Iterative Deepening techniques (ID-A*, ID-B&B) also define a depth limit for the search tree. When no solution is found, the maximum depth is increased by one, and the search continues. A summary of TS-based methods is available in Table 7. In the rest of this section, we describe TS-based methods in terms of components: branching, lower and upper bounds, and pruning.

Table 7: References for Tree Search-based methods

		Problems			
Reference	Methods	rBRP	uBRP	PMP	Notes
Borjian et al. [8]	A*	✓	✓		
Expósito-Izquierdo et al. [31]	A*	✓	✓		
da Silva Firmino et al. [85]	A*	✓			
Tierney et al. [102]	A* ID-A*			✓	
Bacci et al. [4]	BS	✓			
Ting and Wu [104]	BS	✓			
Wang et al. [112]	BS			✓	
Wu and Ting [116]	BS	✓			
de Melo da Silva et al. [83]	BS B&B	✓			block retrieval problem
Zhang et al. [123]	BS B&B	✓			machine learning
Expósito-Izquierdo et al. [32]	B&B	✓			
Kim and Hong [59]	B&B	✓			
Parreño-Torres et al. [77]	B&B			✓	
Prandtstetter [79]	B&B			✓	
Tanaka and Takii [93]	B&B	✓			
Tanaka and Mizuno [92]	B&B	✓	✓		
Tanaka et al. [95]	B&B			✓	
Ünlüyurt and Aydın [108]	B&B	✓			

Continued on next page

Table 7 (continued)

Reference	Methods	Problems rBRP	uBRP	PMP	Notes
Zhang et al. [124]	B&B			✓	
Zweers et al. [128]	B&B	✓			stochastic BRP with time windows
Quispe et al. [80]	B&B ID-A*	✓			abstraction method
Tricoire et al. [105]	B&B RS		✓		
Bacci et al. [5]	B&C	✓			
Zehendner and Feillet [118]	B&P	✓			
Zehendner and Feillet [119]	B&P	✓			
van Brink and van der Zwaan [15]	B&P			✓	
Feng et al. [35]	DT	✓			stochastic BRP with service times
Galle et al. [42]	DT	✓			
Tierney and Voß [103]	ID-A*			✓	robust PMP
Zhu et al. [127]	ID-A*	✓	✓		
Jin and Yu [52]	ID-B&B			✓	
Tanaka and Tierney [94]	ID-B&B			✓	see also [52]
Tanaka and Voß [96]	ID-B&B	✓	✓		with stowage plan
Bortfeldt and Forster [10]	TS			✓	
Forster and Bortfeldt [38]	TS		✓		
Hottung et al. [47]	TS			✓	machine learning
Ku and Arthanari [64]	TS	✓			abstraction method
Ku and Arthanari [63]	TS	✓			with time windows
Zhang et al. [125]	TS		✓		with batch moves

6.1. Branching

The most intuitive branching procedure is to create a child node at each placement, relocation, or retrieval. Nevertheless, for the BRP, creating child nodes for compound moves instead of single moves has been shown efficient by several authors. In particular, Ünlüyurt and Aydın [108], Borjian et al. [8], Expósito-Izquierdo et al. [32] choose to create child nodes at each retrieval, and Forster and Bortfeldt [38], Zhang et al. [125] generate compound moves using a recursive function. For choosing the next node to branch, the majority of TS-based methods adopt a *Depth-First Search* (DFS) strategy, but some methods such as BS use a *Breadth-First Search* strategy. Tanaka et al. [95] compare different tie-breaking criteria to order the branches for the unrestricted BRP. Tierney et al. [102] estimate a cost for each node and select the node having the minimum value for the PMP. Also for the PMP, Hottung et al. [47] determine in which order nodes should be explored using Deep Neural Networks trained on more than 900,000 optimal solutions. For the BRP, Zhang et al. [123] adopt a DFS strategy and choose the next node based on its upper bound.

6.2. Bounds

We review bounds for each category of stacking problems.

Table 8: Lower bounds for the pure BRP

Name	Restricted	Unrestricted	Duplicate priorities	References
LB1	✓	✓	✓	Kim and Hong [59]
LB2	✓		✓	Zhu et al. [127]
LB3	✓			Zhu et al. [127]
LB4	✓			Tanaka and Takii [93]
LB3e	✓		✓	Tanaka and Takii [93]
LB4e	✓		✓	Tanaka and Takii [93]
ELB4	✓			Zhang et al. [123]
LBN	✓			Borjian et al. [8]
LB-LIS	✓			Quispe et al. [80]
LB-PDB	✓			Quispe et al. [80]
LB2u		✓	✓	Forster and Bortfeldt [38]
LB3u		✓		Tricoire et al. [105]
LBNu		✓		Tanaka and Mizuno [92]
LB4u		✓		Lu et al. [72]
IP-based	✓	✓	✓	Section 3

6.2.1. Restricted BRP

The simplest lower bound (we call it LB1) from [59] is obtained by summing the number of realized relocations and the number of blocking items. In some cases, a blocking item remains blocking after being relocated to any target stack, thus a further relocation cannot be avoided. For the restricted BRP, the lower bound LB2 from [127] improves LB1 by adding these unavoidable relocations. Now, consider a target item, and that we have computed LB2 for the items above it. Discard these items including the target item. We identify the next target item and count the number of unavoidable relocations again as done in LB2. Summing up these lower bounds results in LB3 [127]. Repeating this process N times results in the look-ahead lower bound (we call it LBN) proposed in [8]. In addition, suppose that at least two items are blocking the target item, but there is only one available stack not causing unavoidable relocations for these items. When the topmost item is relocated, its destination stack might not be a good choice anymore for the next items if their departure time is earlier. From this observation, a new lower bound LB4 is deduced from [93]. LB2 and LB3 are only valid for the restricted BRP [127], LB3 and LB4 are only valid with distinct priorities [93]. To overcome the latter limitation, Tanaka and Takii [93] propose an extension of LB3 and LB4, respectively called LB3e and LB4e, valid for duplicate priorities.

The idea is to consider the target items one by one, in a hypothetical layout where all other target items of the same priority and items above them are removed. Note that this can apply to LBN as well. Although LB4 is tighter than LB3, the latter is faster to compute. Quispe et al. [80] introduce two lower bounds LB-LIS and LB-PDB. LB-LIS creates a binary matrix that informs which blocking item remains blocking after being relocated to which stack. From this matrix, they compute a sequence of item priorities and find its longest increasing subsequence to deduce LS-LIS. The second lower bound, LS-PDB, exploits the idea of pattern databases from [64]. Their B&B algorithm was more efficient when using LB-LIS as a lower bound compared to LB3, LB4, and LB-PDB, even though LB4 is tighter. Bacci et al. [3, 4] reinterpret some of the previous lower bounds as solutions of the *Generalized Minimum Blocking Problem* (GMBIP) and deduce a new lower bound (named as UBALB). UBALB is obtained by solving a relaxed GMBIP with a polynomial-time algorithm. Note that $LB1 \leq LB2 \leq LB3 \leq LB4, UBALB$. Experiments show that LB4 is not practical for large instances due to its exponential computational time. Whereas UBALB was marginally looser than LB4, it achieved short computational times. Zhang et al. [123] enhance LB4 (ELB4) by considering that items blocking the target item may affect the relocation of the next target items. Galle et al. [42], Zweers et al. [128] give lower bounds for stochastic variants of the BRP.

6.2.2. Unrestricted BRP

All the former lower bounds (except LB1) apply only for the restricted BRP. For the unrestricted BRP, Forster and Bortfeldt [38] propose an extension (LB2u) increasing LB1 by one move when every blocking item remains blocking after being relocated. Tricoire et al. [105] introduce a generalization of LB2u that we call LB3u, observing that lower-level items may also have to remain blocking after a relocation. LB3u was more accurate than LB2u in certain cases but did not bring significant improvement in most of the cases. Tanaka and Mizuno [92] propose a lower bound (we call it LBNu) that exploits a similar idea to LB2u but also checks items above the target item. Lu et al. [72] review the previous lower bounds and reveal fundamental connections between them. They derive a new lower bound (LB4u) that dominates all the previous ones. Without stack height limits, they observed that the optimality gap of LB4u was nearly twice less than the gap of the second-best lower bound, LBNu. Lower bounds for the pure BRP are summarized in Table 8. For the BRP with batch moves, Zhang et al. [125] give a lower bound extending [10]. da Silva Firmino et al. [85] propose a lower bound of the variant of the restricted BRP minimizing the crane travel distance.

6.2.3. IP-based lower bounds for the BRP

Finally, further lower bounds can be obtained by solving the LP relaxations of BRP-I [19], BRP-II* [32], BRP-II-A [117], BRP-III [78], BRP-m1, BRP-m2 [84], BRP-m3 [72], and CRP-I [40] models. Note that BRP-I, BRP-III, BRP-m1, BRP-m2 and BRP-m3 allow voluntary moves whereas BRP-II*, BRP-II-A and CRP-I forbid them. In [78], BRP-I was found tighter than BRP-III, however, took longer to compute. In [84], BRP-m1 obtained better linear relaxations than BRP-III and BRP-m2. In [40], CRP-I was not found as tight as LB1 and LB3 on average.

6.2.4. PreMarshalling Problem

In the PMP, every misplaced item has to be relocated. Thus, a simple lower bound on the number of relocations (we call it LB^L) is the number of misplaced items [66]. Observe that if all the stacks contain misplaced items, then we must first repair at least one stack to sort the storage area. By adding the minimum number of misplaced items over all stacks, we get LB^Z [15, 124]. If relocating an item implies that an additional relocation is necessary, the move is called an *indirect* relocation. By including indirect relocations, Voß [109] further improves LB^Z. To compute the lower bound LB^{BF}, Bortfeldt and Forster [10] compute LB^Z and add a lower bound on the number of well-placed items that must become misplaced later. Tanaka and Tierney [94] introduce the lower bounds IBF^0 and IBF^1 to improve LB^{BF}. IBF^1 obtains better results by considering special cases when all or most of the stacks contain misplaced items. Tanaka et al. [95] extend IBF^1 with three lower bounds IBF^2, IBF^3, and IBF^4. IBF^2 increases IBF^1 by 1 in certain situations. In turn, IBF^3 improves IBF^2 when $IBF^2 = LB^{BF}$, by taking one more case into consideration. Furthermore, IBF^4 improves IBF^3 when the latter fails, in the same manner. The best performance was obtained with IBF^4, closely followed by IBF^3. IBF^4 is used as a base for computing lower bounds for the PMP minimizing crane times [77]. Finally, other lower bounds can be obtained by solving the LP relaxation of the models presented in Section 3. The tighest LP-based lower bounds may be obtained by the models PMP_{m1} [84], IPS6 [76] and IPCT [77].

6.3. Pruning

In the PMP and the BRP, partial sequences of moves can be eliminated based on dominance properties. Expósito-Izquierdo et al. [33] avoid moves that reverse directly previous moves. Tierney et al. [102] and Tanaka and Mizuno [92] identify several types of dominance properties for the PMP and the BRP, respectively. The first one detects unrelated moves (i.e. not sharing any from or to stacks in common) that lead to the same layout. The second avoids transitive moves, i.e. several moves that can be performed in one single move. The third breaks the symmetry caused by multiple empty stacks. Another one, for the BRP, prevents items to be relocated just before their retrieval. Additionally, Tanaka and Tierney [94] propose a dominance rule for the PMP that breaks symmetry when items of the same priority are relocated. Jin and Yu [52] remark that two dominance rules in [94] cause overpruning. They fix this issue by applying a lexicographic dominance principle, which ensures consistency between dominance rules. Some of the former dominance rules for the PMP are generalized in [95] by introducing the concept of invariant stack. A stack is *invariant* to a given sequence of moves if its layout is the same before and after the latter sequence, and the topmost item before the sequence is not moved by the sequence. This allows breaking symmetry when moves can be indistinctly executed before and after a sequence. The authors also provide a rule using the upper and lower bounds of a node. Parreño-Torres et al. [77] give two dominance criteria for the PMP minimizing the crane time, one breaking symmetry, and one for transitive moves. Hottung et al. [47] use Deep Neural Networks (DNNs) to heuristically determine lower bounds for the PMP and determine which branches should be pruned. To do so, their DNNs are trained on more than 900,000 optimal solutions.

Zhang et al. [123] also use machine-learning for pruning branches, based on a random forest trained on known datasets.

6.4. Abstraction method

To reduce further the search space, Ku and Arthanari [64] introduce an abstraction method applicable to the B&B for the BRP. The idea is to replace the original state space with another (the abstract space) that is easier to search. They observed that equivalent layouts (e.g. having simply permutated stacks) appear repeatedly in different paths of the search tree. Using a database of abstract states, visited nodes are cached by projecting them to their corresponding abstract states. To do so, all empty stacks are removed and the remaining stacks are rearranged in ascending order of the priority in the lowest slot of the stack. This way, redundancy is easier to detect, thus recomputations can be avoided. This abstraction method is exploited in a bidirectional search proposed by [64]. Compared with [59], [19] and [118], results showed superior performance. The abstraction method has been further implemented in [42], and in [80] with new lower bounds, successfully showing significant improvements. In their method combining B&B and Dynamic Programming, Prandtstetter [79] also presented an approach to eliminate redundancy of layouts. They reorder the stacks as in [64] and run a heuristic to determine whether two layouts are equivalent.

6.5. Miscellaneous

Wang et al. [112] solve the PMP with a dummy stack using a Beam Search. Computational times are significantly improved by performing compound moves instead of single moves, with a little cost of solution quality. Tierney and Voß [103] extend the ID-A* from [102] to solve the robust PMP, where unordered stackings are defined by a binary matrix instead of departure times. de Melo da Silva et al. [83] tackle the *Block Retrieval Problem*, a special case of the BRP in which only a subset of the items has to be retrieved, in any order. The primary goal is to minimize the number of relocations of the non-target items. As a second objective, the bi-objective BRTP considers the expected number of relocations for the next retrieval, given probabilities that non-target items will be retrieved before other items. The BRTP is solved by BS and B&B methods.

Tanaka and Voß [96] propose a B&B for the *BRP with a Stowage Plan* (BRP-SP), described in Section 2. They formulate a precedence graph indicating whether an item can be retrieved before another. Three lower bounds inspired by the BRP are used: LB2c and LB2c4c are based on the number of cycles in the precedence graph, LBr is based on a relaxation of the BRP-SP. It is observed that LB2c $\leq$ LB2c4c $\leq$ LBr. Even though LBr requires longer computational time than the other two, it was found more efficient in the B&B.

6.6. Future directions

According to [84], the B&B from [92] is the fastest exact method for the restricted BRP. This is mainly due to the fast lower bounds and many dominance rules applied during the search. Better lower bounds may be obtained for the unrestricted BRP [105]. No good exact method exists for the unrestricted BRP with duplicate priorities, according to [92]. To tackle duplicate priorities, the dominance properties applied to distinct priorities should be modified. Some work could also be done on finding better lower bounds. Tanaka and Voß [96] suggest developing faster lower bounds for the BRP-SP since the best-so-far lower bound becomes intractable for large instances. In particular, the lower bound LB5 from [3] has not been implemented in a B&B algorithm yet. Another idea is to incorporate the abstraction method from [64] to reduce the search space, or a local search algorithm such as in [34] to improve bounds.

Hottung et al. [47], Zhang et al. [123] suggest that Deep Learning is a promising approach for various stacking problems. The approach in [123] may be extended for the unrestricted BRP and with duplicate priorities, for comparison with [92]. Moreover, some performance improvements could be achieved by Reinforcement Learning.

Finally, there is room for developing Tree Search-based methods for variants of stacking problems. Tus et al. [106] propose to investigate A^* or ID-A^* for the 2D-PMP.

7. Concluding Remarks

In this paper, we have investigated the literature on solution methods for solving Block Relocation and PreMarshalling Problems. We distinguished and summarized four categories of methods, and determined which stacking problems they cover. We also suggest directions for future research in each category.

Integer programming formulations are still a promising direction. Existing improvements and preprocessing steps for the BRP could be exploited for other problems such as the PMP. We observe that few constraint programming and column generation approaches have been implemented and compared for solving large instances. Another direction is to incorporate uncertainty in these models.

Nowadays, heuristics and metaheuristics perform well on small and medium instances. However, on large instances, state-of-the-art methods are still far from optimal solutions for problems such as the BRP. Using a post-processing phase such as a local search has been shown very efficient on the BRP. Existing methods can also be extended to cover more realistic constraints or different objective functions.

Besides, developing faster and/or tighter lower bounds for tree search-based methods are believed to be suitable for practical industrial applications. Machine learning techniques such as deep learning are promising for complex applications.

Finally, many ideas and methods mentioned in this survey exploit the inherent structure of stacking problems. Therefore, they could be applied or adapted to a wider range of stacking problems, such as loading problems and simultaneous loading/unloading problems.

References

[1] Akyüz, M.H., Lee, C.Y., 2014. A mathematical formulation and efficient heuristics for the dynamic container relocation problem. Naval Research Logistics (NRL) 61, 101–118. URL: https://doi.org/10.1002%2Fnav.21569, doi:10.1002/nav.21569.

[2] Azab, A., Morita, H., 2022. The block relocation problem with appointment scheduling. European Journal of Operational Research 297, 680–694. URL: https://doi.org/10.1016%2Fj.ejor.2021.06.007, doi:10.1016/j.ejor.2021.06.007.

[3] Bacci, T., Mattia, S., Ventura, P., 2018. A new lower bound for the block relocation problem, in: Lecture Notes in Computer Science. Springer International Publishing. volume 11184, pp. 168–174. URL: https://doi.org/10.1007%2F978-3-030-00898-7_10, doi:10.1007/978-3-030-00898-7_10.
[4] Bacci, T., Mattia, S., Ventura, P., 2019. The bounded beam search algorithm for the block relocation problem. Computers & Operations Research 103, 252–264. URL: https://doi.org/10.1016%2Fj.cor.2018.11.008, doi:10.1016/j.cor.2018.11.008.
[5] Bacci, T., Mattia, S., Ventura, P., 2020. A branch and cut algorithm for the restricted block relocation problem. European Journal of Operational Research 287, 452–459. URL: https://doi.org/10.1016%2Fj.ejor.2020.05.029, doi:10.1016/j.ejor.2020.05.029.
[6] Boge, S., Goerigk, M., Knust, S., 2020. Robust optimization for premarshalling with uncertain priority classes. European Journal of Operational Research 287, 191–210. URL: https://doi.org/10.1016%2Fj.ejor.2020.04.049, doi:10.1016/j.ejor.2020.04.049.
[7] Borgman, B., van Asperen, E., Dekker, R., 2010. Online rules for container stacking. OR Spectrum 32, 687–716. URL: https://doi.org/10.1007%2Fs00291-010-0205-4, doi:10.1007/s00291-010-0205-4.
[8] Borjian, S., Galle, V., Manshadi, V.H., Barnhart, C., Jaillet, P., 2015a. Container relocation problem: Approximation, asymptotic, and incomplete information. CoRR abs/1505.04229. URL: http://arxiv.org/abs/1505.04229.
[9] Borjian, S., Manshadi, V.H., Barnhart, C., Jaillet, P., 2015b. Managing relocation and delay in container terminals with flexible service policies. CoRR abs/1503.01535. URL: http://arxiv.org/abs/1503.01535.
[10] Bortfeldt, A., Forster, F., 2012. A tree search procedure for the container pre-marshalling problem. European Journal of Operational Research 217, 531–540. URL: https://doi.org/10.1016%2Fj.ejor.2011.10.005, doi:10.1016/j.ejor.2011.10.005.
[11] Bortfeldt, A., Wäscher, G., 2013. Constraints in container loading – a state-of-the-art review. European Journal of Operational Research 229, 1 – 20. URL: http://www.sciencedirect.com/science/article/pii/S037722171200937X, doi:https://doi.org/10.1016/j.ejor.2012.12.006.
[12] Bowes, P., 2019. Pitney Bowes Parcel Shipping Index. URL: https://www.pitneybowes.com/us/shipping-index.html.
[13] Boysen, N., Briskorn, D., Meisel, F., 2017. A generalized classification scheme for crane scheduling with interference. European Journal of Operational Research 258, 343–357. URL: https://doi.org/10.1016%2Fj.ejor.2016.08.041, doi:10.1016/j.ejor.2016.08.041.
[14] Boysen, N., Stephan, K., 2016. A survey on single crane scheduling in automated storage/retrieval systems. European Journal of Operational Research 254, 691–704. URL: https://doi.org/10.1016%2Fj.ejor.2016.04.008, doi:10.1016/j.ejor.2016.04.008.
[15] van Brink, M., van der Zwaan, R., 2014. A branch and price procedure for the container premarshalling problem, in: Algorithms - ESA 2014. Springer Berlin Heidelberg. volume 8737, pp. 798–809. URL: https://doi.org/10.1007%2F978-3-662-44777-2_66, doi:10.1007/978-3-662-44777-2_66.
[16] 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, 412–430. URL: https://doi.org/10.1016%2Fj.ejor.2013.10.054, doi:10.1016/j.ejor.2013.10.054.
[17] 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, 1–13. URL: https://doi.org/10.1016%2Fj.ejor.2013.11.023, doi:10.1016/j.ejor.2013.11.023.
[18] Caserta, M., Schwarze, S., Voß, S., 2009. A new binary description of the blocks relocation problem and benefits in a look ahead heuristic, in: Evolutionary Computation in Combinatorial Optimization. Springer Berlin Heidelberg. volume 5482, pp. 37–48. URL: https://doi.org/10.1007%2F978-3-642-01009-5_4, doi:10.1007/978-3-642-01009-5_4.
[19] 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. URL: https://doi.org/10.1016%2Fj.ejor.2011.12.039, doi:10.1016/j.ejor.2011.12.039.
[20] Caserta, M., Schwarze, S., Voß, S., 2020. Container rehandling at maritime container terminals: A literature update, in: Operations Research/Computer Science Interfaces Series. Springer International Publishing, pp. 343–382. URL: https://doi.org/10.1007%2F978-3-030-39990-0_16, doi:10.1007/978-3-030-39990-0_16.
[21] Caserta, M., Voß, S., 2009a. A corridor method-based algorithm for the pre-marshalling problem, in: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., Machado, P. (Eds.), Applications of Evolutionary Computing, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 788–797.
[22] Caserta, M., Voß, S., 2009b. Corridor selection and fine tuning for the corridor method, in: Lecture Notes in Computer Science. Springer Berlin Heidelberg. volume 5851, pp. 163–175. URL: https://doi.org/10.1007%2F978-3-642-11169-3_12, doi:10.1007/978-3-642-11169-3_12.
[23] Caserta, M., Voß, S., Sniedovich, M., 2011. Applying the corridor method to a blocks relocation problem. OR Spectrum 33, 915–929. URL: https://doi.org/10.1007/s00291-009-0176-5, doi:10.1007/s00291-009-0176-5.
[24] Covic, F., 2018a. Container Handling in Automated Yard Blocks Based on Time Information. Ph.D. thesis. University of Hamburg.
[25] Covic, F., 2018b. A literature review on container handling in yard blocks, in: Lecture Notes in Computer Science. Springer International Publishing. volume 11184, pp. 139–167. URL: https://doi.org/10.1007%2F978-3-030-00898-7_9, doi:10.1007/978-3-030-00898-7_9.
[26] Database, S.U.R., 2018. Data for container relocation problem. URL: https://research.sabanciuniv.edu/34326/.
[27] Dayama, N.R., Ernst, A., Krishnamoorthy, M., Narayanan, V., Rangaraj, N., 2016. New models and algorithms for the container stack rearrangement problem by yard cranes in maritime ports. EURO Journal on Transportation and Logistics 6, 307–348. URL: https://doi.org/10.1007%2Fs13676-016-0098-8, doi:10.1007/s13676-016-0098-8.
[28] Deb, K., 1999. An introduction to genetic algorithms. Sadhana 24, 293–315. URL: https://doi.org/10.1007%2Fbf02823145, doi:10.1007/bf02823145.
[29] ElWakil, M., Gheith, M., Eltawil, A., 2019. A new simulated annealing based method for the container relocation problem, in: 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), IEEE. pp. 1432–1437. URL: https://doi.org/10.1109%2Fcodit.2019.8820687, doi:10.1109/codit.2019.8820687.
[30] Eskandari, H., Azari, E., 2015. Notes on mathematical formulation and complexity considerations for blocks relocation problem. Scientia Iranica 22, 2722–2728. URL: http://scientiairanica.sharif.edu/article_3815.html.
[31] 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, 327–343. URL: https://doi.org/10.1016%2Fj.aei.2014.03.003, doi:10.1016/j.aei.2014.03.003.
[32] Expósito-Izquierdo, C., Melián-Batista, B., Moreno-Vega, J.M., 2015. An exact approach for the blocks relocation problem. Expert Systems with Applications 42, 6408–6422. URL: https://doi.org/10.1016%2Fj.eswa.2015.04.021, doi:10.1016/j.eswa.2015.04.021.
[33] Expósito-Izquierdo, C., Melián-Batista, B., Moreno-Vega, M., 2012. Pre-marshalling problem: Heuristic solution method and instances generator. Expert Systems with Applications 39, 8337 – 8349. URL: http://www.sciencedirect.com/science/article/pii/S0957417412002151, doi:https://doi.org/10.1016/j.eswa.2012.01.187.
[34] Feillet, D., Parragh, S.N., Tricoire, F., 2019. A local-search based heuristic for the unrestricted block relocation problem. Computers & Operations Research 108, 44–56. URL: https://doi.org/10.1016%2Fj.cor.2019.04.006, doi:10.1016/j.cor.2019.04.006.
[35] Feng, Y., Song, D.P., Li, D., Zeng, Q., 2020. The stochastic container relocation problem with flexible service policies. Transportation Research Part B: Methodological 141, 116–163. URL: https:

//doi.org/10.1016%2Fj.trb.2020.09.006, doi:10.1016/j.trb.2020.09.006.
[36] Feo, T.A., Resende, M.G.C., 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133. URL: https://doi.org/10.1007%2Fbf01096763, doi:10.1007/bf01096763.
[37] Forster, F., Bortfeldt, A., 2012a. A tree search heuristic for the container retrieval problem, in: Klatte, D., Lüthi, H.J., Schmedders, K. (Eds.), Operations Research Proceedings 2011, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 257–262.
[38] Forster, F., Bortfeldt, A., 2012b. A tree search procedure for the container relocation problem. Computers & Operations Research 39, 299–309. URL: https://doi.org/10.1016%2Fj.cor.2011.04.004, doi:10.1016/j.cor.2011.04.004.
[39] Galle, V., 2017. StochasticCRP GitHub repository. URL: https://github.com/vgalle/StochasticCRP.
[40] 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, 467 – 477. URL: http://www.sciencedirect.com/science/article/pii/S0377221717310640, doi:https://doi.org/10.1016/j.ejor.2017.11.053.
[41] Galle, V., Boroujeni, S.B., Manshadi, V., Barnhart, C., Jaillet, P., 2016. An average-case asymptotic analysis of the container relocation problem. Operations Research Letters 44, 723–728. URL: https://doi.org/10.1016%2Fj.orl.2016.08.006, doi:10.1016/j.orl.2016.08.006.
[42] Galle, V., Boroujeni, S.B., Manshadi, V.H., Barnhart, C., Jaillet, P., 2017. The stochastic container relocation problem. CoRR abs/1703.04769. URL: http://arxiv.org/abs/1703.04769.
[43] Gatto, M., Maue, J., Mihalák, M., Widmayer, P., 2009. Shunting for dummies: An introductory algorithmic survey, in: Robust and Online Large-Scale Optimization. Springer Berlin Heidelberg. volume 5868, pp. 310–337. URL: https://doi.org/10.1007%2F978-3-642-05465-5_13, doi:10.1007/978-3-642-05465-5_13.
[44] Gheith, M., Eltawil, A.B., Harraz, N.A., 2015. Solving the container pre-marshalling problem using variable length genetic algorithms. Engineering Optimization 48, 687–705. URL: https://doi.org/10.1080%2F0305215x.2015.1031661, doi:10.1080/0305215x.2015.1031661.
[45] Hamburg, U., 2014. BRP instances (CVS dataset). URL: https://www.bwl.uni-hamburg.de/en/iwi/forschung/projekte/dataprojekte/brp-instances-caserta-etal-2012.zip.
[46] Helo, P., Paukku, H., Sairanen, T., 2018. Containership cargo profiles, cargo systems, and stowage capacity: key performance indicators. Maritime Economics & Logistics 23, 28–48. URL: https://doi.org/10.1057%2Fs41278-018-0106-z, doi:10.1057/s41278-018-0106-z.
[47] Hottung, A., Tanaka, S., Tierney, K., 2020. Deep learning assisted heuristic tree search for the container pre-marshalling problem. Computers & Operations Research 113, 104781. URL: https://doi.org/10.1016%2Fj.cor.2019.104781, doi:10.1016/j.cor.2019.104781.
[48] Hottung, A., Tierney, K., 2016. A biased random-key genetic algorithm for the container pre-marshalling problem. Computers & Operations Research 75, 83–102. URL: https://doi.org/10.1016%2Fj.cor.2016.05.011, doi:10.1016/j.cor.2016.05.011.
[49] Huang, S.H., Lin, T.H., 2012. Heuristic algorithms for container pre-marshalling problems. Computers & Industrial Engineering 62, 13 – 20. URL: http://www.sciencedirect.com/science/article/pii/S0360835211002385, doi:https://doi.org/10.1016/j.cie.2011.08.010.
[50] Iris, C., Pacino, D., 2015. A survey on the ship loading problem, in: Lecture Notes in Computer Science. Springer International Publishing. volume 9335, pp. 238–251. URL: https://doi.org/10.1007%2F978-3-319-24264-4_17, doi:10.1007/978-3-319-24264-4_17.
[51] Ji, M., Guo, W., Zhu, H., Yang, Y., 2015. Optimization of loading sequence and rehandling strategy for multi-quay crane operations in container terminals. Transportation Research Part E: Logistics and Transportation Review 80, 1–19. URL: https://doi.org/10.1016%2Fj.tre.2015.05.004, doi:10.1016/j.tre.2015.05.004.
[52] Jin, B., Yu, M., 2021. Note on the dominance rules in the exact algorithm for the container pre-marshalling problem by tanaka & tierney (2018). European Journal of Operational Research 293, 802–

807. URL: https://doi.org/10.1016%2Fj.ejor.2020.12.041, doi:10.1016/j.ejor.2020.12.041.

[53] Jin, B., Zhu, W., Lim, A., 2015. Solving the container relocation problem by an improved greedy look-ahead heuristic. European Journal of Operational Research 240, 837–847. URL: https://doi.org/10.1016%2Fj.ejor.2014.07.038, doi:10.1016/j.ejor.2014.07.038.

[54] Jovanovic, R., Tanaka, S., Nishi, T., Voß, S., 2018. A GRASP approach for solving the blocks relocation problem with stowage plan. Flexible Services and Manufacturing Journal 31, 702–729. URL: https://doi.org/10.1007%2Fs10696-018-9320-3, doi:10.1007/s10696-018-9320-3.

[55] Jovanovic, R., Tuba, M., Voß, S., 2015. A multi-heuristic approach for solving the pre-marshalling problem. Central European Journal of Operations Research 25, 1–28. URL: https://doi.org/10.1007%2Fs10100-015-0410-y, doi:10.1007/s10100-015-0410-y.

[56] Jovanovic, R., Tuba, M., Voß, S., 2019. An efficient ant colony optimization algorithm for the blocks relocation problem. European Journal of Operational Research 274, 78 – 90. URL: http://www.sciencedirect.com/science/article/pii/S0377221718308208, doi:https://doi.org/10.1016/j.ejor.2018.09.038.

[57] Jovanovic, R., Voß, S., 2014. A chain heuristic for the blocks relocation problem. Computers & Industrial Engineering 75, 79–86. URL: https://doi.org/10.1016%2Fj.cie.2014.06.010, doi:10.1016/j.cie.2014.06.010.

[58] Jovanović, R., 2016. Benchmark data sets for the blocks relocation problem with stowage plan (brlp) URL: http://mail.ipb.ac.rs/~rakaj/brlp/brlp.htm.

[59] Kim, K.H., Hong, G.P., 2006. A heuristic rule for relocating blocks. Computers & Operations Research 33, 940–954. URL: https://doi.org/10.1016%2Fj.cor.2004.08.005, doi:10.1016/j.cor.2004.08.005.

[60] Kim, Y., Kim, T., Lee, H., 2016. Heuristic algorithm for retrieving containers. Computers & Industrial Engineering 101, 352–360. URL: https://doi.org/10.1016%2Fj.cie.2016.08.022, doi:10.1016/j.cie.2016.08.022.

[61] Kizilay, D., Eliiyi, D.T., 2020. A comprehensive review of quay crane scheduling, yard operations and integrations thereof in container terminals. Flexible Services and Manufacturing Journal 33, 1–42. URL: https://doi.org/10.1007%2Fs10696-020-09385-5, doi:10.1007/s10696-020-09385-5.

[62] Ku, D., 2014. Container Relocation Problem with Time Windows (CRPTW). URL: http://crp-timewindow.blogspot.com.

[63] Ku, D., Arthanari, T.S., 2016a. Container relocation problem with time windows for container departure. European Journal of Operational Research 252, 1031–1039. URL: https://doi.org/10.1016%2Fj.ejor.2016.01.055, doi:10.1016/j.ejor.2016.01.055.

[64] Ku, D., Arthanari, T.S., 2016b. On the abstraction method for the container relocation problem. Computers & Operations Research 68, 110–122. URL: https://doi.org/10.1016%2Fj.cor.2015.11.006, doi:10.1016/j.cor.2015.11.006.

[65] de La Laguna, U., 2011. Pre-Marshalling Problem Bay Generator. URL: https://sites.google.com/site/gciports/premarshalling-problem/bay-generator.

[66] Lee, Y., Chao, S.L., 2009. A neighborhood search heuristic for pre-marshalling export containers. European Journal of Operational Research 196, 468–475. URL: https://doi.org/10.1016%2Fj.ejor.2008.03.011, doi:10.1016/j.ejor.2008.03.011.

[67] Lee, Y., Hsu, N.Y., 2007. An optimization model for the container pre-marshalling problem. Computers & Operations Research 34, 3295–3313. URL: https://doi.org/10.1016%2Fj.cor.2005.12.006, doi:10.1016/j.cor.2005.12.006.

[68] Lee, Y., Lee, Y.J., 2010. A heuristic for retrieving containers from a yard. Computers & Operations Research 37, 1139 – 1147. URL: http://www.sciencedirect.com/science/article/pii/S0305054809002433, doi:https://doi.org/10.1016/j.cor.2009.10.005.

[69] Lehnfeld, J., Knust, S., 2014. Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. European Journal of Operational Research 239, 297–312. URL: https://doi.org/10.1016%2Fj.ejor.2014.03.011, doi:10.1016/j.ejor.2014.03.011.

[70] Lin, D.Y., Lee, Y.J., Lee, Y., 2015. The container retrieval problem with respect to relocation.

Transportation Research Part C: Emerging Technologies 52, 132–143. URL: https://doi.org/10.1016%2Fj.trc.2015.01.024, doi:10.1016/j.trc.2015.01.024.

[71] López-Plata, I., Expósito-Izquierdo, C., Lalla-Ruiz, E., Melián-Batista, B., Moreno-Vega, J.M., 2017. Minimizing the waiting times of block retrieval operations in stacking facilities. Computers & Industrial Engineering 103, 70–84. URL: https://doi.org/10.1016%2Fj.cie.2016.11.015, doi:10.1016/j.cie.2016.11.015.

[72] Lu, C., Zeng, B., Liu, S., 2020. A study on the block relocation problem: Lower bound derivations and strong formulations. IEEE Transactions on Automation Science and Engineering , 1829–1853URL: https://doi.org/10.1109%2Ftase.2020.2979868, doi:10.1109/tase.2020.2979868.

[73] Luo, J., Wu, Y., Halldorsson, A., Song, X., 2011. Storage and stacking logistics problems in container terminals. OR Insight 24, 256–275. URL: https://doi.org/10.1057%2Fori.2011.10, doi:10.1057/ori.2011.10.

[74] Murty, K.G., Liu, J., wah Wan, Y., Linn, R., 2005. A decision support system for operations in a container terminal. Decision Support Systems 39, 309–332. URL: https://doi.org/10.1016%2Fj.dss.2003.11.002, doi:10.1016/j.dss.2003.11.002.

[75] Olsen, M., Gross, A., 2014. Average case analysis of blocks relocation heuristics, in: Lecture Notes in Computer Science. Springer International Publishing. volume 8760, pp. 81–92. URL: https://doi.org/10.1007%2F978-3-319-11421-7_6, doi:10.1007/978-3-319-11421-7_6.

[76] Parreño-Torres, C., Alvarez-Valdes, R., Ruiz, R., 2019. Integer programming models for the pre-marshalling problem. European Journal of Operational Research 274, 142–154. URL: https://doi.org/10.1016%2Fj.ejor.2018.09.048, doi:10.1016/j.ejor.2018.09.048.

[77] Parreño-Torres, C., Alvarez-Valdes, R., Ruiz, R., Tierney, K., 2020. Minimizing crane times in pre-marshalling problems. Transportation Research Part E: Logistics and Transportation Review 137, 101917. URL: https://doi.org/10.1016%2Fj.tre.2020.101917, doi:10.1016/j.tre.2020.101917.

[78] 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. URL: https://doi.org/10.1016%2Fj.ejor.2013.05.037, doi:10.1016/j.ejor.2013.05.037.

[79] Prandtstetter, M., 2013. A dynamic programming based branch-and-bound algorithm for the container pre-marshalling problem. (PhD diss). AIT Austrian Institute of Technology Technical .

[80] Quispe, K.E.Y., Lintzmayer, C.N., Xavier, E.C., 2018. An exact algorithm for the blocks relocation problem with new lower bounds. Computers & Operations Research 99, 206 – 217. URL: http://www.sciencedirect.com/science/article/pii/S0305054818301710, doi:https://doi.org/10.1016/j.cor.2018.06.021.

[81] Raggl, S., Beham, A., Tricoire, F., Affenzeller, M., 2018. Solving a real world steel stacking problem. International Journal of Service and Computing Oriented Manufacturing 3, 94–108.

[82] Rendl, A., Prandtstetter, M., 2013. Constraint models for the container pre-marshaling problem. ModRef 2013, 12th.

[83] de Melo da Silva, M., Erdoğan, G., Battarra, M., Strusevich, V., 2018a. The block retrieval problem. European Journal of Operational Research 265, 931–950. URL: https://doi.org/10.1016%2Fj.ejor.2017.08.048, doi:10.1016/j.ejor.2017.08.048.

[84] de Melo da Silva, M., Toulouse, S., Calvo, R.W., 2018b. A new effective unified model for solving the pre-marshalling and block relocation problems. European Journal of Operational Research 271, 40–56. URL: https://doi.org/10.1016%2Fj.ejor.2018.05.004, doi:10.1016/j.ejor.2018.05.004.

[85] da Silva Firmino, A., de Abreu Silva, R.M., Times, V.C., 2016. An exact approach for the container retrieval problem to reduce crane's trajectory, in: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), IEEE. URL: https://doi.org/10.1109%2Fitsc.2016.7795667, doi:10.1109/itsc.2016.7795667.

[86] da Silva Firmino, A., de Abreu Silva, R.M., Times, V.C., 2019. A reactive grasp metaheuristic for the container retrieval problem to reduce crane's working time. Journal of Heuristics 25, 141–173. URL: https://doi.org/10.1007/s10732-018-9390-0, doi:10.1007/s10732-018-9390-0.

[87] Sniedovich, M., Voß, S., 2006. The corridor method: a dynamic programming inspired metaheuristic. Control and Cybernetics 35, 551–578.
[88] Stahlbock, R., Voß, S., 2008. Operations research at container terminals: a literature update. OR Spectrum 30, 1–52. URL: https://doi.org/10.1007%2Fs00291-007-0100-9, doi:10.1007/s00291-007-0100-9.
[89] Świeboda, J., Zając, M., 2016. Analysis of reshuffling cost at a container terminal, in: Dependability Engineering and Complex Systems. Springer International Publishing. volume 470, pp. 491–503. URL: https://doi.org/10.1007%2F978-3-319-39639-2_43, doi:10.1007/978-3-319-39639-2_43.
[90] Tanaka, S., 2021. Block (Container) Pre-Marshalling Problem. URL: https://sites.google.com/site/shunjitanaka/pmp.
[91] Tanaka, S., 2022. Block (Container) Relocation Problem. URL: https://sites.google.com/site/shunjitanaka/brp.
[92] Tanaka, S., Mizuno, F., 2018. An exact algorithm for the unrestricted block relocation problem. Computers & Operations Research 95, 12 – 31. URL: http://www.sciencedirect.com/science/article/pii/S0305054818300583, doi:https://doi.org/10.1016/j.cor.2018.02.019.
[93] Tanaka, S., Takii, K., 2016. A faster branch-and-bound algorithm for the block relocation problem. IEEE Transactions on Automation Science and Engineering 13, 181–190. URL: https://doi.org/10.1109%2Ftase.2015.2434417, doi:10.1109/tase.2015.2434417.
[94] 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, 165–180. URL: https://doi.org/10.1016%2Fj.ejor.2017.05.046, doi:10.1016/j.ejor.2017.05.046.
[95] Tanaka, S., Tierney, K., Parreño-Torres, C., Alvarez-Valdes, R., Ruiz, R., 2019. A branch and bound approach for large pre-marshalling problems. European Journal of Operational Research 278, 211–225. URL: https://doi.org/10.1016%2Fj.ejor.2019.04.005, doi:10.1016/j.ejor.2019.04.005.
[96] Tanaka, S., Voß, S., 2019. An exact algorithm for the block relocation problem with a stowage plan. European Journal of Operational Research 279, 767–781. URL: https://doi.org/10.1016%2Fj.ejor.2019.06.014, doi:10.1016/j.ejor.2019.06.014.
[97] Tanaka, S., Voß, S., 2022. An exact approach to the restricted block relocation problem based on a new integer programming formulation. European Journal of Operational Research 296, 485–503. URL: https://doi.org/10.1016%2Fj.ejor.2021.03.062, doi:10.1016/j.ejor.2021.03.062.
[98] Tang, L., Jiang, W., Liu, J., Dong, Y., 2014. Research into container reshuffling and stacking problems in container terminal yards. IIE Transactions 47, 751–766. URL: https://doi.org/10.1080%2F0740817x.2014.971201, doi:10.1080/0740817x.2014.971201.
[99] Tang, L., Liu, J., Rong, A., Yang, Z., 2002. Modelling and a genetic algorithm solution for the slab stack shuffling problem when implementing steel rolling schedules. International Journal of Production Research 40, 1583–1595. URL: https://doi.org/10.1080/00207540110110118424, doi:10.1080/00207540110110118424.
[100] Tang, L., Ren, H., 2010. Modelling and a segmented dynamic programming-based heuristic approach for the slab stack shuffling problem. Computers & Operations Research 37, 368 – 375. URL: http://www.sciencedirect.com/science/article/pii/S0305054809001531, doi:https://doi.org/10.1016/j.cor.2009.05.011.
[101] Tang, L., Zhao, R., Liu, J., 2012. Models and algorithms for shuffling problems in steel plants. Naval Research Logistics (NRL) 59, 502–524. URL: https://doi.org/10.1002%2Fnav.21503, doi:10.1002/nav.21503.
[102] Tierney, K., Pacino, D., Voß, S., 2016. Solving the pre-marshalling problem to optimality with a* and IDA*. Flexible Services and Manufacturing Journal 29, 223–259. URL: https://doi.org/10.1007%2Fs10696-016-9246-6, doi:10.1007/s10696-016-9246-6.
[103] Tierney, K., Voß, S., 2016. Solving the robust container pre-marshalling problem, in: Lecture Notes in Computer Science. Springer International Publishing. volume 9855, pp. 131–145. URL: https://doi.org/10.1007%2F978-3-319-44896-1_9, doi:10.1007/978-3-319-44896-1_9.
[104] Ting, C.J., Wu, K.C., 2017. Optimizing container relocation operations at container yards with beam

search. Transportation Research Part E: Logistics and Transportation Review 103, 17–31. URL: https://doi.org/10.1016%2Fj.tre.2017.04.010, doi:10.1016/j.tre.2017.04.010.

[105] Tricoire, F., Fechter, J., Beham, A., 2017. New insights on the block relocation problem. Computers & Operations Research 89, 127–139. URL: https://doi.org/10.1016%2Fj.cor.2017.08.010, doi:10.1016/j.cor.2017.08.010.

[106] Tus, A., Rendl, A., Raidl, G.R., 2015. Metaheuristics for the two-dimensional container pre-marshalling problem, in: Lecture Notes in Computer Science. Springer International Publishing. volume 8994, pp. 186–201. URL: https://doi.org/10.1007%2F978-3-319-19084-6_17, doi:10.1007/978-3-319-19084-6_17.

[107] UNCTAD, 2019. Review of Maritime Transport 2019. United Nations Conference on Trade and Development. URL: http://unctad.org/en/PublicationsLibrary/rmt2019_en.pdf.

[108] Ünlüyurt, T., Aydın, C., 2012. Improved rehandling strategies for the container retrieval process. Journal of Advanced Transportation 46, 378–393. URL: https://doi.org/10.1002%2Fatr.1193, doi:10.1002/atr.1193.

[109] Voß, S., 2012. Extended mis-overlay calculation for pre-marshalling containers, in: Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 86–91. URL: https://doi.org/10.1007%2F978-3-642-33587-7_6, doi:10.1007/978-3-642-33587-7_6.

[110] Voß, S., Schwarze, S., 2019. A note on alternative objectives for the blocks relocation problem, in: Lecture Notes in Computer Science. Springer International Publishing. volume 7555, pp. 101–121. URL: https://doi.org/10.1007%2F978-3-030-31140-7_7, doi:10.1007/978-3-030-31140-7_7.

[111] Wan, Y., Liu, J., Tsai, P.C., 2009. The assignment of storage locations to containers for a container stack. Naval Research Logistics 56, 699–713. URL: https://doi.org/10.1002%2Fnav.20373, doi:10.1002/nav.20373.

[112] Wang, N., Jin, B., Lim, A., 2015. Target-guided algorithms for the container pre-marshalling problem. Omega 53, 67–77. URL: https://doi.org/10.1016%2Fj.omega.2014.12.002, doi:10.1016/j.omega.2014.12.002.

[113] Wang, N., Jin, B., Zhang, Z., Lim, A., 2017. A feasibility-based heuristic for the container pre-marshalling problem. European Journal of Operational Research 256, 90–101. URL: https://doi.org/10.1016%2Fj.ejor.2016.05.061, doi:10.1016/j.ejor.2016.05.061.

[114] World Steel Association, 2019. World Steel in Figures 2019. worldsteel. URL: https://www.worldsteel.org/media-centre/press-releases/2019/world-steel-in-figures-2019.html.

[115] Wu, K., Ting, C., 2012. Heuristic approaches for minimizing reshuffle operations at container yard, in: Proceedings of the Asia Pacific industrial engineering & management systems conference, pp. 1407–51.

[116] 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, pp. 15–17.

[117] 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, 415–422. URL: https://doi.org/10.1016%2Fj.ejor.2015.03.032, doi:10.1016/j.ejor.2015.03.032.

[118] Zehendner, E., Feillet, D., 2012. Column generation for the container relocation problem, in: International Annual Conference of the German Operations Research Society, Hannover, Germany. URL: https://hal-emse.ccsd.cnrs.fr/emse-00730787.

[119] Zehendner, E., Feillet, D., 2014. A branch and price approach for the container relocation problem. International Journal of Production Research 52, 7159–7176. URL: https://doi.org/10.1080%2F00207543.2014.965358, doi:10.1080/00207543.2014.965358.

[120] Zehendner, E., Feillet, D., Jaillet, P., 2017. An algorithm with performance guarantee for the online container relocation problem. European Journal of Operational Research 259, 48–62. URL: https://doi.org/10.1016%2Fj.ejor.2016.09.011, doi:10.1016/j.ejor.2016.09.011.

[121] Zeng, Q., Feng, Y., Yang, Z., 2019. Integrated optimization of pickup sequence and container rehandling based on partial truck arrival information. Computers & Industrial Engineering 127, 366–382. URL: https://doi.org/10.1016%2Fj.cie.2018.10.024, doi:10.1016/j.cie.2018.10.024.

[122] Zhang, C., 2000. Resource planning in container storage yard. Ph.D. thesis. Hong Kong University of

Science and Technology.

[123] Zhang, C., Guan, H., Yuan, Y., Chen, W., Wu, T., 2020. Machine learning-driven algorithms for the container relocation problem. Transportation Research Part B: Methodological 139, 102–131. URL: https://doi.org/10.1016%2Fj.trb.2020.05.017, doi:10.1016/j.trb.2020.05.017.

[124] Zhang, R., Jiang, Z.Z., Yun, W.Y., 2015a. Stack pre-marshalling problem: a heuristic-guided branch-and-bound algorithm. International Journal of Industrial Engineering 22, 509–523.

[125] Zhang, R., Liu, S., Kopfer, H., 2015b. Tree search procedures for the blocks relocation problem with batch moves. Flexible Services and Manufacturing Journal 28, 397–424. URL: https://doi.org/10.1007%2Fs10696-015-9229-z, doi:10.1007/s10696-015-9229-z.

[126] Zhang, W., Lin, Y., Ji, Z., Zhang, G., 2008. Review of containership stowage plans for full routes. Journal of Marine Science and Application 7, 278–285. URL: https://doi.org/10.1007%2Fs11804-008-7087-8, doi:10.1007/s11804-008-7087-8.

[127] 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. URL: https://doi.org/10.1109%2Ftase.2012.2198642, doi:10.1109/tase.2012.2198642.

[128] Zweers, B.G., Bhulai, S., van der Mei, R.D., 2020. Optimizing pre-processing and relocation moves in the stochastic container relocation problem. European Journal of Operational Research 283, 954–971. URL: https://doi.org/10.1016%2Fj.ejor.2019.11.067, doi:10.1016/j.ejor.2019.11.067.