Chapter 16
Container Rehandling at Maritime Container Terminals: A Literature Update

Marco Caserta, Silvia Schwarze, and Stefan Voß

Abstract This chapter provides an updated survey on rehandling of containers at maritime container terminals. In particular, we review contributions with a particular focus on post-stacking situations, i.e., problems arising after the stacking area has already been arranged. Three types of post-stacking problems have been identified, namely (1) the re-marshalling problem, (2) the pre-marshalling problem, and (3) the relocation problem. This research area has received an increasing attention since the first version of this contribution appeared in 2011. Within this update, we discuss recent developments presented in literature. In particular, available solution approaches from the fields of exact and (meta-)heuristic methods are given and benchmark datasets are summarized. Moreover, an overview on extensions of post-stacking problems and according solution methods are discussed.

16.1 Introduction

Container terminals can be seen as buffers within larger logistic chains encompassing worldwide distribution systems. The major purpose of using container terminals is to serve as transshipment points. Container terminals are used as temporary storage points for containers, such that, e.g., unloading operations from a vessel and loading operations onto a train or a truck need not be synchronized.

Broadly speaking, a container terminal can be divided into three major areas: The quayside, i.e., the side in which vessels are berthed, the landside, i.e., the side in which other means of transportation operate (trucks, trains), and the container yard, i.e., the area in which containers are stored for future operations. The management of a container terminal yard is of paramount importance in determining

M. Caserta
IE University and IE Business School, Madrid, Spain
e-mail: marco.caserta@ie.edu

S. Schwarze (✉) · S. Voß
Institute of Information Systems, University of Hamburg, Hamburg, Germany
e-mail: silvia.schwarze@uni-hamburg.de; stefan.voss@uni-hamburg.de

J. W. Böse (ed.), *Handbook of Terminal Planning*, Operations Research/Computer Science Interfaces Series 64, https://doi.org/10.1007/978-3-030-39990-0_16

the efficiency of a port. Due to the fierce competition in the global market, container terminal operators are forced to increase the efficiency of storage yard operations, in order to capture and retain customers.

As pointed out by a number of authors, e.g., Choe et al. (2011), Park et al. (2009), Stahlbock and Voß (2008), and Zhang et al. (2003), some performance indicators of container terminal efficiency are: (1) the vessel berthing time, and (2) the throughput of the quay cranes, i.e., the efficiency in unloading/loading containers from/to vessels. While such key performance indicators can be improved at the strategic level by adopting new technologies and structures, such as new equipment or the terminal layout design, at the operational level, a proven means to enhance the efficiency of container terminal operations is the optimization of the way in which such operations are carried out.

While such key performance indicators can be improved through the use of new technology, such as, e.g., new equipment, terminal layout re-design, etc., the efficiency of container terminal operations can also be enhanced by *optimizing* the way in which such operations are carried out. More specifically, a great deal of attention should be devoted to the definition of efficient container stacking policies.

As highlighted in Dekker et al. (2006), stacking can be seen as a three-level problem. *Strategic* stacking decisions must be made with respect to the layout of the container yard, the type of equipment, and the design of the container terminal itself. *Tactical* stacking decisions are concerned with decisions that affect capacity in the medium term, e.g., whether a pre-stacking area should be used, whether pre-arrangement policies should be implemented (re-marshalling, pre-marshalling, etc.). Finally, *operational* stacking decisions deal with the identification of slots to be assigned to containers, the rehandling of containers within the yard, the berth allocation problem, the assignment of equipment to tasks, the definition of a loading/unloading (stowage) plan, etc. In this chapter, we deal with operational stacking decisions, with a special focus on offering a comprehensive overview of published work dealing with operations that are carried out upon an existing stack or set of stacks of containers.

These types of problems, presented under the label "marshalling problems at container terminal yards," have received a great deal of attention in the last years. Two recent surveys, i.e., Lehnfeld and Knust (2014) and Carlo et al. (2014), have proposed classification schemes for the broad set of optimization problems arising at container terminal yards. More specifically, Carlo et al. (2014) classifies storage yard operations at container terminals along a number of dimensions, i.e., (i) yard design, (ii) storage space assignment, (iii) material handling equipment, (iv) container reshuffling optimization. In turn, this fourth dimension, i.e., optimization of container reshuffling, is subdivided into four main problem typologies:

(iv.1) selection of storage location;
(iv.2) retrieval and reshuffling, as in the *blocks relocation problem*;

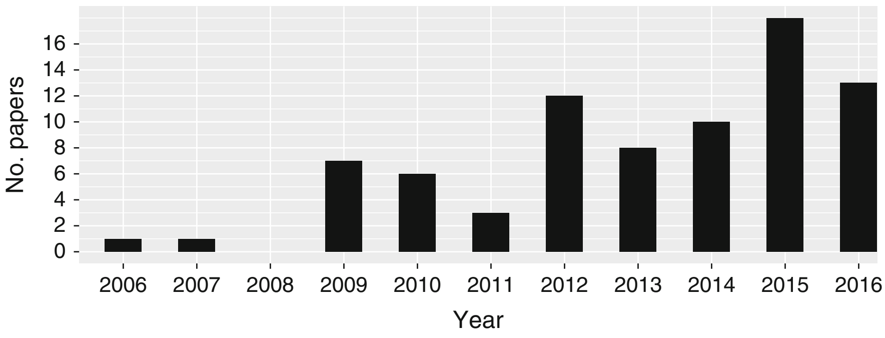

Fig. 16.1 No. of papers on post-stacking problems per year

(iv.3) *pre-marshalling* operations;
(iv.4) *re-marshalling* operations.

Borrowing from this classification, two more survey papers covering typologies (iv.2)–(iv.4) above, i.e., retrieval and marshalling operations, have appeared in recent years, i.e., Caserta et al. (2011a) and Dayama et al. (2016).
In this chapter, we build on the work presented in Caserta et al. (2011a) and provide an up-to-date overview of optimization approaches for marshalling and retrieval operations at container terminal yards. The motivation for this literature update springs from the increasing number of papers on marshalling and stacking problems appeared in recent years. To provide a glimpse of how active the research community in this field has been, Table 16.1 and Fig. 16.1 summarize the number of publications that appeared since 2006.[1] From Table 16.1, we can observe that 79 papers have been published in the last 11 years, of which 61 in the last 5 years alone. From the operational point of view, we focus on three problems:

- the *B*locks *R*elocation *P*roblem (BRP), also known as the *C*ontainer *R*elocation *P*roblem (CRP);
- the *C*ontainer *P*re-*M*arshalling *P*roblem (CPMP), i.e., intra-bay marshalling;
- and the *C*ontainer *R*e-*M*arshalling *P*roblem (CRMP), i.e., intra-block marshalling.

From the solution approach point of view, we hereby collect contributions on optimization methods for any of the three aforementioned problems. Broadly speaking, we identify the following solution approaches across the three problems:

- greedy heuristics, i.e., rules-of-thumb employed to select the next best move;
- metaheuristics, i.e., master mechanisms that coordinate the use of a pool of heuristic rules;

[1] In Table 16.1 and Fig. 16.1, a single paper being published in 2017 (see Wang et al. 2017) is assigned to 2016, the year of the online-first publication.

Table 16.1 Number of publications on post-stacking problems from 2006 to 2016

Year	Total no. papers	BRP	BRP extension	Re-marshalling	Pre-marshalling	Survey
2016	12	4	3		5	1
2015	18	6	4	3	5	
2014	10	5	1		3	1
2013	8	2	2	2	2	
2012	12	4	4		4	
2011	3	1		1		1
2010	6	3	2	1		
2009	7	3	1	1	2	
2008	0					
2007	1				1	
2006	1	1				
Total	79	29	17	8	22	3

Sources: Google Scholar, Scopus, ProQuesti, and EBSCOhost

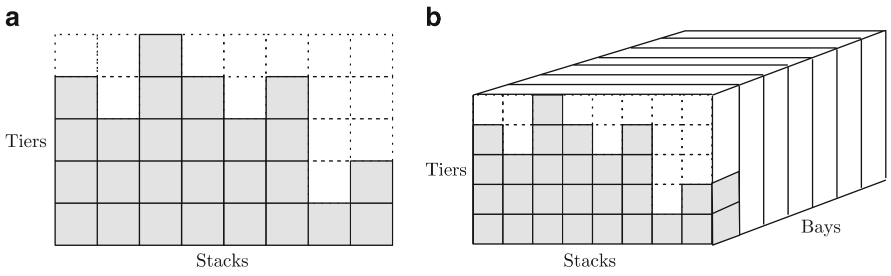

Fig. 16.2 Container bay and block

- exact approaches, i.e., approaches that guarantee the optimality of the provided solution; and, more recently,
- robust approaches, i.e., formulations and solution approaches that attempt to capture part of the uncertainty of the problem, thus providing a solution that should be of good quality even when some disruptive events occur.

In the sequel, we follow the typical terminology adopted in the context of container terminal operations. We indicate with the term *bay* a two-dimensional portion of the container yard, made up by a number of *stacks*, i.e., the width, and *tiers*, i.e., the height, as illustrated in Fig. 16.2a. A *block* is a set of consecutive bays, as presented in Fig. 16.2b. Finally, a *container yard* is made up by a set of blocks.
In addition, we assume that a *priority* (exact or estimated) is associated with each container in the stacking area. Priorities account for a number of different factors, such as (1) category: e.g., containers with the same priority might belong to the same category and could be piled up on top of each other; (2) departure time: e.g., containers with earlier departure time will have higher priority than containers

with later departure time; (3) size and weight: e.g., typically, containers with higher weight are not stored on top of containers with lower weight, in order to respect overall ship balancing constraints. It is worth mentioning that, since the precise departure date of a container might not be known when the container reaches the stacking area, it might be the case that *estimated* priority values are assigned to containers. A new line of research focuses on the definition of *robust optimization* methods to tackle this type of uncertainty.

The two terms *retrieving* and *rehandling* are used to describe movement of containers. More specifically, the term *retrieving* is used to indicate a movement of a container from the bay to the vessel. Conversely, we use the term *rehandling* to indicate a move of a container within the container yard, both in the case of intra-bay or intra-block movements. In all cases, we consider the layout of the stacking area as given, i.e., the position and priority, exact or estimated, of each container in the stacking area is known. Therefore, our interest is not centered on finding effective stacking policies. Rather, given a stacking area, we wish to determine how containers should be rehandled or retrieved in order to minimize the total number of unproductive movements.

The structure of the chapter is as follows: In Sect. 16.2, the complexity of post-stacking problems is discussed. Afterwards, Sect. 16.3 is devoted to the presentation of marshalling problems, aimed at reshuffling the storage area in order to eliminate, or reduce, the total number of future rehandling. Section 16.4 deals with a different type of problem, the blocks relocation problem. Section 16.5 constitutes a bridge between rehandling problems at maritime container terminals and similar problems arising in different realms. Some references to related work in other application domains are provided in this section. Finally, Sect. 16.6 concludes offering a brief overview of the current status in the container handling discipline along with a glimpse of future challenges and opportunities.

16.2 Complexity of Post-stacking Problems

Before surveying the available work in the field of post-stacking problems, we provide a brief overview on complexity issues:

First, the complexity of the BRP is stated as $\mathcal{NP}$-hard, see Caserta et al. (2012), by a reduction from the *M*utual *E*xclusion *S*cheduling (MES) problem on permutation graphs, proved to be $\mathcal{NP}$-hard in Jansen (2003). It is moreover shown that a particular case of the BRP, known as the *restricted* BRP (see, Sect. 16.4.1) is still $\mathcal{NP}$-hard. Moreover, regarding alternative objectives, the BRP minimizing the crane movement time, generalizes the BRP and, therefore, is $\mathcal{NP}$-hard, too; see, Schwarze and Voß (2015). Moreover, recent papers have addressed the issue of computational complexity of different variants of the marshalling problem. More precisely:

- The re-marshalling problem: Caserta et al. (2011a) proved that the problem is $\mathcal{NP}$-hard by reduction from the BRP.
- The pre-marshalling problem with fixed height: van Brink and van der Zwaan (2014) proved that both the *priority stacking* and the *configuration stacking*, i.e., a variant in which a pre-specified bay layout must be reached, are $\mathcal{NP}$-hard when the height is fixed $H \geq 6$. All reductions are from the MES problem on permutation graphs.
- The pre-marshalling problem with unlimited height: In van Brink and van der Zwaan (2014), a proof that the *priority stacking* with unlimited height is $\mathcal{NP}$-hard is presented. Again, all reductions are from the MES problem on permutation graphs. However, there is no formal proof for the complexity of the *configuration stacking* problem with unlimited height. To the best of the authors' knowledge, as of today, this seems to be an open question.

16.3 Container Marshalling Problems

In this section, we focus on the pre-marshalling and re-marshalling problems. In line with Carlo et al. (2014), we make the following assumptions:

A1-D. The container retrieval sequence, based on container priorities, is known in advance. This also implies that no further containers are expected to arrive; or

A1-S. The exact container retrieval sequence is not known, since the precise priority values of containers are not determined yet. This might be due, e.g., to uncertainty of the arrival time of collecting vessels. We thus assume that containers are assigned a time interval within which they are expected to leave.

A2. The reshuffling of containers is limited to the same bay (*pre-marshalling*) or the same block (*re-marshalling*);

Assumptions A1-D and A1-S are mutually exclusive and define the deterministic and stochastic versions of the corresponding marshalling problem, respectively. More rigorously, following the accepted terminology from the literature, we define the two marshalling problems as follows:

Pre-marshalling The pre-marshalling problem is concerned with finding an optimal, i.e., shortest, sequence of reshufflings that reorganizes the containers within a bay in such a way that, for a known retrieval sequence, no further reshuffling is required. This problem is also called *intra-bay re-marshalling*, since the containers are reshuffled within the same bay.

Intra-bay re-marshalling, or pre-marshalling, is motivated by the use of a specific technology. As pointed out by Lee and Chao (2009) and Lee and Hsu (2007), yards that use rail mounted gantry cranes as major container handling equipment typically solve the marshalling problem at bay level. For safety reasons, in some terminals where access of containers to and from the block is usually from the side, a gantry crane is not moved from one bay or block to another while carrying a container. Therefore, in those terminals, to move a container from one bay to another, it would be necessary to temporarily unload the container from the crane, put it on a truck, move the truck and, possibly, the empty crane to the target bay, pickup the container from the truck with the crane, and store the container within the target bay. This operation is time consuming and, therefore, it is avoided whenever possible. This consideration motivates the study, from a practical perspective, of the intra-bay pre-marshalling problem. The goal of the pre-marshalling problem is, therefore, to rehandle containers within the same bay in order to eliminate (or minimize) future rehandling while minimizing the total number of rehandlings during the pre-marshalling process itself. Two observations are in place here:

- The pre-marshalling problem does not require to reach a pre-specified bay configuration. In other words, as long as no further reshuffles will be needed during the subsequent loading/unloading phase, the bay configuration is considered optimal. Thus, a variant of the classical pre-marshalling problem can be envisioned, in which a pre-specified bay configuration must be reached, not only in terms of containers priority, as in the classical pre-marshalling, but also in terms of specific layout of the bay. The only authors that take into account this variant of the pre-marshalling are Lee and Hsu (2007) and van Brink and van der Zwaan (2014). Lee and Hsu (2007) defined a *M*ixed-*I*nteger *P*rogramming (MIP) model for the classical pre-marshalling problem that can be tailored to achieve a pre-specified bay configuration with the addition of a set of side constraints. Along the same line, in van Brink and van der Zwaan (2014) the difference between the *priority stacking* and the *configuration stacking* pre-marshalling problems is highlighted. They use the term priority stacking pre-marshalling problem to identify the "classical" version of the pre-marshalling, while the configuration stacking pre-marshalling identifies the variant in which a pre-specified bay layout must be reached.
- As long as the relocation of containers occurs within the same bay, only the number of crane movements is to be minimized. In other words, the distance covered by the crane is negligible and, therefore, is not taken into account in the optimization process. On the other hand, if containers need to be moved from one bay to another within the same block, or from one block to another, then some "transportation costs," typically proportional to the distance covered, should be taken into account.

Re-marshalling The re-marshalling problem is concerned with finding the minimum length sequence of container movements aimed at retrieving containers from a source bay and position them to a target bay (or bays) assigned to a specific vessel (or vessels) in such a way that no further reshuffling will be needed. This type of problem is also called *intra-block re-marshalling*, since movements of containers typically occur within the same block. These types of problems are not just a simple extension of the pre-marshalling problem, since cranes interference and transportation costs should now be taken into account.

As illustrated in Table 16.1 and Fig. 16.1, in the last years, a growing body of literature on optimization approaches for marshalling problems has been developing. More precisely, with respect to the pre-marshalling and the re-marshalling problems described above, we found 22 papers on the former problem and 8 papers on the latter problem, without including survey papers mentioning marshalling problems but not specifically dealing with the aforementioned problems.

From the solution approach point of view, these papers can be classified in the following four groups. In this regard, a comprehensive list of papers classified along the related dimensions is provided in Table 16.2:

- *Greedy, target-guided heuristic approaches*: These approaches typically define greedy scores to select a target container and a target stack among a pool of candidates. The targets are chosen according to the value of the greedy score and the moves needed to relocate the target container to the target stack are carried out. The types of moves defined can be *single moves*, i.e., at each step only one container is moved to a new position, or *compound moves*, in which cases all the relocations needed to achieve the target (moving the target containers to the target stack) are carried out. Examples of these approaches are Bortfeldt and Forster (2012), Expósito-Izquierdo et al. (2012), Jovanovic et al. (2017), among others.
- *Metaheuristic approaches*: These algorithms make use of the greedy rules and moves described above within the context of a metaheuristic, e.g., the corridor method (Caserta and Voß 2009b), simulated annealing (Choe et al. 2011), genetic algorithm (Gheith et al. 2016; Hottung and Tierney 2016), the pilot method (Tus et al. 2015), among others. In some instances, exact approaches, e.g., dynamic programming, are used in a metaheuristic fashion, in line with the definition of "matheuristics" (See, e.g., Caserta and Voß 2009b).
- *Exact approaches*: These are algorithms aimed at finding an optimal solution that exploit (1) Mathematical programming techniques, e.g., branch-and-bound (Zhang et al. 2015), dynamic programming (Prandtstetter 2013), branch-and-price (van Brink and van der Zwaan 2014), network optimization (Lee and Hsu 2007); (2) Constraint programming, e.g., Rendl and Prandtstetter (2013); (3) Search algorithms, e.g., A* and IDA* (Tierney et al. 2017).
- *Robust approaches*: This new line of research is currently represented by two papers, i.e., Tierney and Voß (2016) and Rendl and Prandtstetter (2013). Robust optimization attempts to capture the uncertainty of real-world marshalling problems due to potential delays of vessels arrivals. Since the arrival time of vessels at a berth is only an "expected" time, this uncertainty affects the priority value of containers which, consequently, should be dealt with as if it were a stochastic value. A common approach to deal with uncertainty is to treat container priority values as intervals, rather than deterministic parameters.

Table 16.2 List of papers on the CPMP, from 2007 to 2016

Paper	Approach	Method	Benchmark	Comment
Wang et al. (2017)	Heuristic	Target-guided/Greedy	CV, BF	Feasibility of a configuration is tested
Tierney and Voß (2016)	Robust/Exact	CP, IDA*	TV	Robust CPMP relaxed via CP and solved using IDA*
Hottung and Tierney (2016)	Metaheuristic	bRKGA	CV, BF	Learning mechanism, parameters fine tuning
Gheith et al. (2016)	Metaheuristic	GA	LH, LC, CV	Variable length of the chromosome
Tierney et al. (2017)	Exact	A*, IDA*	CV, BF, IG	Problem specific symmetry breaking rules for A* and IDA*
Wang et al. (2015)	Metaheuristic	BS	LC, CV, BF, WJL	Target-guided heuristic. Use of "giant" moves. Dummy stacks
Tus et al. (2015)	Metaheuristic	Pilot+AS	TRR	2D version, with lateral access to the bay
Tierney and Malitsky (2015)	Metaheuristic	Algorithm Selection	CV, BF, IG	Latent Features Analysis
Zhang et al. (2015)	Exact	B&B+Heuristic	CV, ZJY, LH1	Limited to Nr. Stacks $\times$ Max height ≤ 35
Ren and Zhang (2015)	Heuristic	Greedy Rules	LH, LC	Three-step iterative heuristic
van Brink and van der Zwaan (2014)	Exact	B&P	BZ	Two pre-marshalling variants: Priority *vs.* configuration stacking
Jovanovic et al. (2017)	Heuristic	Target Driven/Greedy	IG	Comparison of different heuristics within the framework of LPFH
Gheith et al. (2014)	Heuristic	Greedy Rules	LH	Three-step iterative algorithm
Rendl and Prandtstetter (2013)	Exact+Robust	CP	IG	Robust version defined using priority ranges
Prandtstetter (2013)	Exact	DP+BVB	EMM	Initialization via LPFH. Heuristic DP is also proposed
Bortfeldt and Forster (2012)	Heuristic	Tree Search	LH, LC, CV, BF	Computation of lower bound for number of moves
Expósito-Izquierdo et al. (2012)	Heuristic	Target Driven/Greedy	EMM	Instance generator IG. A* for small-size instances

(continued)

Table 16.2 (continued)

Paper	Approach	Method	Benchmark	Comment
Huang and Lin (2012)	Heuristics	Greedy Rules	LH	Labeling algorithms for two versions of the CPMP
Voß (2012)	Heuristic	Bound	LC	Computation of lower bound for number of moves
Lee and Chao (2009)	Heuristic	NS	LC-1, LC-2, LC-3	Neighborhood Search+IP Formulation
Caserta and Voß (2009b)	Metaheuristic	CM	LC, CV	Dynamic Programming + Corridor Method
Lee and Hsu (2007)	Exact	MCNF	LH	For both priority and configuration stacking. A heuristic is also proposed

A*	Algorithm for graph traversal
AS	Ant system
B&B	Branch-and-bound algorithm
B&P	Branch-and-price algorithm
bRKGA	Biased random key genetic algorithm
BS	Beam search algorithm
CM	Corridor method
CP	Constraint programming
DP	Dynamic programming
GA	Genetic algorithm
IDA*	Iterative deepening A* algorithm
LPFH	Lowest priority first heuristic
MCNF	Minimum cost network flow model
NS	Neighborhood search
Pilot	Pilot method

16.3.1 Container Pre-marshalling Problem

In this section, we present a brief overview of each paper dealing with the CPMP appeared in the years 2007–2016. A comprehensive list of publications, along with the solution approach, the benchmark instances used, and a short comment on each paper can be found in Table 16.2. In addition, we provide a list of publicly available benchmark instances for the CPMP in Table 16.3. These instances have been used by a number of authors to test the effectiveness of their algorithms and constitute a large library of instances, with different degrees of complexity, which could be used to test future work on the CPMP. We hereby present the contributions appeared in peer-reviewed outlets, starting with the most recent ones and going backward to the first work on the CPMP.

Wang et al. (2017) introduce a target-guided heuristic for the CPMP. A target-driven heuristic is a heuristic in which targets, i.e., containers, are moved to specific slots one at a time. Thus, at each step, the heuristic identifies the target, i.e., the container, to be moved and the destination stack. This relocation of the target container to the destination stack is called "valid task," and the way in which valid tasks are identified is based on a set of greedy rules. However, a difference that stands out between the approach presented in this paper and other target-driven rules in the literature is that, while in general target containers are selected in a predetermined order, e.g., according to the container priorities, in this approach the order is not fixed beforehand but, rather, dynamically determined during the search phase. In other words, the target container is selected depending on, among other things, the current layout of the bay. Finally, once a valid move is identified, the target container is fixed at the destination stack and will no longer be moved during the optimization process. The algorithm repeats the aforementioned steps until all the containers are fixed. To speed up the selection of moves, the authors propose a novel state feasibility test. Prior to moving a container to a slot, the feasibility of the resulting state, i.e., bay configuration, is tested and, if a move leads to an infeasible layout, that move is discarded. This feasibility check allows to explore large portions of the solution space in an efficient fashion. The authors tested their algorithm on two benchmark sets from the literature, CV and BF,[2] and the results obtained show the effectiveness of the proposed scheme.

Hottung and Tierney (2016) present a metaheuristic that employs the *b*iased *R*andom *K*ey *G*enetic *A*lgorithm (bRKGA) framework to guide a three-step iterative heuristic. The bRKGA is in charge of two major tasks: On the one hand, the metaheuristic learns and fine-tunes the decision as to which container to move and which stack to use in each step. That is, the rating mechanisms employed to judge the quality of candidate moves are guided by the bRKGA via the encoding and decoding of a part of the chromosome, thus allowing successive generations of the bRKGA to select better moves. Since the mechanism has some learning

[2]See Table 16.3 for the source of the benchmark and a description of the same.

Table 16.3 Benchmark instances for the CPMP

Name	Source	Nr. Instances	Description
LH	Lee and Hsu (2007)	2	The first one with $S = 6$ and $H = 4$, and the second with $S = 12$ and $H = 5$
CV	Caserta and Voß (2009b)	840	Twentyone groups of 40 instances each, with $H \in \{3, 4, 5, 6, 10\}$ and $S \in \{3, \dots, 10\}$
LC	Lee and Chao (2009)	12	Three groups, of 1, 1, and 10 instances each, with $S \in \{10, 12\}$ and $H \in \{5, 6\}$
BF	Bortfeldt and Forster (2012)	640	Thirtytwo groups of 20 instances each, with $S \in \{16, 20\}$ and $H \in \{5, 8\}$
EMM	Expósito-Izquierdo et al. (2012)	3600	Three groups of 1200 instances each, with $S = 4$ and $H \in \{4, 7, 10\}$
IG	Expósito-Izquierdo et al. (2012)	–	Instance generator for the CPMP
BZ	van Brink and van der Zwaan (2014)	960	Fortyeight groups of 20 instances each, with $S \in \{3, 5, 7, 9\}$ and $H \in \{4, 6\}$
WJL	Wang et al. (2015)	1080	Thirtysix groups of 30 instances each, with $S \in \{6, 8, 10, \}$ and $H \in \{5, 8\}$. Designed with "dummy stack."
TRR	Tus et al. (2015)	600	Twevl groups of 50 instances each, with $S \in \{4, 6, 8, 10, 12, 14\}$ and $H = 4$. Instances obtained using IG
ZJY	Zhang et al. (2015)	100	Five groups of 20 instances each, with $S \in \{6, 7, 8, 9\}$ and $H \in \{4, 5\}$
TV	Tierney and Voß (2016)	900	Ninety groups of 10 instances each, with $S \in \{5, 8, 10\}$ and $H \in \{3, 4, 5, 6, 10\}$

features, it benefits from solving the same instance over and over, i.e., along multiple generations. On the other hand, the same chromosome devotes some of the genes to the encoding and decoding of values of some algorithmic parameters. Consequently, a second task accomplished by the bRKGA is that of fine-tuning the algorithmic parameters. The algorithm has been tested on the CV and BF instances and, at the time of writing, together with the Beam Search algorithm of Wang et al. (2015), the results reported in this paper are the best presented in the literature.

Gheith et al. (2016) discuss a solution approach that employs genetic algorithms. The encoding is such that each move in the bay is specified via two consecutive genes, one indicating the stack the container is removed from and the next specifying the receiving stack. A chromosome thus encodes the full set of moves of a solution to the pre-marshalling problem. The interesting variation introduced in the paper is that, since the exact number of moves required to reach the final configuration is not known, a variable length *G*enetic *A*lgorithm (GA) is used instead. The length of the chromosome is thus proportional to the number of moves required by a given solution. The fitness value associated to a chromosome is composed of two terms, i.e., the number of moves required (proportional to the chromosome length), and the number of mis-overlays of the final configuration, i.e., the bay configuration obtained after implementing all the movements encoded in the chromosome (ideally equal to zero for a proper pre-marshalling solution). The algorithm has been tested on instances LH, LC, and CV (see Table 16.3).

Tierney et al. (2017) present an exact algorithm based on a problem specific implementation of the A* and IDA* algorithms. The authors model the CPMP as a graph, in which the tree structure used to capture the problem is as follows: The root node is associated to the initial bay configuration; each branch of the tree leads to a node associated to a bay configuration that can be reached from the current state via a single move, i.e., relocating only one container. The leaves of the tree correspond to final solutions. At each node a cost is computed. Such cost is the sum of two terms, i.e., the cost of reaching the solution associated to that node, and the cost of completing such a solution, i.e., to reach the closest leave. Lower bounds for the latter term are obtained using the lower bound method proposed in Voß (2012) and Bortfeldt and Forster (2012). At each node, a branch for each possible container move is created. Problem specific symmetry breaking rules have been designed to speed up the search process and to prune dominated branches of the tree. The authors evaluated their approach on instances CV, BF, and a new set of randomly generated instances obtained using the instance generator of Expósito-Izquierdo et al. (2012). The proposed algorithm (IDA*) was able to solve over 500 previously unsolved instances to optimality.

Tierney and Voß (2016) discuss a robust variant of the CPMP, in which the priority of containers is not deterministically known. Rather, a time interval within which the container must be retrieved is provided. This problem is labeled *R*obust *C*ontainer *P*re-*M*arshalling *P*roblem (RCPMP). The authors first find a relaxation of the RCPMP solving a binary constraint satisfaction problem, which takes as input a "blocking matrix" and provides as output a deterministic CPMP, in which container priorities have been fixed respecting the blocking matrix structure. This deterministic CPMP is a relaxation of the original robust problem and is fed as input to the IDA* algorithm of Tierney et al. (2017), which, in turn, provides an optimal solution. Interestingly, the authors prove that a reasonable lower bound for the RCPMP can be found using a lower bound for the CPMP, i.e., the relaxed, deterministic version. The authors tested their algorithm on 900 new randomly generated instances, labeled TV in Tables 16.2 and 16.3, and compared their approach with the only available algorithm dealing with the RCPMP, that of Rendl and Prandtstetter (2013).

Wang et al. (2015) present a target-guided heuristic to tackle the standard CPMP and a variant of the same problem, called the *C*ontainer *P*re-*M*arshalling *P*roblem with *D*ummy *S*tack. The new variant of the CPMP arises from the observations that some block layouts at container terminals have a transfer line parallel to the block itself, as opposed to having transfer lines at both ends of the block. This lateral transfer line can be used by the gantry crane operating on a bay. Therefore, the pre-marshalling problem can be redefined as having an extra "dummy" stack, which can be used to temporarily store containers during the reshuffling operations of the pre-marshalling task. The only caution that must be taken is that the dummy stack must be emptied at the end of the pre-marshalling work. The authors label this variant of the pre-marshalling as CPMPDS. The key idea of the target heuristic is to fix containers to a certain position in a descending order of priorities. Containers are relocated using both compound moves, called *giant moves*, as well as single moves, called *baby moves*. The greedy heuristics presented in the paper are finally embedded into a metaheuristic scheme, the beam search, which allows to escape from suboptimal solutions. The proposed algorithm is tested on a large pool of instances, namely LC, CV, BF, and a newly generated set of random instances, called WJL (see Table 16.3), specifically designed for the CPMPDS. The results reported in this paper for the beam search algorithm are, at the time of writing, the best in the literature, along with those of Hottung and Tierney (2016).

Another variant of the CPMP is presented in Tus et al. (2015). These authors consider the case of small-medium size container terminals, in which, rather than gantry cranes, reach stackers are used. Reach stackers are forklifts that can only access the top containers of the leftmost and rightmost stacks of a container bay. They named this variant of the standard pre-marshalling problem the *2-D*imensional *C*ontainer *P*re-*M*arshalling *P*roblem (2D-CPMP). The authors adapt a lowest priority first heuristic, initially designed for the CPMP, to this variant of the pre-marshalling. Next, they embed this heuristic within two metaheuristics, the Pilot method and a Max-Min Ant System. To test the effectiveness of the proposed scheme, the authors generated a new set of instances using the instance generator of Expósito-Izquierdo et al. (2012). These instances are now available online and are labeled TRR (see Table 16.3). Their empirical analysis shows that the Min-Max Ant System is the best performing algorithm among those proposed in this paper, and that the difference with the other schemes is statistically significant.

An original approach to the CPMP is provided in Tierney and Malitsky (2015). They use algorithm selection to find the best performing algorithm for each instance. More specifically, four parameterizations of the A* and IDA* algorithms of Tierney et al. (2017) are used to form a pool of solvers. With respect to the instance pool, they use instances from CV, BG (see Table 16.3), and some newly randomly generated instances obtained using the instance generator of Expósito-Izquierdo et al. (2012). Each instance is characterized by a set of features, both observable and latent. These features are evaluated using *C*ost-*S*ensitive *H*ierarchical *C*lustering (CSHC). The performance of the portfolio obtained using CSHC is then compared with that of the best single solver, and the virtual best portfolio, i.e., a portfolio that always selects the best algorithm. The authors conclude discussing the importance of enriching the instance description with the use of latent features which, in turn, prove beneficial in the algorithm selection phase.

Zhang et al. (2015) present an exact approach for the CPMP. They design a heuristic-guided branch-and-bound approach, which effectively solves medium-size instances to optimality. The authors state that the algorithm requires an acceptable amount of running time as long as the product of the number of stacks with the maximum height ($S \times H$) is around 35. The problem is framed in the context of a branch-and-bound tree, in which the root node corresponds to the initial layout, each node of the tree is an intermediate layout, and each leave is a final solution. The role of the guiding heuristic is to generate a set of potential branches at each node. They also present an approach to compute a valid lower bound for the number of relocations required and they point out that such lower bound is looser than the one presented in Bortfeldt and Forster (2012), but easier to compute. The lower bound is of paramount importance in pruning the branches of the tree at each node, thus allowing for a more efficient exploration of the branch-and-bound tree. They tested their algorithm on a new randomly generated set of small-medium size instances, called ZJY (see Table 16.3), as well as on the small instances from CV and they were able to achieve an optimal solution for most of these instances in a reasonable amount of computational time.

Ren and Zhang (2015) design a three-step rule-based iterative algorithm. The first step, called local optimization, move ill-placed containers, i.e., containers creating mis-overlays in the current stack, to stacks with zero mis-overlays. The first step ends when moves of this type are no longer available. The second step aims at emptying one stack of the bay. Greedy rules are used to select the stack to be emptied and the destination stacks of containers removed from the emptying stack. Finally, the third stage takes care of refilling the empty stack. Again, heuristic rules are used to prevent deadlocks and to identify the relocated containers. The proposed algorithm has been tested on only three benchmark instances, two from LH and one from LC.

van Brink and van der Zwaan (2014) present an exact algorithm for two versions of the pre-marshalling problem, the priority stacking and the configuration stacking. The former describes the case in which no final specific bay layout is required, as long as no container with low priority is left on top of containers with higher priority. Conversely, the second problem describes the case in which a pre-defined bay layout should be reached, i.e., each container should be placed in a specific position in the bay. The exact method is based on branch-and-price and column generation. The problem is formulated as an integer linear program, and the task of generating new columns, i.e., a sequence of moves, with negative reduced cost is almost equivalent to finding a maximum weight independent set in a circle graph, which is polynomially solvable using dynamic programming. Thus, at each node of the tree, a series of linear programming relaxations with the addition of new columns is solved, until no further columns with negative reduced cost can be added. A lower bound is then used to either prune the branch of the tree or to create further branches. Another valuable result presented in this paper is related to the proof of complexity of the priority stacking and configuration stacking pre-marshalling problems with fixed height. The authors proved that both versions of the CPMP are $\mathscr{NP}$-hard, as discussed in Sect. 16.2 of this paper.

Jovanovic et al. (2017) revisit the *L*owest *P*riority *F*irst *H*euristic (LPFH) presented in Expósito-Izquierdo et al. (2012) and modify each of the four basic components of such heuristic. The key idea is to identify the best heuristic to be used at each stage of the algorithm. The authors point out that, given a pool of competing heuristics for a given task, their performance is highly dependent on the features and properties of the instance at hand. However, since establishing a correlation between instance features and heuristic performance is often difficult, their approach is to test all the heuristics available for the task and select the best performing one. More precisely, at each stage of the LPFH, a pool of heuristics is used to come up with different solutions. A look-ahead mechanism and a backtracking procedure are employed to avoid reaching infeasible bay configurations. The authors tested the proposed method on instances obtained using the instance generator of Expósito-Izquierdo et al. (2012) and the results prove that the proposed algorithm outperforms the original LPFH. In the concluding remarks, the authors point out that (1) no heuristic outperforms the other on a complete set of instances, and, connected with this, (2) the performance of each heuristic is dependent on the features and characteristics of the instance at hand. These final remarks are in direct connection with the findings of Tierney and Malitsky (2015).

Gheith et al. (2014) proposed a rule-based heuristic, composed of three main steps: (1) sort container groups according to the frequency of mis-overlays; (2) find a destination stack employing a number of heuristic rules, (3) move the target container to the destination stack, again employing a number of heuristic rules. The three-step algorithm is iteratively applied until no further mis-overlays are present. The algorithm has been tested on instance LH, and on three randomly generated instances. Thus, a comparison of this heuristic with other approaches from the literature is difficult to carry out.

Rendl and Prandtstetter (2013) take a different approach to the CPMP, in which they formulate and solve the problem employing *C*onstraint *P*rogramming (CP). They iteratively try to solve the CP model in exactly k steps, i.e., number of relocations. The initial value of k is found computing a valid lower bound as in Bortfeldt and Forster (2012) and, at each iteration, k is increased by one if no solution could be found. Thus, the first solution returned is an optimal solution to the CPMP. Two sets of variables are employed in the CP model: The first set defines bay configurations, i.e., the layout of the bay after a certain number of steps; the second set is used to keep track of the moves performed at each point in time. Logical constraints ensure the feasibility of each intermediate configuration and drive the search toward a desired final layout. In addition, a specialized search heuristic, applied on the bay variables is employed to evaluate all the possible moves based on the current bay configuration. Another interesting contribution of this paper is related to the presentation of a robust variant of the CPMP. The authors point out that, in real-world settings, the arrival time of vessels is far from certain. Thus, in reality, only the *expected* arrival time of a vessel is known. The implication is that, since container priorities are based on the exact arrival time of a vessel, the final priority of a container is also uncertain. The goal is thus to produce a final bay layout that is *robust* with respect to vessel delays and container priority variations. More precisely, rather than dealing with a specific priority value, each container should be associated to a *priority range* $\{h, \ldots, l\}$, where h is the highest priority that could be associated to a container, i.e., the earliest possible time that a container will be collected, and l is the lowest possible priority of the same container. The authors modified and adapted the CP formulation to deal with the robust variant of the CPMP. Both models are tested on a set of instances produced using the instance generator of Expósito-Izquierdo et al. (2012).

Prandtstetter (2013) presents an exact approach for the CPMP. The key idea is related to the design of a *D*ynamic *P*rogramming (DP) scheme, which is then embedded into a branch-and-bound framework. To further shrink the DP tree, the author developed a method that allows to recognize the equivalence of DP states: Equivalent DP states should be evaluated only once and, therefore, when an equivalent state is reached, the corresponding branch of the tree can be pruned. In the branch-and-bound scheme, a lower bound is computed at each node using the method of Bortfeldt and Forster (2012) and, together with the upper bound value, allow to further prune the tree. In addition, to further reduce the size of the three, the author introduces a heuristic evaluation of equivalence of two states. Such evaluation of equivalence is “heuristic” in the sense that, while it further shrinks the state space explored by the DP scheme, it does not guarantee that (optimal) states will not be missed. The different variants of the proposed scheme have been tested on the benchmark instances EMM from Expósito-Izquierdo et al. (2012). The DP scheme embedded into the branch-and-bound scheme was able to solve to optimality a large number of instances from the EMM dataset within a maximum running time of 3600 s.

Bortfeldt and Forster (2012) present a tree search heuristic procedure, effectively coupled with the computation of a tight lower bound on the number of moves required to reach the final bay layout, given the current bay configuration. The use of such lower bound is paramount in pruning branches of the tree, thus making the tree search algorithm very effective, even when dealing with large instances. In the tree, the root node corresponds to the initial bay layout, while the leaves of the tree correspond to final configurations. Each node in the tree defines an intermediate state, reachable from its predecessor via a *compound move*, i.e., a sequence of relocations. The procedure was tested on a large set of benchmark instances, namely

LH, LC, and CV. In addition, the authors generated a new set of instances, labeled BF, composed of 640 instances of different size and complexity.[3]

Expósito-Izquierdo et al. (2012) propose a lowest priority first heuristic that iteratively places containers either at the bottom of a stack or above containers with lower priorities. Thus, the heuristic attempts to place containers in reverse order, starting with low priority containers first and eventually relocating containers with the highest priority. The proposed heuristic is stochastic in nature, since some of the decisions, e.g., the destination stacks of containers to be relocated, are randomly selected among a pool of candidate stacks. To assert the goodness of the proposed algorithm, the heuristic has been tested on instances CV and compared with the results from the literature as well as with an A* algorithm, implemented by the authors, which provided optimal values for the small-size instances. In addition, the authors also propose an *I*nstance *G*enerator, called IG in the sequel, which takes as input a set of parameters, e.g., the number of stacks and tiers of the bay, the set of container priorities, the bay occupancy rate, and a handful of parameters that affect the bay layout, and produces instances with varying difficulty levels. Finally, a computational study aimed at finding the correlation between instance difficulty and bay occupancy rate and container distribution was carried out.

Huang and Lin (2012) discuss two versions of the CPMP: Type A is the standard pre-marshalling, in which a final configuration with no mis-overlays must be reached; type B is a variant of the CPMP, in which a pre-specified bay configuration should be enforced. The authors propose two heuristics for the two variants of the problem. Both methods are labeling algorithms, in which stacks receive a label related to the condition of the stack itself (e.g., wrongly arranged, correctly arranged). The evaluation of the method has been conducted on two instances of the set LH for the type A version and on a randomly generated instance for the type B problem.

Caserta and Voß (2009b) present a metaheuristic algorithm for the pre-marshalling problem. The central idea of the approach relies on iteratively solving to optimality smaller portions of the original problem. The algorithm consists of four different phases, in which ideas from the corridor method, roulette-wheel selection, and local search techniques are intertwined to foster intensification around an incumbent solution. The algorithm is stochastic in nature and is based upon a set of greedy rules that bias the behavior of the scheme toward the selection of the most appealing moves.

Lee and Chao (2009) define a bi-objective problem: On the one hand, the authors attempt to create a reshuffled bay that requires the minimum amount of rehandlings during the loading phase; on the other hand, such desired configuration should be reached in the minimum amount of steps, i.e., the final configuration should be reached minimizing the total number of rehandling operations. The approach is hybrid in the sense that heuristic techniques, such as neighborhood search, and mathematical programming techniques, such as integer programming,

[3]See Table 16.3 for a description of this benchmark set.

are intertwined to deal with different subproblems. First, the neighborhood search heuristic is used to find a chain of movements to sort out the bay, in such a way that the number of further rehandling required during the loading phase is minimum. Next, a binary integer programming model is solved to reduce the number of movements required to reach that final configuration. A number of minor heuristic rules are used to foster the effectiveness of the proposed algorithm.

The first work on pre-marshalling was presented by Lee and Hsu (2007). They propose an integer programming model based upon a multi-commodity network flow formulation. The network accounts for two dimensions, time and space. Each level of the network describes a specific point in time and captures the state of the bay at that instant. Connections among different levels of the network account for moves of containers over time and space, i.e., edges within the network are used to model the movement of a container from one stack to another in a given time period. The basic mathematical model, along with some extensions, is presented in the paper. Finally, in order to reduce the number of variables and to make the model tractable, some simplifications are introduced. One drawback of the model concerns the need to pre-define a parameter T, i.e., the total number of time periods required to completely reshuffle the bay (which is unknown). The appropriate choice of the value of T has a strong bearing on the computational time required to solve the model. If T is chosen too large, then a very large number of variables is created and, therefore, the MIP solver might not be able to reach the optimal solution in a reasonable amount of computational time. On the other hand, if T is chosen too small, a feasible solution might not even exist. Some analysis about this trade off is presented by the authors.

16.3.2 Container Re-marshalling Problem

Typically, the CRMP refers to the problem of moving a set of containers to pre-specified bays within the same block. As indicated in Kang et al. (2006), the bays in which the target containers are located before re-marshalling are called *source* bays and the empty bays to which these containers should be moved are called *target* bays. Containers within a block are characterized by two types of information:

- a *group* or *category*, accounting for, e.g., the port of destination. In order to minimize the distance traveled by the cranes during the loading phase, containers belonging to the same group are placed in adjacent slots within the same block;
- a *priority*, accounting for, e.g., weight information, order of retrieval, etc. Within the same group, containers should be stacked by ensuring that no container with lower priority is found on top of a container with higher priority.

Therefore, the two-objective problem of intra-block re-marshalling is aimed at grouping together containers belonging to the same category and, for each set of containers of the same category, at piling up such containers taking into account priorities.

As pointed out in Caserta et al. (2011a), the CRMP should be seen as more than a simple extension of intra-bay pre-marshalling, since more than one crane could be used to handle the containers. Therefore, typically the re-marshalling problem also encompasses some considerations with respect to avoiding or minimizing interference among cranes within the same block. As mentioned in Sect. 16.2 of this paper, the authors proved that the CRMP is $\mathcal{NP}$-hard.

In the sequel, we present a brief summary of the contributions from the literature dealing with the CRMP. Table 16.4 provides a list of the papers hereby presented, along with the type of approach used, and a short comment on the paper itself.

Shin and Kim (2015) deal with the study of steal plate storage systems, in which a multi-state re-marshalling problem is addressed. It is common practice to divide the storage yard into zones, each dedicated to the storage of plates with remaining duration of stay within a specified range. Then, plates are assigned to zones depending on their remaining duration of stay. When a period of time passes, the durations of stay of the plates are updated and, consequently, it might be required that some plates are relocated from their current zone to the next zone in the yard. Thus, the re-marshalling is done periodically between zones with consecutive remaining duration of stay ranges. Via a formulation and some enumerative procedures, the proposed approach finds the optimal number of stacks and the optimal frequency of re-marshalling operations, i.e., the set of parameters that minimizes the expected number of re-marshalling operations.

Choe et al. (2015) propose a novel approach to the re-marshalling problem. Most of the works presented in the literature assume that enough time is given to carry out a complete re-marshalling. More recent contributions have introduced the notion of "selective re-marshalling," e.g., Park et al. (2013) and Park et al. (2010). However, the constant feature of all the approaches presented in the literature is that the re-marshalling work is carried out in batches. In other words, a starting time for the re-marshalling is given and, considering the selective re-marshalling, an ending time is also provided. Within this time horizon, the goal is to find the best possible (partial) re-marshalling plan. This paper proposes to intertwine the scheduling of the two cranes typically assigned to a block, used to perform ordinary duties, with some re-marshalling operations, whenever such cranes are idle. Consequently, given a time horizon, the goal is to mix together the scheduling of ordinary tasks at the block with a partial re-marshalling. The scheduling of ordinary tasks is still the priority and, for this reason, one of the objectives is to minimize the delay of these tasks. However, a new objective is also introduced, i.e., the minimization of the makespan of *all* the jobs, both the ordinary and those due to re-marshalling. The re-marshalling jobs to be included in the time horizon are selected using heuristics inspired in the selective re-marshalling of Park et al. (2013) and Park et al. (2010). The authors use a GA for the iterative rescheduling and run extensive simulations to assert the effectiveness of the proposed approach.

A variant of the re-marshalling problem is presented in Ji et al. (2015), where an algorithm for the relocation of containers to vessels, along with the crane scheduling, is presented. Loading sequence and rehandling strategies are integrated within the same optimization model, which leads to the identification of the optimal

Table 16.4 List of contributions from the literature for the re-marshalling problem

Paper	Approach	Method	Comment
Shin and Kim (2015)	Exact	Enumeration	Applied to steel plate storage systems in port terminals
Choe et al. (2015)	Metaheuristic	GA	Re-Marshalling carried out during regular operations exploiting crane idle times
Ji et al. (2015)	Heuristic	Math Model	Optimal loading sequence with minimization of rehandles
Ayachi et al. (2013)	Heuristic	Greedy Rules	Different container types
Park et al. (2013)	Metaheuristic	CCEA	Only a selected subset of containers is rearranged
Choe et al. (2011)	Metaheuristic	SA	Two-stage, with depth-limited A*
Park et al. (2010)	Metaheuristic	GA	Dynamic replanning
Park et al. (2009)	Metaheuristic	CCEA	Two-stage greedy approach
Kang et al. (2006)	Metaheuristic	SA	Two-stage, partial-order graph, neighborhood search
Kim and Bae (1998)	Heuristic	DP	Loading sequence not available

CCEA Cooperative co-evolutionary algorithm
GA Genetic Algorithm
SA Simulated Annealing
DP Dynamic Programming

loading sequence and the minimization of required rehandling. Three strategies are considered, i.e., the lowest stack strategy, the nearest stack strategy, and the optimization strategy. The latter is the most effective strategy in terms of reducing the number of rehandles.

Ayachi et al. (2013) present a heuristic method for the re-marshalling of both inbound and outbound containers under uncertainty. The uncertainty arises from the imperfect information related to arrival and departure times. The authors show how to deal with different container types. Their method finds an optimal storage plan with respect to container departure time and minimizes the required re-marshalling operations at their departure time.

Park et al. (2013) consider the selective re-marshalling presented in Park et al. (2010), i.e., they consider the case in which the time allocated to re-marshalling is limited and, therefore, only a subset of containers can be reshuffled. The authors propose a three-step cooperative co-evolutionary algorithm: Container selection, target location identification, and re-marshalling schedule. In addition, a cooperative parallel search is carried out to find good solutions to each of the subproblems. In a fashion similar to what is done in Park et al. (2010), the method is iteratively repeated to deal with the uncertainty and the estimation errors introduced by the real-time operation of cranes.

Choe et al. (2011) study the intra-block re-marshalling problem where more than one crane is used to handle containers. Therefore, interference among cranes is taken into account. The authors propose a two-phase algorithm: During the first phase the target slots to which handled containers should be moved are identified, and in the second phase an optimal schedule of the cranes to actually perform the relocation of containers is found. The proposed algorithm, based upon simulated annealing, is aimed at finding a rehandling-free configuration of the block that can be achieved in the minimum amount of time. Based upon a partial order graph that captures all the feasible moves leading from the current block configuration to a target configuration, at each step of the search phase the algorithm evaluates the goodness of a candidate solution configuration by heuristically creating a crane schedule and estimating the time needed to complete re-marshalling to reach that particular configuration.

Park et al. (2010) introduce a new feature into the re-marshalling problem. The authors point out that it is quite possible that not enough time is given to carry out a complete re-marshalling of a block. Consequently, a "selective" re-marshalling must be carried out, in which only a subset of the containers is actually sorted out. The authors propose a two-step iterative algorithm: In the first step, an appropriate subset of containers is selected using heuristic measures; the second step is then focused on building the re-marshalling schedule for the selected containers. In addition, since the uncertainty associated to the crane scheduling at the block might introduce estimation errors, the two-step approach is iteratively applied within the context of a GA that exploits the solutions obtained in the previous iterations.

Park et al. (2009) analyze the re-marshalling problem with respect to export containers. Typical dimensions of the considered problem are 41 bays per block, where each bay is made up by 10 stacks and 6 tiers. A block is managed through the use of two cranes, one for export containers and another for import containers. Due to the large size of the considered blocks, the authors identify two sources of inefficiencies in the handling of containers. The first one is related to the horizontal movement of the cranes used to load containers to the vessel. Typically, export containers are unloaded from trucks and, therefore, are piled up near the landside of the block. Therefore, during the loading operations, the crane operating on the waterside is forced to travel long distances toward the landside of the block to pick-up export containers, hence affecting the overall time of the loading operation. A second source of inefficiency can be ascribed to the stacking of high priority containers below low priority containers, forcing a rehandling of the uppermost containers. The authors present a two-stage heuristic algorithm. The first stage uses heuristic rules to identify where, i.e., in which stacks, containers must be relocated. In the second stage of the algorithm, a cooperative co-evolutionary algorithm is used to identify the precise slot within which containers should be relocated (stack and tier), along with the order of movements of the containers to be reshuffled. Two populations are created to identify the slots and to define the order of movements. Information is exchanged in the following way: Initially, a solution for the target slots identification is found; such solution is then fed as input to the subproblem dealing with the movements sequence. In turn, the movements sequence defined by this last subproblem is used to find a better set of target slots, and the cooperative approach is repeated in cycles.

Similarly, Kang et al. (2006) deal with export containers, and the objective is to find a rearrangement that avoids future rehandling during the loading operation. As in Choe et al. (2011), multiple cranes are used within a block and, therefore, interference among cranes is also minimized. The proposed approach is similar to the one of Choe et al. (2011), since a two-phase algorithm is designed. First, a set of target locations is defined. Next, a partial order graph is created, with the goal of finding a set of feasible moves leading from the source configuration to the target configuration. The partial order graph captures all the possible moves leading from source to target configuration. Next, simulated annealing is used to find a solution that aims to minimize the overall time required to carry on the re-marshalling operations. Finally, a heuristic is employed to find a crane's feasible schedule. An interesting point brought out by the authors is related to the notion of neighbor solutions. Given a partial order graph, a neighbor of such graph is obtained by appropriately modifying the current one via the application of swapping among containers stored on different stacks of the same bay.

In a seminal work, Kim and Bae (1998) deal with the problem of how to efficiently move a set of containers from source bays to target bays. Containers in the target bays should be accommodated according to a pre-specified layout, called target layout. The intra-block re-marshalling problem is decomposed into two subproblems: (1) the bay matching and move planning problem, in which each source bay within the block is matched with the target bay in the target layout. Decisions with respect to how many containers should be moved between any two bays are made in this stage. This part of the problem is solved using dynamic programming (to define the bay matching needs) and the transportation algorithm (to plan the movement of containers among bays and assignment to cranes). Whenever crane interference arises due to container movements, the bay matching is called again under additional constraints that prohibit the conflicting bay matching; (2) the movement sequencing problem, in which the actual movements required to reach the target layout are scheduled. The authors adopt a “macroscopic” perspective of the problem, i.e., only the number of containers per group type and bay are considered, whereas the actual positions and rehandling within a bay are neglected.

16.4 Relocation and Retrieval

In this section, we provide an overview on publications related to relocation and retrieval at container ports. Such kind of problems, such as the BRP, are closely related to the previously discussed pre- and re-marshalling problems. However, a major difference arises: Pre- and re-marshalling problems only consider rehandling operations, but no retrieval activities. That is, moving a container from a bay to a destination vessel is not feasible. On the other hand, in the BRP, retrieval operations are included. That is, retrieving and rehandling operations are carried out in parallel. Consequently, the number of containers in the bay decreases for the BRP, whereas the number of containers in the bay (block) remains constant for pre-marshalling (re-marshalling) problems. The term CRP is an alternative name for the class of BRP. In the literature, it is an often used convention to apply the term BRP for two-dimensional scenarios, i.e., if a single bay is considered. The CRP, however, is introduced as a more general concept, for the treatment of two- or three-dimensional instances, i.e., described by bays or blocks. As the majority of papers is still addressing the two-dimensional case, in the remainder of this paper, we use mainly the term BRP, implicitly addressing also the CRP.

In recent years, the research activity in the area of the BRP has increased a lot. A total of 46 publications since 2006 can be identified.[4] In the sequel, after discussing the problem properties in Sect. 16.4.1, we focus on solution approaches in Sect. 16.4.2 and on problem extensions in Sect. 16.4.3.

16.4.1 Properties

Since the introduction of the BRP, several variations and extension of the BRP have emerged in the literature. However, there is a set of basic properties that hold for all BRP variants, which are presented next.

[4]See Table 16.1 from which 17 references address extensions of the BRP.

- Containers are piled up vertically in stacks, i.e., only the uppermost container of each stack is accessible for rehandling or retrieving; in addition, each container is either placed on the ground or on top of another container.
- The number of stacks describes the width of the bay.
- The height of stacks is bounded by the number of tiers.
- The number of bays defines the depth of the block (3D-case, only).
- The total initial number of containers in the bay is denoted by N.
- The initial configuration of the bay/block is given in advance.
- Each container in the bay is associated with a priority number, where more than one container can belong to the same priority group (indicated by the priority number).
- Containers have to be retrieved from the bay according to their priority number, i.e., a container with a certain priority can only be retrieved if all containers with higher priorities have already been removed.
- Containers to be removed next are called *target containers*. Rehandling operations become necessary, if no target container is accessible.
- A majority of models given in the literature add the following condition: (A1) Only containers located in the same stack as and above the current target container are allowed to be rehandled (see, e.g., Kim and Hong 2006). This stack is called *target stack*. Following the notation provided in the literature, see, e.g., Zhu et al. (2012), we call a BRP under A1 as *restricted* and a BRP neglecting A1 as *unrestricted*.

Moreover, there is a set of properties that are valid for the BRP. However, when some of these properties are relaxed or modified, extensions of the BRP are obtained. See Sect. 16.4.3 for an introduction to extended versions of the BRP.

- The objective of the BRP is to retrieve all the containers from the bay in the prescribed order while minimizing the number of rehandling operations.
- The retrieval sequence, indicated by the priority numbers of the containers, is given in advance.
- There are no containers entering the bay/block.

16.4.2 Solution Methods

In this section, we provide an overview on available solution approaches in the field of the BRP. As already detailed in Sect. 16.3 in relation to the container marshalling problems, solution methods for the BRP stem from the fields of exact approaches, metaheuristics, and greedy, target-guided heuristics. Tables 16.5 and 16.6 provide an overview on available references in this area sorted by the year of publication. For each publication, the chosen method and benchmark set as well as the BRP version are reported. An overview on benchmark instances for the BRP is given in Table 16.7. To survey the available literature in more detail, we first provide in this section an overview on exact methods and distinguish within this context work

Table 16.5 List of papers on the BRP and CRP, from 2006 to 2016

Paper	Approach	Method	Benchmark	BRP version
Galle et al. (2016)	Heuristic	Average case analysis	–	Restricted
Ku and Arthanari (2016b)	Exact	Abstraction/tree	LL	Restricted
Ku and Arthanari (2016a)	Exact/Heuristic	Stochastic opt./abstraction/tree	KA	Extension (CRPTW)
Tanaka and Takii (2016)	Exact	B&B[c]	ZQLZ	Restricted
Tricoire et al. (2016)	Exact/Metaheuristic	B&B[c]/rake search	CVS	Restricted/unrestricted
Zhang et al. (2016)	Heuristic	Tree	ZLK/CSV	Extension (BRP-BM)
Zehendner et al. (2017)	Heuristic	Target-guided/Online	CVS	Extension (OCRP)
Borjian et al. (2015)	Heuristic	Average case analysis	–	Restricted
Eskandari and Azari (2015)	Exact	MILP[a]	CVS	Restricted
Expósito-Izquierdo et al. (2015b)	Exact	B&B[c]	CVS	Restricted
Expósito-Izquierdo et al. (2015a)	Heuristic	Target-guided	ELAMM	Extension (SP)
Jin et al. (2015)	Metaheuristic	Look-ahead/tree	CVS/BF/ZQLZ	Restricted
Lin et al. (2015)	Heuristic	Target-guided	LL	Extension (crane time)
Schwarze and Voß (2015)	Exact	MILP[a]	CVS	Extension (crane time)
Tanaka and Mizuno (2015)	Exact	B&B[c]	ZQLZ	Unrestricted
Tang et al. (2015)	Exact/Heuristic	Target-guided/simulation	TJLD	Restricted/extension (DCRP)
Zehendner et al. (2015)	Exact	MILP[a]	CVS	Restricted
Akyüz and Lee (2014)	Heuristic	MILP[a]/target-guided/B S[b]	AL	Extension (DCRP)
Expósito-Izquierdo et al. (2014)	Exact/Heuristic	A*/knowledge-based	CVS	Restricted/unrestricted
Jovanovic and Voß (2014)	Metaheuristic	Chain heuristic	WT	Restricted
Olsen and Gross (2014)	Heuristic	Average case analysis	–	Restricted
Tanaka and Takii (2014)	Exact	B&B[c]	ZQLZ	Restricted
Zehendner and Feillet (2014)	Exact	BVB[c]	CVS	Restricted
Lehnfeld and Knust (2014)	Survey	–	–	–

[a]Mixed-integer linear programming
[b]Beam search
[c]Branch-and-bound

Table 16.6 List of papers on the BRP and CRP, from 2006 to 2016, cont.

Paper	Approach	Method	Benchmark	BRP version
Borjian et al. (2013)	Heuristic	Stochastic opt.	BMBJ	Extension (DCRP)
Hussein and Petering (2013)	Heuristic	Target-guided/GA[a]	HP13	Extension (BRP-W)
Jin et al. (2011)	Metaheuristic	Look-ahead/tree	CVS/BF	Restricted
Petering and Hussein (2013)	Exact/Metaheuristic	MILP[b]/Look-ahead	CSV/LL	Unrestricted
Caserta et al. (2012)	Exact/Heuristic	MILP[b]/priority	CSV	Restricted
Forster and Bortfeldt (2012b)	Heuristic	Tree	CVS/BF	Restricted
Forster and Bortfeldt (2012a)	Heuristic	Tree	LL	Extension (crane time)
Hussein and Petering (2012)	Heuristic	Target-guided/GA[a]	HP12	Extension (BRP-W)
Rei and Pedroso (2012b)	Heuristic	Tree/multiple-simul.	RP	Extension (SP)
Rei and Pedroso (2012a)	Heuristic	MILP[b]	RP2	Extension (SP)
Ünlüyurt and Aydin (2012)	Exact	B&B[e]/target-guided	ÜA	Extension (crane time)
Zhu et al. (2012)	Exact	A*/IDA*[d]	ZQLZ/CVS/LL	Restricted/unrestricted
Caserta et al. (2011b)	Exact	Dynamic progr.	CVS	Restricted
Caserta et al. (2011a)	Survey	–	–	–
Lee and Lee (2010)	Heuristic	MILP[b]	LL	Extension (crane time)
Wu and Ting (2010)	Metaheuristic	BS[c]	WT	Restricted
Wu et al. (2010)	Metaheuristic	Tabu search	WHT	Restricted
Zhang et al. (2010)	Metaheuristic	IDA*[d]	ZGZLC	Restricted
Zhu et al. (2010)	Heuristic	Filtered BS[c]	ZF	Extension (crane time)
Caserta et al. (2009)	Metaheuristic	Look-ahead	CVS	Restricted
Caserta and Voß (2009a)	Metaheuristic	Corridor method	CVS	Restricted
Caserta and Voß (2009c)	Metaheuristic	Corridor method	CVS	Restricted
Wan et al. (2009)	Exact	MILP[b]/target-guided	WLT	Restricted/extension (DCRP)
Kim and Hong (2006)	Exact/Heuristic	B&B[e]/target-guided	KH	Restricted

[a]Genetic algorithm
[b]Mixed-integer linear programming
[c]Beam search
[d]Iterative deepening A*
[e]Branch-and-bound

Table 16.7 Benchmark instances for the BRP

Name	Source	Inst	groups	Inst/group	B	W	H
KH	Kim and Hong (2006)	13	13	1	–	$3,\dots,8$	$3,\dots,5$
WLT	Wan et al. (2009)	600	12	50	–	6	2,3,4,5
LL[a]	Lee and Lee (2010)	70	10	5	1,2,4,6,8	16	6,8
WT	Wu and Ting (2010)	1920	48	40	–	$3,\dots,10$	$3,\dots,8$
ZGZLC	Zhang et al. (2010)	12500	125	100	–	$6,\dots,10$	$3,\dots,7$
WHT	Wu et al. (2010)	600	60	10	–	$3,\dots,12$	$3,\dots,8$
ZF	Zhu et al. (2010)	10	10	1	–	$3,\dots,10$	$3,\dots,10$
CSV[b]	Caserta et al. (2012)	840	21	40	–	$3,\dots,10$	3,4,5,6,10
BF	Bortfeldt and Forster (2012)	640	32	20	–	16,20	5,8
ZQLZ	Zhu et al. (2012)	12500	125	100	–	–	$3,\dots,7$
ÜA	Ünlüyurt and Aydin (2012)	8000	200	40	–	$3,\dots,7$	$4,\dots,7$
HP12	Hussein and Petering (2012)	1200	12	100	–	3,6,10,14	3,5,7
RP	Rei and Pedroso (2012b)	24	6	4	–	2,3,4,10,20,40	–
RP2	Rei and Pedroso (2012a)	12	6	2	–	2,3,4,10,20,40	–
HP13	Hussein and Petering (2013)	1200	12	100	–	3,6,10,14	3,5,7
BMBJ	Borjian et al. (2013)	150	2	120,30	–	3,4	3,4
AL	Akyüz and Lee (2014)	2400	4	600	–	6	$2,\dots,6$
TJLD	Tang et al. (2015)	400	8	50	–	6	$2,\dots,5$
ELAMM	Expósito-Izquierdo et al. (2015a)	420	42	10	–	4,5,6,7,8,9,10	5,6,7,8,9,10
ZLK	Zhang et al. (2016)	15	15	5	–	$5,\dots,10$	$4,\dots,10$
KA[c]	Ku and Arthanari (2016a)	720	24	30	–	$5,\dots,10$	$3,\dots,6$

[a]https://sites.google.com/site/smallcontainerworld/
[b]https://www.bwl.uni-hamburg.de/iwi/forschung/projekte/dataprojekte/brp-instances-caserta-etal-2012.zip
[c]http://crp-timewindow.blogspot.com

regarding the restricted and the unrestricted BRP. Later, we investigate heuristic methods and separate this area accordingly into material on the restricted and on the unrestricted BRP. Afterwards, in Sect. 16.4.3, extensions of the BRP are discussed together with a description of the corresponding solution approaches.

Several exact methods for the BRP are available in the areas of mathematical modeling: Branch-and-bound, tree search, A*-algorithms, and dynamic programming. A subset of articles focus only on exact methods, whereas other references introduce exact methods but add heuristics for addressing medium and larger instance sizes, see column "Approach" in Tables 16.5 and 16.6. In the sequel, the respective work is clustered and discussed according to the chosen approaches and BRP version. First, exact approaches are given for the restricted and afterward for the unrestricted BRP.

For the unrestricted BRP (without assumption A1), Caserta et al. (2012) provide a first mathematical formulation (BRP-I). Later, Petering and Hussein (2013) introduce BRP-III, an alternative mathematical model for the unrestricted BRP that requires a reduced number of decision variables. The improvement of running times when using the BRP-III is illustrated in experiments. Moreover, Expósito-Izquierdo et al. (2014) provide an A* algorithm that can be adapted to both, the restricted as well as the unrestricted BRP. Similarly, the Iterative Deepening A* algorithm presented by Zhu et al. (2012) can be applied to both versions of the BRP. Moreover, Tricoire et al. (2016) introduce a branch-and-bound approach and compare it against the A* algorithm of Expósito-Izquierdo et al. (2014). Their results indicate that their branch-and-bound approach with depth-first policy outperforms the A* algorithm concerning the number of solved instances in a given time frame. Finally, Tanaka and Mizuno (2015) develop dominance criteria for excluding a subset of feasible solutions from the search space. They apply this approach within a branch-and-bound method.

A first mathematical model for the restricted BRP is proposed by Wan et al. (2009) and used in the extended context of locating ingoing containers, see Sect. 16.4.3. Furthermore, the mathematical model BRP-I serves as basis for the BRP-II, a mixed-integer linear program modeling the restricted BRP (Caserta et al. 2012). A corrected and improved version of the BRP-II is provided by Zehendner et al. (2015) together with a pre-processing procedure and a new upper bound that is implemented as cut in the model. An alternative correction of the BRP-II is proposed by Eskandari and Azari (2015). Experiments illustrate that the improved BRP-II-A performs better than the corrected BRP formulation regarding computational time and number of solved instances. Moreover, branch-and-bound approaches are suggested by Kim and Hong (2006), Ünlüyurt and Aydin (2012), Expósito-Izquierdo et al. (2015b), and Tanaka and Takii (2016) (see Tanaka and Takii 2014 for an earlier version of this article). As stated above, Expósito-Izquierdo et al. (2014) and Zhu et al. (2012) provide A* and Iterative Deepening A* algorithms for the restricted BRP. Finally, a dynamic programming method is introduced by Caserta et al. (2011b) and a branch-and-price method is presented by Zehendner and Feillet (2014). Recently, Ku and Arthanari (2016b) proposed an abstraction method for the restricted BRP which allows to reduce the search space and that is applied within a tree search.

The $\mathcal{NP}$-hardness of the (restricted and unrestricted) BRP justifies the usage of heuristic approaches that in particular become relevant when addressing realistic problem instances of larger sizes. For the unrestricted case, Petering and Hussein (2013) present a look-ahead algorithm which extends a similar approach given by Caserta et al. (2009) for the restricted case. In this approach, Petering and Hussein (2013) include voluntary moves into the set of activities. That is, the rearrangement of a block that is not located in the target stack is feasible. Tricoire et al. (2016) joins voluntary as well as forced relocation options under a modified approach, introducing a set of policies for choosing moves. These policies are embedded within *rake search*, a metaheuristic framework based on tree search. Furthermore, Expósito-Izquierdo et al. (2014) present a domain-specific knowledge-based heuristic which consists of a set of basic rules and a heuristic evaluation for guiding the search strategy.

For the restricted BRP, Kim and Hong (2006) propose a first heuristic method that chooses the next move based on the *E*xpected *N*umber of *A*dditional *R*elocations (ENAR) in the resulting bay layout. The heuristic is experimentally compared with the exact branch-and-bound approach proposed in the same paper, indicating an average increase of moves by up to 7.3%. Furthermore, a simple heuristic priority rule is proposed by Caserta et al. (2012) and measured against the exact solution and the heuristic solution of Kim and Hong (2006). Olsen and Gross (2014) investigate a priority heuristic similar to that one provided by Caserta et al. (2012) and add a discussion of its performance. More detailed, an average case analysis based on assumptions on initial stack height and stack capacity is given. Along the same line, Galle et al. (2016) study the performance of the heuristic given by Caserta et al. (2012) for the case of asymptotically growing number of stacks. For this case the convergence of the expected number of relocations to a lower bound is proved. Moreover, Borjian et al. (2015) carry out similar considerations for the A* algorithm. A metaheuristic approach for the restricted BRP is presented by Caserta and Voß (2009b). In this work, the corridor method is adapted to the BRP, where the corridor limits the number of potential stacks for relocation. The presented approach embeds a dynamic programming scheme and applies it by iteratively solving to optimality "constrained" versions of the original BRP. Metaheuristic approaches adapted and applied to the restricted BRP are presented by Caserta and Voß (2009c) and Caserta and Voß (2009a). In these approaches, parameterization and tuning methods for the corridor method are proposed, where the corridor limits the number of potential stacks for relocation.

Caserta et al. (2009) describe an alternative encoding of the bay using a binary matrix, which enables fast access to layout information and fast bay transformation. This encoding is applied for the implementation of a random-guided look-ahead procedure that explores the quality of potential moves by evaluating their potential future performance. Look-ahead policies are later also considered by Jin et al. (2011) and Jin et al. (2015). In these approaches, a tree search is performed including inspection by look-ahead procedures combined with a locally applied probing heuristic. A particular version of look-ahead is performed within the chain heuristic, proposed by Jovanovic and Voß (2014), where information about the container to be moved next is included in a current decision. The evaluation of simple move strategies within a tree search based evaluation is found in further approaches. For instance, Wu and Ting (2010) design a beam search algorithm that inspects only a subset of the search tree. Moreover, Forster and Bortfeldt (2012b) develop a tree search approach including lower bounds on the number of relocations. Related to tree search approaches is the class of A* approaches which can be performed as heuristics by reducing the search space. Zhang et al. (2010) and Zhu et al. (2012) propose an *I*terative *D*eepening *A** (IDA*) algorithm that includes lower bounds and a heuristic probing approach to evaluate and prune nodes during the search. Finally, a tabu search approach is implemented by Wu et al. (2010) and compared based on a simple branch-and-bound presented in the same paper.

16.4.3 Problem Extensions

Since the introduction of the BRP, several extensions have emerged in the literature. By relaxing particular properties of the original BRP, e.g., the property that no ingoing containers are allowed, or, by adding additional parameters, like the weight of containers, new versions of the BRP arise. In the sequel, we present an overview on recent models and solution approaches.

Already mentioned above is the extension from a two-dimensional to a three-dimensional stacking area. This extension in dimension directly elevates the relevance of crane activities for modeling approaches as the time consumption for a crane movement across a bay is usually different from the time required for crane movements within a bay such that a more detailed consideration of crane working times might be of interest for a realistic model. The consideration of crane working times naturally leads to the definition of alternative objective functions. The standard objective function for the BRP, as introduced by Kim and Hong (2006) is to minimize the number of relocations. As an alternative approach, objectives can be designed based on the crane working time including time consumption for picking-up/placing-down containers, for moving trolleys across the stacks and for moving gantries across bays. A basic model for the crane time supposes that the time for picking-up/placing-down is constant, i.e., independent of the number of tiers that are crossed. A more detailed approach includes tier-dependent pick-up/place-down effort in an extended crane time model. Lee and Lee (2010) develop a three-phase heuristic to minimize the sum of relocations and basic crane time in a three-dimensional setting. Also with respect to a three-dimensional yard, Forster and Bortfeldt (2012a) introduce a tree search heuristic to minimize the basic crane time. For a two-dimensional bay, i.e., neglecting gantry operations, Ünlüyurt and Aydin (2012) propose a branch-and-bound approach as well as a heuristic. Finally, Zhu et al. (2010) include a consideration of spreader and trolley movements and thus address extended crane times. A filtered-beam search approach is suggested. Moreover, Lin et al. (2015) considers extended crane times including a tier-dependent effort for picking-up/placing-down and includes those measures into a heuristic that is, however, focusing on minimization of the number of relocations. Finally, Schwarze and Voß (2015) investigates the relation between different objectives by analyzing to which extent optimal solutions are changed when the objective function is replaced.

The consideration of fuel consumption is included into the *BRP* with *W*eights (BRP-W), see, Hussein and Petering (2012). In that setting, the weight of each container is known and impacts the energy consumption for container movement. Consequently, the BRP-W aims at minimizing the total energy required for removing all containers from the stacking area. Hussein and Petering (2012) propose a *G*lobal *R*etrieval *H*euristic (GRH) that relies on a set of parameters describing preferences for container movement. Using these parameters, a penalty score is computed for each stack. The GRH is embedded in a genetic algorithm that searches for a good configuration of the parameters. The results are extended by Hussein and Petering (2013) by modifying the penalty function.

One assumption of the BRP is that there are only retrieval or relocation activities; however, storing new items into the bay is not feasible. Nevertheless, such operations are often required in practice, such that it is a natural extension to allow incoming items. The DCRP is a *dynamical* variant of the BRP that joins relocation, retrieving and stacking of incoming items. For this problem class, Wan et al. (2009) propose heuristic approaches including basic priority rules for choosing stacks as new location for incoming or relocated containers. In a second approach the expected number of additional relocations is considered, inspired by the heuristic of Kim and Hong (2006). Moreover, a further heuristic is proposed that includes the solution of a series of CRP formulations. Later, Tang et al. (2015) propose rule-based heuristics for the DCRP and evaluates them through a simulation approach. Furthermore, Borjian et al. (2013) add uncertainty to the DCRP by assuming incomplete information and solving the resulting problem using a two-stage stochastic optimization model. A mathematical formulation for the (deterministic) DCRP is provided by Akyüz and Lee (2014). Furthermore, in this work, three heuristics are developed for the DCRP. First, index-based heuristics add weights to the columns in order to position incoming and relocated containers. A second heuristic applies the mathematical model for the DCRP on small portions of the planning horizon. Finally, a beam search heuristic is proposed including upper and lower bound approaches allowing to reduce the size of the search tree. Similar to the DCRP is the *S*tacking *P*roblem (SP), which was formulated for an application in the steel industry by Rei and Pedroso (2012b) and Rei and Pedroso (2012a). A simulation approach is proposed that combines simulation with a construction heuristic. Furthermore, a probabilistic tree search method is developed. A two-phase heuristic for the SP is proposed by Expósito-Izquierdo et al. (2015a). Their approach joins two basic steps, namely the selection of target stacks for relocated or ingoing containers and, second, the exploitation of idle crane time to resolve conflicts and improve the crane productivity.

A further extension of the BRP that includes time windows is described by Ku and Arthanari (2016a). In their approach, the case of import containers is considered. Import containers are stored at the yard until their pick-up for the hinterland transport. In such scenarios, *T*ruck *A*ppointment *S*ystems (TAS) handle the visit of trucks from the hinterland and monitor announced time slots of arrival. Within this pre-defined time slot, the actual arrival time of the truck is unknown. The *CRP* with *T*ime *W*indows (CRPTW) includes this uncertainty by allowing stochastic retrieval sequences within time windows under the objective of finding a minimum expected number of rehandles. To that end, a stochastic dynamic programming model is developed and an exact tree search method with depth-first search is proposed together with an abstraction heuristic that allows to reduce the search space. Furthermore, an index-based heuristic is proposed that evaluates the expected number of containers in a column that depart earlier than a container that is potentially relocated to this column.

An alternative approach that considers incomplete information regarding the retrieval sequence is proposed by Zehendner et al. (2017). In line with the above described CRPTW, the *O*nline *CRP* (OCRP) relaxes the assumption of known priority numbers. However, while the CRPTW includes stochastic retrieval sequences for time windows, the OCRP assumes the retrieval sequence to be revealed in an online fashion over time. Consequently, online optimization techniques are applied to evaluate the success of the proposed target-guided leveling heuristic. More specifically, the competitiveness ratio of the leveling heuristic is determined and average and worst-case analysis are carried out. Finally, a recent extension of the BRP addresses new crane technology that enables lifting of more than one item and that could become an option, e.g., in steel plants. The *BRP* with *B*atch *M*oves (BRP-BM) is introduced by Zhang et al. (2016). This problem formulation addresses new features of crane technology that allow to lift more than one item at the same time. Such moves are called batch moves. Zhang et al. (2016) propose a greedy heuristic for the BRP-BM. Furthermore, lower bounds on the number of relocations are proposed and applied within tree search methods.

16.5 Related Research Fields

In this section, we give a brief outlook on work in related research fields and the relationship to the above discussed post-stacking problems in order to refer the interested reader to related notions and concepts. However, we do not aim at giving a comprehensive overview over work in those fields beyond maritime shipping.

In the previous sections, we have studied pre-, re-marshalling, and retrieval in container yards. As pointed out by Steenken et al. (2004), the selection of storage locations for incoming containers is an additional main task in container reshuffling and thus related to the aforementioned problems. This relation is already addressed in the DCRP, see Sect. 16.4.3, by proposing a joint handling of incoming items and reshuffling operations. Moreover, see Bruns et al. (2016) for a recent study on complexity issues for storage loading problems. In a broader context, the task of locating incoming containers includes moreover the assignment of storage space to containers or container groups, see, e.g., Chen and Lu (2012); Woo and Kim (2011) as well as the selection of storage allocations, i.e., the question of how containers should be piled up in the stacking area, see, e.g., Borgman et al. (2010); Dekker et al. (2006); Jang et al. (2013). Moreover, see Carlo et al. (2014) for more detailed classification and literature on storage space assignment.

In the event of available crane time, pre- and re-marshalling, see Sect. 16.3, can be carried out to resolve conflicts within a bay or block before the retrieval process starts and in order to speed up the subsequent stowing operations. This idea of saving berthing time by carrying out operations in advance, before the arrival of the vessel, is transferred to the complete yard area through the approach of transporting containers to positions closer to the scheduled berth. This process is known as *housekeeping*, see, e.g., Legato et al. (2013), Ehleiter and Jaehn (2016), and Cordeau et al. (2015) for more details in this field.

Moreover, stacking, sorting, and rehandling problems are discussed not only in the context of containers and ports, but also in different areas like warehousing, production planning, and artificial intelligence. Some warehouses are organized following the stacking principle, by storing uniform items piled up on top of each other, where access is only granted for the uppermost item. Stacking operations in those warehouses follow similar rules as in container yards. However, a major difference between container yards and warehouses is given by the item flow, as warehouses have to offer retrieving and receiving operations in parallel (see, e.g., Nishi and Konishi (2010)), whereas in container yards, the receiving operations are usually completed before the retrieval operations take place. Moreover, in general, warehouses handle a much larger number of items than container yards. In addition, the physical properties of the items in a warehouse might differ from that of a box-shaped container. For instance, in the steel industry, coils are stored by stacking them on top of each other. The resulting storage setting is not forming “stand-alone” stacks, as each coil is placed on top of two consecutive coils from the row below (see, e.g., Zäpfel and Wasner 2006). See Tang and Ren (2010) and Tang et al. (2012) for approaches that include crane times into problems from stacking in the steel industry.

Also the handling of trains involves stacking operations; see, e.g., Felsner and Pergel (2008). A train can be seen as a sequence of wagons. It might happen that the wagon sequence of a single train needs to be changed or that the wagons of several trains have to be *reshuffled* to new collections of trains. These operations are physically carried out on dead end sidings, where trains or parts of trains can be stored intermediately and taken away later on. Thus, on dead end sidings, trains can be “stacked” together and moreover, rehandling of wagons is possible. Each of those dead end sidings relates to a stack in the container yard, where only the uppermost container/wagon is accessible.

A well-known concept in artificial intelligence is that of blocks-world. (See, e.g., Romero and Alquézar 2004, Gupta and Nau 1992.) The blocks-world is carried out on a “table” where blocks are stacked on top of each other. A typical blocks-world instance consists of a given initial table state and a desired goal state. The task is to transform the initial state to the goal state with a minimum number of moves. Variants of blocks-world incorporate limitations on the table size and different levels of given conditions for the goal state. Gupta and Nau (1992) prove the $\mathcal{NP}$-hardness of blocks-world and Caserta et al. (2012) show that the BRP is a particular case of blocks-world.

16.6 Conclusion and Future Challenges

Ever since the first containers were introduced in the early 1960s, container handling techniques and strategies have always been key factors in measuring the efficiency of major ports. However, due to the growth of container vessels in recent years, whenever one of such ships berths at a port, a number of containers that would have been unthinkable some time ago must be handled in just a few hours. This poses a serious challenge for container terminal operators, since the volume of traffic has grown substantially while the available surface for managing such traffic has remained virtually unchanged. Therefore, optimization techniques for handling and rehandling containers acquire a prominent role in fostering efficiency of container terminal operations.

Moreover, in the stages of design, construction, and operation of a container terminal, simulation tools have turned out to play a crucial role, examples are given in, e.g., Gambardella et al. (1998) and Yun and Choi (1999). Questions of interest are, among others, the layout of the terminal itself, including location and size of facilities (container yards, mooring, maintenance areas, etc.), design and operation of transport systems (AGVs, cranes, etc.), and modeling of container flows. Optimization methods, like those addressing rehandling and stacking operations at ports, are suited to extend and enhance classical simulation approaches. For instance, integrated simulation-optimization establishes a simulation tool on a superior level which has the permission to call optimization methods on a sublevel. In such a setting, the optimization algorithm can, e.g., take over a tactical position and be used to define and control general system parameters on an aggregate level (Saccone and Siri (2009)). In an alternative setting, optimization tools could be used to take decisions on a detailed, operational level. For instance, while analyzing transport systems at a container terminal using simulation, it is helpful to call optimization tools that solve particular rehandling and stacking problems to obtain information on capacity utilization of cranes and vehicles. Along the same line, while designing a terminal layout through simulation, analysis of detailed stacking operations at container yards is relevant to determine required storage and handling capacities. The availability of fast optimization techniques is a crucial issue of integrated simulation-optimization tools as typically, optimization methods will be called quite often. Thus, the development of efficient optimization techniques is an important matter of terminal planning.

In this chapter, we have presented an updated survey on techniques for post-stacking situations, based on an earlier version (Caserta et al. 2011a). We have focused on three classes of post-stacking problems, namely the re-marshalling, the pre-marshalling, and the relocation problem and provided a comprehensive overview on exact and (meta-)heuristic methods in this areas. This includes a summary of available benchmark instances and a description of problem extensions. Moreover, work in related fields has been discussed.

In Caserta et al. (2011a), we mentioned the design of efficient algorithms for online optimization, the use of recent findings in the metaheuristic field, and the development of broader, integrated approaches for container terminal logistics as open challenges for the aforementioned problems. It can be observed that since then, a number of publications have addressed problems from the mentioned areas. Examples of this research stream can be found, e.g., in the field of BRP extensions. Integrated approaches for related operational tasks, like handling of incoming items and the subsequent restacking, are proposed and combinations with online optimization policies are suggested there. Moreover, Tables 16.5 and 16.6 illustrate that the variety of existing metaheuristic approaches for post-stacking problems is rich. Although this activity illustrates that the interest in this area is high and that quite a bit of work has been done to answer research questions, still the mentioned challenges remain as open working areas and offer opportunities for new research. For instance, in the broader context of housekeeping in container yards, it will be worthwhile to transfer available methods and knowledge from the field of post-stacking to related problem areas and focus on integrated solution approaches.

A further avenue for advancing the work in this field is identified in Caserta et al. (2011a) as the exploitation of recent methods in computer technology, like parallel computing and grid computing. The increase in computational power obtained by using such techniques will not only allow to address larger problem sizes but could also enable to handle problem types of a higher integration level, that are harder tractable from a computational perspective.

References

Akyüz M, Lee CY (2014) A mathematical formulation and efficient heuristics for the dynamic container relocation problem. Nav Res Logist 61(2):101–118

Ayachi I, Kammarti R, Ksouri M, Borne P (2013) A heuristic for re-marshalling unbound and outbound containers. In: 2013 2nd International Conference on Systems and Computer Science (ICSCS), pp 291–296

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

Borjian S, Manshadi V, Barnhart C, Jaillet P (2013) Dynamic stochastic optimization of relocations in container terminals. Technical report, MIT working paper

Borjian S, Galle V, Manshadi VH, Barnhart C, Jaillet P (2015) Container relocation problem: approximation, asymptotic, and incomplete information. arXiv:1505.04229. https://arxiv.org/abs/1505.04229

Bortfeldt A, Forster F (2012) A tree search procedure for the container pre-marshalling problem. Eur J Oper Res 217(3):531–540

Bruns F, Knust S, Shakhlevich NV (2016) Complexity results for storage loading problems with stacking constraints. Eur J Oper Res 249:1074–1081

Carlo H, Vis I, Roodbergen K (2014) Storage yard operations in container terminals: literature overview, trends, and research directions. Eur J Oper Res 235(2):412–430

Caserta M, Voß S (2009a) A cooperative strategy for guiding the corridor method. In: Nature inspired cooperative strategies for optimization (NICSO 2008). Springer, Berlin, pp 273–286

Caserta M, Voß S (2009b) A corridor method-based algorithm for the pre-marshalling problem. Lect Notes Comput Sci 5484:788–797

Caserta M, Voß S (2009c) Corridor selection and fine tuning for the corridor method. Lect Notes Comput Sci 5851:163–175

Caserta M, Schwarze S, Voß S (2009) A new binary description of the blocks relocation problem and benefits in a look ahead heuristic. Lect Notes Comput Sci 5482:37–48

Caserta M, Schwarze S, Voß S (2011a) Container rehandling at maritime container terminals. In: Böse JW (ed) Handbook of terminal planning. OR/CS interfaces series, vol 49. Springer, New York, pp 247–269

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

Caserta M, Schwarze S, Voß S (2012) A mathematical formulation and complexity considerations for the blocks relocation problem. Eur J Operat Res 219(1):96–104

Chen L, Lu Z (2012) The storage location assignment problem for outbound containers in a maritime terminal. Int J Prod Econ 135:73–80

Choe R, Park T, Oh M, Kang J, Ryu K (2011) Generating a rehandling-free intra-block remarshaling plan for an automated container yard. J Intel Manuf 22(2):201–217

Choe R, Kim T, Kim T, Ryu K (2015) Crane scheduling for opportunistic remarshaling of containers in an automated stacking yard. Flex Serv Manuf J 27(2):331–349

Cordeau JF, Legato P, Mazza RM, Trunfio R (2015) Simulation-based optimization for housekeeping in a container transshipment terminal. Comput Oper Res 53:81–95

Dayama N, 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 J Trans Log 6:1–42

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

Ehleiter A, Jaehn F (2016) Housekeeping: foresightful container repositioning. Int J Prod Econ 179:203–211

Eskandari H, Azari E (2015) Notes on mathematical formulation and complexity considerations for blocks relocation problem. Sci Iran Trans E Indu Eng 22(6):2722–2728

Expósito-Izquierdo C, Melián-Batista B, Moreno-Vega M (2012) Pre-marshalling problem: heuristic solution method and instances generator. Expert Syst Appl 39(9):8337–8349

Expósito-Izquierdo C, Melián-Batista B, Moreno-Vega JM (2014) A domain-specific knowledge-based heuristic for the blocks relocation problem. Adv Eng Inform 28(4):327–343

Expósito-Izquierdo C, Lalla-Ruiz E, de Armas J, Melián-Batista B, Moreno-Vega JM (2015a) A heuristic algorithm based on an improvement strategy to exploit idle time periods for the stacking problem. Comput Ind Eng 87:410–424

Expósito-Izquierdo C, Melián-Batista B, Moreno-Vega JM (2015b) An exact approach for the blocks relocation problem. Expert Syst Appl 42:6408–6422

Felsner S, Pergel M (2008) The complexity of sorting with networks of stacks and queues. Lect Notes Comput Sci 5193:417–429

Forster F, Bortfeldt A (2012a) A tree search heuristic for the container retrieval problem. In: Operations research proceedings 2011, pp 257–262

Forster F, Bortfeldt A (2012b) A tree search procedure for the container relocation problem. Comput Oper Res 39:299–309

Galle V, Borjian Boroujeni S, Manshadi VH, Barnhart C, Jaillet P (2016) An average-case asymptotic analysis of container relocation problem. Oper Res Lett 44:723–728

Gambardella L, Rizzoli A, Zaffalon M (1998) Simulation and planning of an intermodal container terminal. Simulation 71(2):107–116

Gheith M, Eltawil A, Harraz N (2014) A rule-based heuristic procedure for the container pre-marshalling problem. In: 2014 IEEE International Conference on Industrial Engineering and Engineering Management, pp 662–666

Gheith M, Eltawil A, Harraz N (2016) Solving the container pre-marshalling problem using variable length genetic algorithms. Eng Optim 48(4):687–705

Gupta N, Nau D (1992) On the complexity of blocks-world planning. Artif Int 56(2–3):223–254

Hottung A, Tierney K (2016) A biased random-key genetic algorithm for the container pre-marshalling problem. Comput Oper Res 75:83–102

Huang S, Lin T (2012) Heuristic algorithms for container pre-marshalling problems. Comput Ind Eng 62(1):13–20

Hussein MI, Petering MEH (2012) Global retrieval heuristic and genetic algorithm in block relocation problem. In: Lim G, Herrmann JW (eds) Proceedings of the 2012 industrial and systems engineering research conference

Hussein MI, Petering MEH (2013) Linear penalty relation in genetic based algorithms in block relocation problem weights. In: Krishnamurthy A, Chan WKV (eds) Proceedings of the 2013 industrial and systems engineering research conference, pp 1245–1254

Jang DW, Kim SW, Kim KH (2013) The optimization of mixed block stacking requiring relocations. Int J Prod Econom 143:256–262

Jansen K (2003) The mutual exclusion scheduling problem for permutation and comparability graphs. Inform Comput 180(2):71–81

Ji M, Guo W, Zhu H, Yang Y (2015) Optimization of loading sequence and rehandling strategy for multi-quay crane operations in container terminals. Transp Res Part E Logist Trans Rev 80:1–19

Jin B, Lim A, Zhu W (2011) A greedy look-ahead heuristic for the container relocation problem. Lect Notes Comput Sci 7906:181–190

Jin B, Zhu W, Lim A (2015) Solving the container relocation problem by an improved greedy look-ahead heuristic. Eur J Oper Res 240:837–847

Jovanovic R, Voß S (2014) A chain heuristic for the blocks relocation problem. Comput Ind Eng 75:79–86

Jovanovic R, Tuba M, Voß S (2017) A multi-heuristic approach for solving the pre-marshalling problem. Cent Eur J Oper Res 25(1):1–28

Kang J, Oh M, Ahn E, Ryu K, Kim K (2006) Planning for intra-block remarshalling in a container terminal. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp 1211–1220

Kim K, Bae J (1998) Re-marshaling export containers in port container terminals. Comput Ind Eng 35(3):655–658

Kim KH, Hong GP (2006) A heuristic rule for relocating blocks. Comput Oper Res 33(4):940–954

Ku D, Arthanari TS (2016a) Container relocation problem with time windows for container departure. Eur J Oper Res 252(3):1031–1039

Ku D, Arthanari TS (2016b) On the abstraction method for the container relocation problem. Comput Oper Res 68:110–122

Lee Y, Chao SL (2009) A neighborhood search heuristic for pre-marshalling export containers. Eur J Oper Res 196(2):468–475

Lee Y, Hsu N (2007) An optimization model for the container pre-marshalling problem. Comput Oper Res 34(11):3295–3313

Lee Y, Lee YJ (2010) A heuristic for retrieving containers from a yard. Comput Oper Res 37:1139–1147

Legato P, Mazza RM, Trunfio R (2013) Medcenter container terminal SpA uses simulation in housekeeping operations. Interfaces 43(4):313–324

Lehnfeld J, Knust S (2014) Loading, unloading and premarshalling of stacks in storage areas: survey and classification. Eur J Oper Res 239(2):297–312

Lin DY, Lee YJ, Lee Y (2015) The container retrieval problem with respect to relocation. Trans Res Part C 52:132–143

Nishi T, Konishi M (2010) An optimisation model and its effective beam search heuristics for floor-storage warehousing systems. Int J Produc Res 48(7):1947–1966

Olsen M, Gross A (2014) Average case analysis of blocks relocation heuristics. Lect Notes Comput Sci 8760:81–92

Park K, Park T, Ryu K (2009) Planning for remarshaling in an automated container terminal using cooperative coevolutionary algorithms. In: SAC 2009 – Honolulu, Hawai, pp 1098–1105

Park T, Kim J, Ryu K (2010) Iterative replanning using genetic algorithms for remarshaling in a container terminal. In: Hamza H (ed) Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, pp 22–28

Park K, Park T, Ryu K (2013) Planning for selective remarshaling in an automated container terminal using coevolutionary algorithms. Int J Ind Eng 20:176–187

Petering MEH, Hussein MI (2013) A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem. Eur J Oper Res 231:120–130

Prandtstetter M (2013) A dynamic programming based branch-and-bound algorithm for the container pre-marshalling problem. Technical repot, IT Austrian institute of technology

Rei R, Pedroso JP (2012a) Heuristic search for the stacking problem. Int Trans Oper Res 19(3):379–395

Rei R, Pedroso JP (2012b) Tree search for the stacking problem. Ann Oper Res 203:371–388

Ren Z, Zhang C (2015) An iterative three-stage algorithm for the pre-marshalling problem in container terminals. In: 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp 1232–1236

Rendl A, Prandtstetter M (2013) Constraint models for the container pre-marshaling problem. In: Katsirelos G, Quimper C (eds) Modref 2013: 12th International Workshop on Constraint Modelling and Reformulation, pp 44–56

Romero AG, Alquézar R (2004) To block or not to block? Lect Notes Comput Sci 3315:134–143

Saccone S, Siri S (2009) An integrated simulation-optimization framework for the operational planning of seaport container terminals. Math Comput Model Dyn Syst 15(3):275–293

Schwarze S, Voß S (2015) A note on alternative objectives for the blocks relocation problem, working paper. University of Hamburg

Shin E, Kim K (2015) Hierarchical remarshaling operations in block stacking storage systems considering duration of stay. Comput Ind Eng 89:43–52

Stahlbock R, Voß S (2008) Operations research at container terminals: a literature update. OR Spectrum 30(1):1–52

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

Tanaka S, Mizuno F (2015) Dominance properties for the unrestricted block relocation problem and their application to a branch-and-bound algorithm. In: 2015 IEEE International Conference on Automation Science and Engineering (CASE), pp 509–514

Tanaka S, Takii K (2014) A faster branch-and-bound algorithm for the block relocation problem. In: International Conference on Automation Science and Engineering (CASE). IEEE, Piscataway, pp 7–12

Tanaka S, Takii K (2016) A faster branch-and-bound algorithm for the block relocation problem. IEEE Trans Automat Sci Eng 13(1):181–190

Tang L, Ren H (2010) Modelling and a segmented dynamic programming-based heuristic approach for the slab stack shuffling problem. Comput Oper Res 37:368–375

Tang L, Zhao R, Liu J (2012) Models and algorithms for shuffling problems in steel plants. Naval Res Logist 59(7):502–524

Tang L, Jiang W, Liu J, Dong Y (2015) Research into container reshuffling and stacking problems in container terminal yards. IIE Trans 47(7):751–766

Tierney K, Malitsky Y (2015) An algorithm selection benchmark of the container pre-marshalling problem. Lect Notes Comput Sci 8994:17–22

Tierney K, Voß S (2016) Solving the robust container pre-marshalling problem. Lect Notes Comput Sci 9855:131–145

Tierney K, Pacino D, Voß S (2017) Solving the pre-marshalling problem to optimality with A* and IDA*. Flex Serv Manuf J 29(2):223–259

Tricoire F, Fechter J, Beham A (2016) New solution methods for the blocks relocation problem. Department of Business Administration, University of Vienna, Working Paper. http://www.optimization-online.org/DB_FILE/2016/03/5365.pdf

Tus A, Rendl A, Raidl G (2015) Metaheuristics for the two-dimensional container pre-marshalling problem. In: International Conference on Learning and Intelligent Optimization. Springer, Berlin, pp 186–201

Ünlüyurt T, Aydin C (2012) Improved rehandling strategies for the container retrieval process. J Adv Trans 46:378–393

van Brink M, van der Zwaan R (2014) A branch and price procedure for the container premarshalling problem. Lect Notes Comput Sci 8737:798–809

Voß S (2012) Extended mis-overlay calculation for pre-marshalling containers. Lect Notes Comput Sci 7555:86–91

Wan YW, Liu J, Tsai PC (2009) The assignment of storage locations to containers for a container stack. Naval Res Logist 56(8):699–713

Wang N, Jin B, Lim A (2015) Target-guided algorithms for the container pre-marshalling problem. Omega 53:67–77

Wang N, Jin B, Zhang Z, Lim A (2017) A feasibility-based heuristic for the container premarshalling problem. Eur J Oper Res 256:90–101

Woo JW, Kim KH (2011) Estimating the space requirement for outbound container inventories in port container terminals. Int J Produ Econ 133:293–301

Wu KC, Ting CJ (2010) A beam search algorithm for minimizing reshuffle operations at container yards. In: Proceedings of the International Conference on Logistics and Maritime Systems, September 15–17, Busan, Korea, pp 703–710

Wu KC, Hernández R, Ting CJ (2010) Applying tabu search for minimizing reshuffle operations at container yards. J Eastern Asia Soc Trans Stud 8:2379–2393

Yun W, Choi Y (1999) Simulation model for container-terminal operation analysis using an object-oriented approach. Int J Prod Econ 59(1):221–230

Zäpfel G, Wasner M (2006) Warehouse sequencing in the steel supply chain as a generalized job shop model. Int J Prod Econ 104(2):482–501

Zehendner E, Feillet D (2014) A branch and price approach for the container relocation problem. Int J Prod Res 52(24):7159–7176

Zehendner E, Caserta M, Feillet D, Schwarze S, Voß S (2015) An improved mathematical formulation for the blocks relocation problem. Eur J Oper Res 245(2):415–422

Zehendner E, Feillet D, Jaillet P (2017) An algorithm with performance guarantee for the online container relocation problem. Eur J Oper Res 259:48–62

Zhang C, Liu J, Wan Y, Murty KG, Linn RJ (2003) Storage space allocation in container terminals. Trans Res B 37(10):883–903

Zhang H, Guo S, Zhu W, Lim A, Cheang B (2010) An investigation of IDA* algorithms for the container relocation problem. Lect Notes Comput Sci 6096:31–40

Zhang R, Jiang Z, Yun W (2015) Stack pre-marshalling problem: a heuristic-guided branch-and-bound algorithm. Int J Ind Eng 22(5):509–523

Zhang R, Liu S, Kopfer H (2016) Tree search procedures for the blocks relocation problem with batch moves. Flex Services Manuf J 28(3):397–424

Zhu M, Fan X, He Q (2010) A heuristic approach for transportation planning optimization in container yard. In: Proceedings of industrial engineering and engineering management (IEEM). IEEE, Piscataway, pp 1766–1770

Zhu W, Qin H, Lim A, Zhang H (2012) Iterative deepening A* algorithms for the container relocation problem. IEEE Trans Autom Sci Eng 9:710–722