OR Spectrum (2010) 32:569–591
DOI 10.1007/s00291-009-0191-6

REGULAR ARTICLE

Container sequencing for quay cranes with internal reshuffles

Frank Meisel · Matthias Wichmann

Published online: 29 December 2009

Abstract Fast handling of vessels is one of the most important goals in container terminal operations planning. In recent studies, quay crane double cycling has been investigated to accelerate the service of vessels. In our paper, we show that the service process can be further accelerated by changing the treatment of so-called reshuffle containers. Reshuffle containers have to be removed from their position in the vessel only to gain access to containers stacked below them. Our approach enables to reposition reshuffle containers directly within the bay of a vessel, referred to as internal reshuffles, instead of temporarily unloading them. A mathematical problem formulation and a heuristic solution method, namely a greedy randomized adaptive search procedure, are provided for planning crane operations under internal reshuffles. Computational tests prove that the consideration of internal reshuffles leads to a further shortening of vessel handling times compared to a sole application of crane double cycling.

Keywords Container terminal · Container sequencing problem · Double cycling · Internal reshuffle · GRASP

1 Introduction

Since the offspring of containerization in maritime transportation in the 1950s, continuous efforts have been made to increase the size of container vessels with the

F. Meisel (✉)
School of Economics and Business, Martin-Luther-University Halle-Wittenberg,
Gr. Steinstr. 73, 06108 Halle, Germany
e-mail: frank.meisel@wiwi.uni-halle.de

M. Wichmann
Institute of Automotive Management and Industrial Production,
Technical University Braunschweig, Braunschweig, Germany
e-mail: ma.wichmann@tu-bs.de

goal to decrease the transport cost per container unit. Nowadays, large vessels carry up to 11,000 20-foot equivalent container units (TEUs). Often several thousand containers have to be transshipped for a single vessel at a container terminal (CT). Even in situations where such huge numbers of containers must be transshipped, the vessel operators expect fast service processes to ensure that the economic advantages of large vessels are not canceled out by long port stay times. The minimization of vessel handling times is, therefore, one of the most important goals of CT operators when planning service processes of a terminal. The key to minimum vessel handling times is often a high productive utilization of the quay cranes (QCs), which transship the containers between vessels and the terminal area. In our paper, we provide a methodology for planning the transshipment operations of one QC at one bay of a vessel. The objective is to minimize the total time needed to fulfill the container transshipments at the bay. The consideration is confined to single bays, because terminal operators avoid frequent repositioning of the slow-moving cranes by instructing QCs to process one bay completely before moving to a next bay.

Planning the transshipment operations for one bay of a vessel means to determine a sequence of container moves that converts an *arrival configuration* of the considered bay, i.e., the stowage of containers on-board the vessel at the time of arrival, into a *departure configuration*, i.e., the projected stowage of containers on-board at the time of departure. Arrival configuration and departure configuration are preset for the planning of transshipment operations, because they are determined by so-called stowage planning, which is solved ahead and detached of crane operations planning in nowadays CTs. Figure 1 shows an illustrating example of configurations where containers in a bay are stored in four stacks up to three tiers high. Individual containers in a configuration are described by their slot position (specified by a stack index and a tier index) and their container class. For the transshipment process, containers are classified as import containers, export containers, fixed containers, and reshuffle containers. *Import containers* have to be unloaded from the vessel at the considered terminal whereas *export containers* have to be loaded. Hence, import containers will appear in the arrival configuration only, whereas export containers appear in the departure configuration only. *Fixed containers* remain at the vessel without being moved at all,

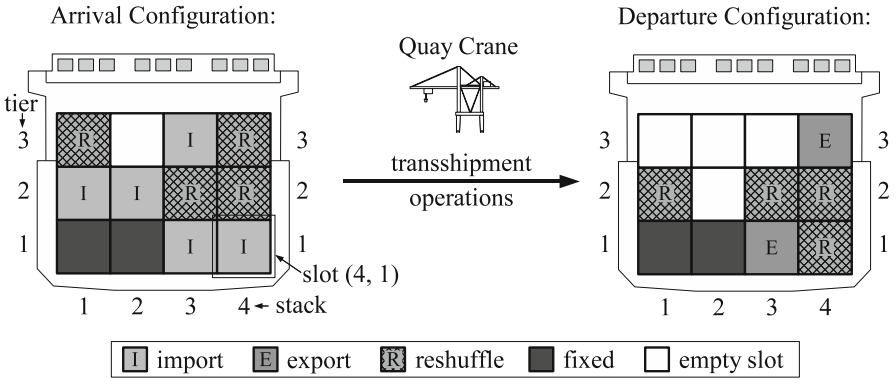


Fig. 1 Configurations of a bay (cross-sectional view of a vessel)

which means they appear in both configurations at identical slot positions. *Reshuffle containers* (also called rehandles) are not dedicated to the considered terminal but they stay on top of import containers, or export containers have to be placed below of them. Hence, since containers in a stack can be accessed only from top, reshuffle containers must be (temporarily) removed from their slots to access other containers. For this purpose, reshuffle containers are usually unloaded from the vessel, stored in a buffer area in the CT's yard, and later reloaded to the vessel such that they are on-board again at the time of departure. Given that all reshuffle containers belong to a same container class, they are considered to be exchangeable, i.e., every reshuffle container found in the arrival configuration can be positioned at any reshuffle slot in the departure configuration. Obviously, the number of reshuffle containers is the same in the arrival and the departure configuration, but their slot positions may differ, see Fig. 1.

In practice, rules of thumb are applied to determine a sequence of container moves that converts the arrival configuration of a bay into the departure configuration. Common rules are to clear a bay by unloading containers tier wise or stack wise, starting at the quayside and ending at the seaside of the vessel. When unloading is completed, export containers are loaded by reversing the unloading pattern. Rules of thumb attract by negligible computational effort and simple instructions for crane drivers. However, the achieved crane productivity is quite low. One reason for low crane productivity is the temporal decoupling of loading and unloading operations (*single cycling*), which requires an unproductive travel of the empty QC spreader in-between every two consecutive container moves. In contrast, by alternating loading and unloading operations (*double cycling*), empty movement of the QC spreader is reduced and the service of a bay is accelerated. The potential of QC double cycling has been investigated in recent studies of Goodchild and Daganzo (2006, 2007). In their approach, loading and unloading of different stacks is parallelized in the crane schedule, which enables to process the stacks in double cycling mode. Computational tests demonstrate that double cycling creates significant gains in crane productivity. However, a second source of low crane productivity remains even in this innovative approach. Since QC operations are planned on the basis of complete container stacks, the reshuffle containers in a stack are unloaded from the vessel to the quay and reloaded at a later point in time, referred to as *external reshuffles*. A preferable way of handling reshuffle containers is to reposition them directly within the bay whenever possible, which we refer to as *internal reshuffles*. Internal reshuffles accelerate the service of a bay, because the containers are moved only one time instead of two times. To consider internal reshuffles in a crane schedule, QC operations must be planned on the basis of single container moves.

In our paper, we present a methodology for planning crane operations that enables to exploit both, the potential of QC double cycling as well as the potential of internal reshuffles. The task is to find a feasible sequence of container moves that converts a given arrival configuration of a bay into a given departure configuration within minimum service time. We refer to this optimization problem as the container sequencing problem (CSP).

The paper is organized as follows. A literature review is given in Sect. 2. The CSP is described and mathematically formulated in Sect. 3. A heuristic solution method is

presented in Sect. 4. Results of a computational study are reported in Sect. 5. Section 6 concludes the paper.

2 Literature review

Various CT operations planning problems are investigated by recent research, see Stahlbock and Voß (2008). While the CSP has not yet been addressed itself, its core issues like crane operations planning, double cycling, and container reshuffling have been partly investigated in the context of stowage planning, yard crane scheduling, and QC scheduling.

Stowage planning means to arrange containers on board a vessel by assigning a slot position to every container that has to be transported from a considered terminal to successive terminals on a vessel's route, see Avriel and Penn (1993). The assignment of containers to slots has to respect container attributes like size and weight Ambrosino et al. (2004) as well as stability constraints of the vessels Imai et al. (2006). A typical objective is the minimization of reshuffling of containers, which occurs whenever a container dedicated to a late terminal in the vessel's route is stowed on top a container that has to be unloaded earlier, e.g., Avriel and Penn (1993) and Kang and Kim (2002). Since scheduling of transshipment operations is out of scope of stowage planning, reshuffles are generally supposed to be handled as external reshuffles. The solutions of stowage planning serve as input for the CSP, namely as the arrival and the departure configurations of bays.

Reshuffling of containers occurs also in the yard of a CT, because of the stacking of containers in the yard bays. Yard crane scheduling for retrieving containers of a yard bay at a minimum number of reshuffles has been investigated by Kim and Hong (2006) and Caserta et al. (2008). In their problem, the task is to decide on the reshuffling of containers within the yard bay, such that the containers can be retrieved in a preset order. Contrasting the CSP, the processing order of the containers is given and merely retrieval operations are considered, which prevents the investigation of crane double cycling issues. Further studies that investigate the interrelations of stowage planning, container load sequencing, and yard crane scheduling consider either the unloading or the loading process of a vessel and thus, also neglect crane double cycling, see Imai et al. (2002), Ambrosino and Sciomachen (2003), and Kim et al. (2004).

Scheduling problems for QC operations have been investigated intensively in the recent literature, see the survey of Bierwirth and Meisel (2010). In most studies, operations are planned for a set of cranes that jointly serve a vessel. To cope with the complexity of scheduling several thousand container moves on multiple QCs, the total transshipment workload of a vessel is usually partitioned into a set of aggregated QC tasks. Tasks most often refer to all containers to load and unload at a bay (e.g., Daganzo, 1989; Zhu and Lim, 2006; Liu et al., 2006; Lim et al., 2007) or to a group of containers within a bay (e.g., Kim and Park, 2004; Moccia et al., 2006; Sammarra et al., 2007; Bierwirth and Meisel, 2009). Double cycling issues and the handling of reshuffle containers are, however, not considered on these aggregation levels of QC operations planning.

Crane double cycling issues have been incorporated into QC scheduling first by Goodchild and Daganzo (2006). In this approach, the service process of a single QC

at a bay of a container vessel is considered. Every container stack in the considered bay is represented by two precedence-related tasks, one for unloading the stack and one for loading the stack. The task processing times are determined by the number of containers to (un-)load. Crane double cycling is realized in a crane schedule by parallelization of unloading tasks and loading tasks of different stacks. The problem is to find a sequence for processing the stacks, which minimizes the makespan of the schedule by maximizing crane double cycling. For this purpose, the authors formulate the problem as a two-machine flow shop scheduling problem. This problem is solved exactly by the rule of Johnson (1954), yielding optimal schedules where a crane performs a maximum number of double cycles and the bay is served within minimum crane operation time. In Goodchild and Daganzo (2007), the economic advantages of QC double cycling are proven in terms of increased crane productivity, berth utilization, and vessel utilization. Zhang and Kim (2009) extend the approach of Goodchild and Daganzo by considering the impact of hatch covers on QC operations. Johnson's rule is modified and extended by local search to solve the problem. While the stack-based QC scheduling enables to incorporate double cycling issues, Goodchild and Daganzo (2006) admit that reshuffle containers are always treated as external reshuffles when transshipping containers stack-wise. For this reason, in our paper, moves of single containers are sequenced within the CSP to enable the consideration of internal reshuffles within crane operations planning. This complicates the planning process, because the problem size increases when container moves are sequenced instead of complete container stacks and additional constraints must be respected, such as the stacking dependent accessibility of containers. The enabled incorporation of internal reshuffles, however, promises solutions of shorter crane operation time.

3 The container sequencing problem

3.1 Problem description

In the CSP, we are given an arrival configuration and a departure configuration of containers in a bay. The problem is to find a sequence of container moves that converts the arrival configuration into the departure configuration within minimum service time of the bay. The service time of the bay is thereby defined as the total processing time of container moves and empty crane movement in a solution. A solution to the problem must respect the stacking dependent accessibility of containers. We distinguish five types of container moves that can be performed by the QC, denoted by VY, YV, VB, BV, and VV:

- VY An import container is unloaded from the Vessel to the Yard of the terminal.
- YV An export container is retrieved from the Yard and loaded onto the Vessel.
- VB A reshuffle container is unloaded from the Vessel to a Buffer area in the yard.
- BV A reshuffle container is retrieved from the Buffer area in the yard and reloaded onto the Vessel.
- VV A reshuffle container is internally reshuffled by repositioning it from one slot in the considered Vessel bay to another slot in the Vessel bay.

We use the CSP instance shown in Fig. 1 as a running example for illustrating the problem and the solution method. Five import containers, four reshuffle containers, and two export containers have to be transshipped in this instance. The containers at slots (1,1) and (2,1) are fixed and not affected by the transshipment process.

A solution to this CSP instance is shown in Table 1. The service of the bay is fulfilled in a sequence of 13 container moves. In the first move, the reshuffle container at slot (4,3) of the arrival configuration is retrieved and unloaded to the buffer area in the yard (move type VB). In the second move, the crane unloads the reshuffle at slot (4,2). It then unloads the import container at slot (4,1) to the yard (move type VY) before loading an externally buffered reshuffle to slot (4,1) in the fourth move, and so on. Note that slot positions in the bay are described precisely by a stack-index and a tier-index, whereas the positions of containers in the yard and in the buffer are subsumed by capitals Y and B, because precise yard and buffer positions are out of scope of the QC operations. The described move sequence respects the stacking dependent accessibility of containers within the stacks of the bay. Internal reshuffles are realized in moves 6 and 9 of the solution, where the reshuffle containers at slots (3,2) and (1,3) in the arrival configuration are directly repositioned within the bay to slots (4,2) and (3,2) in the departure configuration, respectively. The two other reshuffle containers are externally reshuffled by unloading them in moves 1 and 2 and reloading them in moves 4 and 11. Note that an internal reshuffle requires only one container move of type VV compared to two moves of type VB and BV for an external reshuffle. Double cycling is performed in the move-subsequences (3, 4, 5), (7, 8), and (10, 11, 12, 13) of the solution, where unloading and loading operations are performed alternately.

To assess a container move sequence in terms of the resulting service time of the bay, processing times of container moves and empty spreader movement must be provided as a further input to the CSP. To ease the formulation of the CSP, we refrain from considering slot-specific container move processing times. Instead, we assume that the processing time of a container move depends merely on the type of the move. We derive a rough estimate of processing times, if we assume that a QC performs about 30 moves per hour. This is the typical range of nowadays crane productivity, cf. Chu and Huang (2002), where neither double cycling nor internal reshuffling is applied. Hence, a single cycle takes two minutes on average. Let us assume that this time span is composed from 100 s for the container move (including lifting, moving, and lowering the container), and 20 s for the empty movement of the crane spreader to the next container. Hence, every container move where a container is transshipped between the vessel and the quay or vice versa, i.e., moves of type VB, VY, YV, or

Table 1 A sequence of container moves

move:	1	2	3	4	5	6	7	8	9	10	11	12	13	
source:	(4,3)	(4,2)	(4,1)	B	(3,3)	(3,2)	(3,1)	Y	(1,3)	(1,2)	B	(2,2)	Y	
proc.time:	100 20	100 20	100 10	100 10	100 20	90 10	100 10	100 10	90 10	100 10	100 10	100 10	100	$\rightarrow Z = 1430$
sink:	B	B	Y	(4,1)	Y	(4,2)	Y	(3,1)	(3,2)	Y	(1,2)	Y	(4,3)	
type:	VB	VB	VY	BV	VY	VV	VY	YV	VV	VY	BV	VY	YV	

Table 2 Processing times of container moves and empty crane spreader movement

(a)

Processing time d^t of a container move of type t [seconds]

t	d^t
VY	100
YV	100
VB	100
BV	100
VV	90

(b)

Time d^{tu} needed for empty crane spreader movement in-between a move of type t and a consecutive move of type u

$t\downarrow \setminus u \rightarrow$	VY	YV	VB	BV	VV
VY	20	10	20	10	20
YV	10	20	10	20	10
VB	20	10	20	10	20
BV	10	20	10	20	10
VV	10	20	10	20	10

BV, takes 100 s. Internal reshuffles (move type VV) get assigned a processing time of 90 s, because the containers are repositioned within the bay over a comparably short distance. The time needed for empty crane spreader movement in-between two consecutive container moves depends on the types of the involved moves. We assume a processing time of 20 s for empty spreader movement, if the spreader has to be moved from the quay to the vessel or vice versa, e.g., in-between a VY-move and a VB-move, and to 10 s otherwise. In the later case, the time is needed to reposition the crane spreader to another slot position in the bay or to wait for the exchange of horizontal transport vehicles at the transition point at the quay. The processing times for container moves and empty crane spreader movement are summarized in Table 2.

In the solution in Table 1, the processing times of the container moves are shown by the labeling of the downward oriented arcs and the time needed for empty crane spreader movement is shown by the labeling of the upward oriented arcs. The overall bay service time is $Z = 23.83$ min (1,430 s) in this solution.

3.2 Assumptions

A1 Import containers, export containers, and internal reshuffles are processed by exactly one container move. This forbids, for example, to reposition an import container to another slot in the bay before it is finally unloaded. External reshuffles are processed in exactly two container moves, one to unload the container to the buffer and one to load the container back to the vessel. Note that for faster services, external reshuffles could be handled through indirect internal reshuffles, i.e., by first repositioning a container temporarily to any bay slot before later moving it to a meanwhile cleared reshuffle slot in the departure configuration. However, the benefit of such moves is limited, because the temporarily stored containers hinder the processing of the occupied stacks and container positions must be tracked all the time. Therefore, this way of handling reshuffles is omitted for reasons of practicability.

A2 Reshuffle containers are assumed to be exchangeable, i.e., every reshuffle container found in the arrival configuration can be positioned at any reshuffle slot in the departure configuration. Moreover, the number of reshuffle containers in a stack is the same in the arrival and the departure configuration. This ensures

that after the unloading of a stack, a sufficient number of reshuffles is available for subsequently loading the stack completely.

A3 The arrival configuration, the departure configuration, and every possible in-between load configuration of the bay are feasible with respect to the stability constraints of the vessel. This means that crane operations cannot cause instabilities and, therefore, stability issues are not considered in the CSP.

A4 A hatchcoverless container vessel is considered, i.e., containers below and above deck are not separated by hatch covers, providing more flexibility in the application of crane double cycling, see Bendall and Stent (1996).

A5 Horizontal transport vehicles are always available at the QC, waiting to receive unloaded containers and providing containers to load. Hence, the service time of the bay depends on the move sequence of the QC but not on the transport operations of the vehicles.

3.3 Notation

Input data describing the considered bay and its load configurations:

m number of stacks in the bay, $m = 4$ in the example of Fig. 1,
n number of tiers in the bay, $n = 3$ in the example,
A^{I} set of slots that hold an import container in the arrival configuration, $A^{\mathrm{I}} = \{(1, 2), (2, 2), (3, 1), (3, 3), (4, 1)\}$ in the example,
A^{R} set of slots that hold a reshuffle container in the arrival configuration, $A^{\mathrm{R}} = \{(1, 3), (3, 2), (4, 2), (4, 3)\}$ in the example,
A set of slots to remove a container from, i.e., slots that hold an import container or a reshuffle container in the arrival configuration, $A = A^{\mathrm{I}} \cup A^{\mathrm{R}}$,
D^{E} set of slots that hold an export container in the departure configuration, $D^{\mathrm{E}} = \{(3, 1), (4, 3)\}$ in the example,
D^{R} set of slots that hold a reshuffle container in the departure configuration, $D^{\mathrm{R}} = \{(1, 2), (3, 2), (4, 1), (4, 2)\}$ in the example,
D set of slots to place a container at, i.e., slots that hold an export container or a reshuffle container in the departure configuration, $D = D^{\mathrm{E}} \cup D^{\mathrm{R}}$.

Input data describing the transshipment operations:

T set of container move types, $T = \{\mathrm{VY}, \mathrm{YV}, \mathrm{VB}, \mathrm{BV}, \mathrm{VV}\}$,
d^t processing time of a container move of type $t \in T$, see Table 2a,
d^{tu} time needed for empty crane spreader movement in-between a container move of type t and a container move of type u with $t, u \in T$, see Table 2b,
l an upper bound on the number of moves needed to process the bay,
$K, \overline{K}$ index sets of moves, $K = \{1, 2, \ldots l\}$, $\overline{K} = K \backslash \{l\}$.

Note that the number of container moves in a solution to a CSP instance cannot be determined a priori, because reshuffle containers may be either internally reshuffled (one move per container) or externally reshuffled (two moves per container). Nevertheless, for a mathematical formulation of the CSP, an upper bound l on this number must be provided. This upper bound is calculated as the number of moves needed to

process the bay in case that every reshuffle container is externally buffered, i.e.

$$l = |A^{\mathrm{I}}| + |A^{\mathrm{R}}| + |D^{\mathrm{R}}| + |D^{\mathrm{E}}| = |A| + |D|. \tag{1}$$

For the example in Fig. 1, Eq. (1) delivers $l = 15$ moves.

Decision variables:

x^{V}_{ijk} binary, 1 iff the container at bay slot (i, j) is picked in move $k \in K$,
x^{Y}_{k} binary, 1 iff a container is picked from the yard in move k,
x^{B}_{k} binary, 1 iff a container is picked from the buffer in move k,
y^{V}_{ijk} binary, 1 iff a container is dropped at bay slot (i, j) in move k,
y^{Y}_{k} binary, 1 iff a container is dropped at the yard in move k,
y^{B}_{k} binary, 1 iff a container is dropped at the buffer in move k,
τ^{t}_{k} binary, 1 iff container move k is of move type $t \in T$,
e^{tu}_{k} binary, 1 iff container move k is of type $t \in T$ and move $k+1$ is of type $u \in T$, i.e., $\tau^{t}_{k} = \tau^{u}_{k+1} = 1$, indicates the required empty crane movement between container moves k and $k+1$,
b_k integer, number of externally buffered reshuffle containers at the end of move k.

3.4 Optimization model

$$\text{minimize } Z = \sum_{k \in K} \sum_{t \in T} d^{t} \cdot \tau^{t}_{k} + \sum_{k \in \overline{K}} \sum_{t,u \in T} d^{tu} \cdot e^{tu}_{k} \tag{2}$$

$$\sum_{k \in K} x^{\mathrm{V}}_{ijk} = 1 \quad \forall (i, j) \in A \tag{3}$$

$$\sum_{k \in K} y^{\mathrm{V}}_{ijk} = 1 \quad \forall (i, j) \in D \tag{4}$$

$$\sum_{k \in K} x^{\mathrm{V}}_{i,j+1,k} \cdot k \leq \sum_{k \in K} x^{\mathrm{V}}_{ijk} \cdot k \quad \forall (i, j), (i, j+1) \in A \tag{5}$$

$$\sum_{k \in K} y^{\mathrm{V}}_{ijk} \cdot k \leq \sum_{k \in K} y^{\mathrm{V}}_{i,j+1,k} \cdot k \quad \forall (i, j), (i, j+1) \in D \tag{6}$$

$$\sum_{k \in K} x^{\mathrm{V}}_{ijk} \cdot k \leq \sum_{k \in K} y^{\mathrm{V}}_{ijk} \cdot k \quad \forall (i, j) \in A \cap D \tag{7}$$

$$\sum_{t \in T} \tau^{t}_{k} \leq 1 \quad \forall k \in K \tag{8}$$

$$\tau_k^{\mathrm{VY}} = \sum_{(i,j)\in A^{\mathrm{I}}} x_{ijk}^{\mathrm{V}} \quad \forall k \in K \tag{9}$$

$$\tau_k^{\mathrm{VY}} = y_k^{\mathrm{Y}} \quad \forall k \in K \tag{10}$$

$$\tau_k^{\mathrm{YV}} = x_k^{\mathrm{Y}} \quad \forall k \in K \tag{11}$$

$$\tau_k^{\mathrm{YV}} = \sum_{(i,j)\in D^{\mathrm{E}}} y_{ijk}^{\mathrm{V}} \quad \forall k \in K \tag{12}$$

$$\tau_k^{\mathrm{VV}} + \tau_k^{\mathrm{VB}} = \sum_{(i,j)\in A^{\mathrm{R}}} x_{ijk}^{\mathrm{V}} \quad \forall k \in K \tag{13}$$

$$\tau_k^{\mathrm{VB}} = y_k^{\mathrm{B}} \quad \forall k \in K \tag{14}$$

$$\tau_k^{\mathrm{BV}} = x_k^{\mathrm{B}} \quad \forall k \in K \tag{15}$$

$$\tau_k^{\mathrm{VV}} + \tau_k^{\mathrm{BV}} = \sum_{(i,j)\in D^{\mathrm{R}}} y_{ijk}^{\mathrm{V}} \quad \forall k \in K \tag{16}$$

$$\sum_{t\in T} \tau_k^t \geq \sum_{t\in T} \tau_{k+1}^t \quad \forall k \in \overline{K} \tag{17}$$

$$e_k^{tu} \geq \tau_k^t + \tau_{k+1}^u - 1 \quad \forall k \in \overline{K},\ t, u \in T \tag{18}$$

$$b_0 = 0 \tag{19}$$

$$b_k = b_{k-1} - x_k^{\mathrm{B}} + y_k^{\mathrm{B}} \quad \forall k \in K \tag{20}$$

$$b_k \geq 0, \text{integer} \quad \forall k \in K \tag{21}$$

$$x_{ijk}^{\mathrm{V}}, x_k^{\mathrm{Y}}, x_k^{\mathrm{B}}, y_{i'j'k}^{\mathrm{V}}, y_k^{\mathrm{Y}}, y_k^{\mathrm{B}}, \tau_k^t, e_k^{tu} \in \{0; 1\} \quad \forall (i, j) \in A, (i', j') \in D, k \in K,\ t, u \in T \tag{22}$$

The pursued objective is to minimize the service time of the bay, which is calculated in (2) as the total processing time of container moves and empty crane movement.

Constraints (3) ensure that every import and every reshuffle container in the arrival configuration is retrieved exactly once by the QC. Constraints (4) ensure that exactly one container is placed at every export and every reshuffle slot in the departure configuration.

The stacking of containers within the bay necessitates precedence constraints (5)–(7). Constraints (5) consider pairs of containers to unload that stay on top of each other in a stack. They ensure for every of these pairs that the container in tier $j+1$ is retrieved in an earlier move than the container in tier j. Similarly, (6) ensures that a container is placed at slot (i, j) before a container is placed at slot $(i, j+1)$. Constraints (7) ensure that a container is removed from a slot before a new container is loaded to this slot.

From Constraints (8) to (16) at most one container is moved in move $k \in K$. The constraints ensure that a picked container is dropped again and that the operation represents one of the allowed move types. They, therefore, relate one non-zero x variable, one non-zero y variable, and one non-zero τ variable. Constraints (8) choose the type of container move k by setting one variable τ_k to 1. If, for example, an import container is unloaded in move k, τ_k^{VY} is set to 1 and, then, Constraints (9) ensure that exactly one import container is picked from the vessel, and Constraints (10) ensure that the container is transferred to the yard. Similarly, (11) and (12) handle the loading of export containers. Constraints (13)–(16) handle the reshuffling of containers. Depending on the selected τ_k variable, reshuffles are either externally buffered by (13) and (14) before being reloaded by (15) and (16) in a later move, or internally reshuffled by Constraints (13) and (16). If no move type is selected in (8), i.e., $\sum_{t \in T} \tau_k^t = 0$, no container is handled in move k.

In case that less than l moves are contained in a solution, Constraints (17) enforce that the decision variables represent an uninterrupted sequence of consecutive container moves. Constraints (18) identify the type of empty movement between any pair of consecutively performed container moves by setting e_k^{tu} to 1, if the kth container move is of type $t \in T$ and move $k+1$ is of type $u \in T$. Constraint (19) initializes the buffer area for external reshuffles empty. Constraints (20) calculate the resulting number of externally buffered reshuffle containers after a move.

Domains of the decision variables are defined in (21) and (22). Note that the integrality constraints for variables b_k, e_k^{tu} and τ_k^t can be dropped, because these variables inevitably take integer values from their relation to variables x and y in the model. Unfortunately, even if these constraints are dropped, standard solvers like ILOG CPLEX merely solve instances of very small size. The optimization model for the example in Fig. 1 is solved by CPLEX 11 within one minute of computation time on a PC P-IV 2.8 GHz. The optimal solution, which is the one shown in Table 1, is represented by the following non-zero decision variables for picking and dropping containers in 13 moves: $x^{\mathrm{V}}_{4,3,1} = y^{\mathrm{B}}_1 = x^{\mathrm{V}}_{4,2,2} = y^{\mathrm{B}}_2 = x^{\mathrm{V}}_{4,1,3} = y^{\mathrm{Y}}_3 = x^{\mathrm{B}}_4 = y^{\mathrm{V}}_{4,1,4} = x^{\mathrm{V}}_{3,3,5} = y^{\mathrm{Y}}_5 = x^{\mathrm{V}}_{3,2,6} = y^{\mathrm{V}}_{4,2,6} = x^{\mathrm{V}}_{3,1,7} = y^{\mathrm{Y}}_7 = x^{\mathrm{Y}}_8 = y^{\mathrm{V}}_{3,1,8} = x^{\mathrm{V}}_{1,3,9} = y^{\mathrm{V}}_{3,2,9} = x^{\mathrm{V}}_{1,2,10} = y^{\mathrm{Y}}_{10} = x^{\mathrm{B}}_{11} = y^{\mathrm{V}}_{1,2,11} = x^{\mathrm{V}}_{2,2,12} = y^{\mathrm{Y}}_{12} = x^{\mathrm{Y}}_{13} = y^{\mathrm{V}}_{4,3,13} = 1$, and the non-zero buffer variables $b_1 = 1, b_2 = b_3 = 2$, and $b_4 = b_5 = b_6 = b_7 = b_8 = b_9 = b_{10} = 1$. While this small instance is solved quickly, CPLEX even fails to find an integer feasible solution within 2 h of computation when turning to instances of size $m = n = 10$. Hence, instances of practical size cannot be solved optimal by standard solvers within reasonable time. Although the complexity status of the CSP is an open question even when pursuing to minimize external reshuffling, the computational evidence of the bad performance of the CPLEX solver motivates a heuristic solution of CSP instances of practical size.

4 GRASP heuristic

We propose a multi-start meta-heuristic called greedy randomized adaptive search procedure (GRASP) for solving the CSP. The idea of GRASP is to build a feasible solution in a construction phase and improve this solution in a local search phase, where randomization is applied within the construction phase to generate not only a single but multiple different solutions, cf. Resende and Ribeiro (2003). GRASP is attractive for being adopted to the CSP, because problem specific knowledge can be implemented in the construction phase and in the local search phase. The procedure is terminated after repeating both phases for a preset number of iterations. The best found solution is returned by GRASP as the final solution to a CSP instance.

4.1 Construction phase

The constructor starts with an empty sequence of container moves. It iteratively extends the sequence by repeating three steps: (i) Building a list of candidate moves, (ii) selecting one move from the list, and (iii) updating the solution. It terminates as soon as the constructed sequence of container moves represents a complete service of the bay.

(i) Building the candidate list: In this step, moves that can feasibly extend the current partial solution are identified and stored in a so-called candidate list CL. Initially, when the partial solution is empty, CL contains unloading moves for those import or reshuffle containers that stay ontop the stacks in the arrival configuration and moves for loading export containers into empty stacks. In later iterations of the construction phase, moves become available for internal reshuffling of containers, for reloading externally buffered reshuffles, and for loading export containers into cleared stacks. More precisely, a move for unloading a container enters CL as soon as all containers staying ontop in the arrival configuration have been removed. A move for loading a container enters CL as soon as all containers stacked below in the departure configuration have been loaded. Internal reshuffle moves enter CL if an unprocessed reshuffle container is accessible and a reshuffle-slot of the departure configuration is ready for receiving a container. To guide the construction process toward promising solutions, certain moves are not taken up in CL, even if they can feasibly extend the current partial solution. Since loading of a stack cannot start before its unloading has been completed, frequent switching among different stacks to unload hinders fast service processes. Therefore, if the QC has begun to unload a stack, unloading containers from other stacks is postponed by excluding the corresponding moves from CL. As soon as the stack has been completely unloaded, all unloading moves enter CL again, enabling the selection of a new stack to unload. Feasible moves of type VV, YV, and BV are never restricted and always taken up in CL.
If CL remains empty in an iteration, it means that all containers have been processed and, thus, the already generated sequence of container moves represents a complete solution. In this case, the constructor terminates by returning the obtained solution.

(ii) Selection of a move: Exactly one move is selected from CL using a probabilistic selection process. For assigning an individual selection probability to move $i \in CL$, let δ_i be the increase of the objective function (2) that results from appending move i to the already generated partial solution. More precisely, δ_i is the sum of the processing time of move i and the time needed for empty spreader movement in-between move i and the last move of the current partial solution. If move i represents the unloading of a reshuffle container (move type VB), a penalty of d^{BV} is added to δ_i, which reflects that the external reshuffle must be reloaded at a later point in time. Having calculated δ values for all moves in CL, the selection probability of move i is calculated as

$$\psi_i = \frac{(1/\delta_i)}{\sum_{j \in CL} (1/\delta_j)}. \tag{23}$$

From the inverses in Eq. (23), the lower the δ-value of a move is, the higher is its assigned selection probability. Having calculated selection probabilities for the moves in CL, exactly one move is chosen using roulette wheel selection.

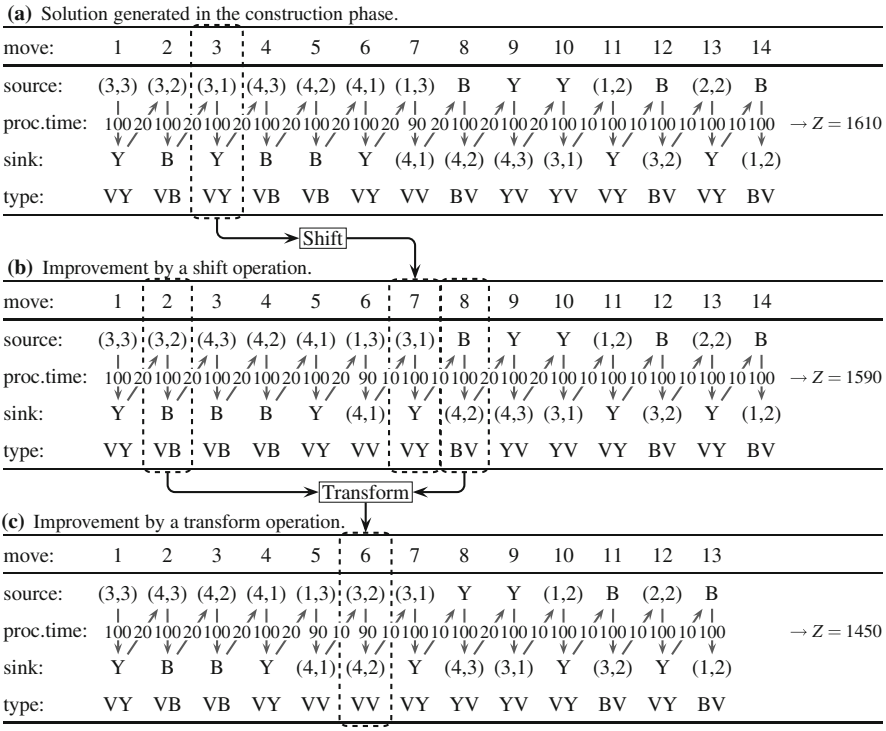
(iii) Update of the solution: The selected move is appended to the current partial solution. If the move is of type VB or BV, the number of externally buffered reshuffle containers is accordingly increased or decreased.

Example: Table 3a shows one heuristic solution obtained from the construction phase for the instance in Fig. 1. The sequence contains 14 container moves and leads to a service time of 26.83 min (1,610 s). For illustrating the steps involved in one iteration of the constructor, let us consider the partial solution of moves 1 to 6. In the seventh iteration, five container moves are available that can feasibly extend the partial solution. The import container at slot (2,2) can be unloaded to the yard, the reshuffle container at slot (1,3) can be unloaded to the external buffer, an export container can be loaded to slot (3,1), an externally buffered reshuffle container can be reloaded to slot (4,1), or the reshuffle container at slot (1,3) can be internally reshuffled to slot (4,1). These moves are taken up in the candidate list $\mathrm{CL} = ((2,2) \rightarrow \mathrm{Y}, (1,3) \rightarrow \mathrm{B}, \mathrm{Y} \rightarrow (3,1), \mathrm{B} \rightarrow (4,1), (1,3) \rightarrow (4,1))$. Note that all unloading moves enter CL because in this iteration the QC is not restricted to continue unloading a previously started stack. With move 6 being of type VY and processing times as given in Table 2, we calculate δ-values for the moves in CL as $\delta = (d^{\mathrm{VY,VY}} + d^{\mathrm{VY}} = 120,\ d^{\mathrm{VY,VB}} + d^{\mathrm{VB}} + d^{\mathrm{BV}} = 220,\ d^{\mathrm{VY,YV}} + d^{\mathrm{YV}} = 110,\ d^{\mathrm{VY,BV}} + d^{\mathrm{BV}} = 110,\ d^{\mathrm{VY,VV}} + d^{\mathrm{VV}} = 110)$. From Eq. (23) we obtain selection probabilities $\psi = (0.208, 0.113, 0.226, 0.226, 0.226)$. In the solution of Table 3a, the VV-move (last element of CL) has been randomly selected to be the seventh move in the sequence.

4.2 Local search phase

High quality solutions of a CSP instance comprise a high ratio of crane double cycles for loading and unloading containers as well as a high ratio of internally handled

Table 3 Example solutions generated by the constructor and the improvement procedures



reshuffle containers. Hence, heuristic solutions may be improved by shifting container moves to another position and by transforming external reshuffles into internal ones. The corresponding shift-neighborhood and transformation-neighborhood of solutions both rely on so-called insert windows of container moves. The *insert window* of a container move is the subsequence of a solution in which the move can be shifted in front of any contained element without violating the solution's feasibility.

Determination of insert windows: Due to the stacking of containers in a bay, shifting a container move to another position in a given solution is restricted by the transshipment of containers stacked above and below of the considered container. According to Constraints (5)–(7), shifting of a container move is bounded as follows: The unloading of an import or reshuffle container, i.e., moves of type VY and VB, cannot take place before all containers staying ontop have been removed. Hence, the begin of the insert window of such a move is bounded to the removal of the container that stays right ontop of the considered container. The end of the insert window is determined by the move that retrieves the container stacked below of the considered container or by the move that places a new container at the considered slot, whichever appears (earlier) in the given sequence. Similarly, loading moves of type YV and BV cannot start before the sink slot in the bay has been cleared and before all containers below in

the stack have been loaded. For a move of type BV, the begin of the insert window is additionally restricted by the availability of a reshuffle container in the buffer. The end of insert windows of loading moves is determined by the move that stacks a container ontop of the considered container. For moves of type VV, a selection of the above criteria applies jointly, because the reshuffle container must be accessible and the sink slot must be ready for receiving the container.

As an example, consider the third move of the sequence in Table 3a, which is of type VY. The addressed import container cannot be unloaded before the container ontop has been removed (move 2) but it must have been unloaded before an export container is loaded to this slot (move 10). Hence, the move can be feasibly shifted in front of moves 3, 4, 5, ... 10 in the current solution, which is represented by the insert window [3, 10].

Improvement by shifting moves: Shifting container moves reduces empty crane spreader movement whenever single cycling can be replaced by double cycling in a solution. The improvement procedure investigates the shift-neighborhood of a given solution by considering the moves in the sequence one by one. For each move, every alternative position within its insert window is assessed by the change of the empty spreader movement that results from shifting the move to this position. If a reduction of the movement time is found, the container move is shifted to the corresponding position in the sequence, the insert windows of all moves are updated, and the procedure continues with the next move. The procedure is repeated if at least one improvement has been realized within the investigation of the neighborhood of the current solution.

As an example, we improve the heuristic solution shown in Table 3a by shifting the third move of this sequence. The insert window of this move is [3,10] as described above. From shifting this move in front of move 8 in the solution, we obtain the new sequence shown in Table 3b. The shift operation reduces the empty movement of the crane spreader by 20 s, leading to a service time of the bay of 26.5 min (1,590 s).

Improvement by transforming moves: The service time of a bay is furthermore decreased, if external reshuffles in a solution can be transformed into internal reshuffles. Every such transformation realizes a time saving of $d^{\mathrm{VB}} + d^{\mathrm{BV}} - d^{\mathrm{VV}}$ for moving the container plus a reduction of empty crane spreader movement, which's amount depends on the type of preceding and succeeding container moves.

The improvement procedure considers pairs of one VB-move and one BV-move in a given solution. For every of these pairs, it is tested whether the insert windows of the two moves intersect. Intersecting windows indicate that both moves can be feasibly shifted to adjacent positions within the sequence. Hence, since both moves can be performed consecutively, the reshuffle container can be repositioned directly within the bay. This is realized by replacing the two moves representing the external reshuffle by a single internal reshuffle move. The new move is inserted at the first index of the insert window intersection. Then, the improvement procedure investigates the next pair of container moves, and so on. The procedure is repeated if at least one improvement has been realized within the investigation of the neighborhood of the current solution.

As an example, consider moves 2 and 8 in the sequence in Table 3b. The VB-move's insert window is [2,7] and the BV-move's insert window is [7,9]. The two

moves are replaced by an internal reshuffle move that is inserted in front of move 7, see Table 3c. The transformation decreases the service time of the bay by 140 s, leading to an improved solution of 24.17 min (1,450 s).

Repetition and termination: Shifting moves can open up new potential for transforming external reshuffles and vice versa. Hence, the two improvement procedures are executed in a loop until no further improvement is observed in a loop iteration.

4.3 Lower bound

For assessing the quality of heuristic solutions, we determine a lower bound on the total service time of a bay. The lower bound LB is calculated as the sum of a lower bound on the time needed to handle internal reshuffle containers, denoted by LB^{R}, and a lower bound for loading and unloading imports, exports, and external reshuffles, denoted by $\mathrm{LB}^{\mathrm{UL}}$.

The lower bound LB^{R} calculates the processing time for the maximum number of reshuffle containers that can be repositioned internally in an instance. Let $R = card(A^{\mathrm{R}})$ be the number of reshuffle containers in an instance. Let R^{Ext} be the maximum number of reshuffle containers in a stack among all stacks of an instance ($R^{\mathrm{Ext}} = 2$ for the example in Fig. 1). Let us assume these R^{Ext} containers should be internally reshuffled in a solution. This requires that R^{Ext} reshuffle slots in other stacks of the bay are available to receive containers, which necessitates that these stacks have been cleared earlier. From assumption A2, we know that these stacks themselves hold at least R^{Ext} reshuffle containers, which must be removed in order to clear the stacks. Repositioning these removed containers internally requires again R^{Ext} available reshuffle slots in other stacks and so on. Finally, at least R^{Ext} reshuffle containers must be unloaded and externally buffered because no slots for internal repositionings will be available. Hence, at most $R^{\mathrm{Int}} = R - R^{\mathrm{Ext}}$ reshuffle containers can be handled internally in a solution to a CSP instance. LB^{R} sums up the processing time of the R^{Int} internal reshuffles in an instance plus the interconnecting empty spreader movements

$$\mathrm{LB}^{\mathrm{R}} = \begin{cases} R^{\mathrm{Int}} \cdot d^{\mathrm{VV}} + \left(R^{\mathrm{Int}} - 1\right) \cdot d^{\mathrm{VV,VV}}, & \text{if } R^{\mathrm{Int}} > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

For the example in Fig. 1, we observe $R = 4$, $R^{\mathrm{Ext}} = 2$, and $R^{\mathrm{Int}} = 2$. Using the processing times of Table 2, Eq. (24) delivers $\mathrm{LB}^{\mathrm{R}} = 190$ s.

The lower bound $\mathrm{LB}^{\mathrm{UL}}$ bases on the ideal situation where a maximum number of double cycles is realized in a solution for handling imports, exports, and external reshuffles. Let $U = card(A^{\mathrm{I}}) + R^{\mathrm{Ext}}$ be the number of containers that must be unloaded from the vessel, composed by the import containers in an instance and the unavoidable external reshuffles. Let $L = card(D^{\mathrm{E}}) + R^{\mathrm{Ext}}$ be the number of containers to load onto the vessel, composed by the export containers and the external reshuffles. Let $m^{\mathrm{DC}} = 2 \cdot \min\{U, L\}$ and $m^{\mathrm{SC}} = |U - L|$. Consider the case $U = L > 0$, i.e., the number of containers to unload equals the number of containers to load. Here, all loading and unloading moves can be processed under double cycling, leading to $m^{\mathrm{DC}} - 1$ short empty movements, where the crane spreader remains at the

vessel or at the quay. Consider the case $U \neq L$. Here, the crane performs m^{DC} short empty movements for double cycles and $m^{\mathrm{SC}} - 1$ long empty movements for the single cycles where the spreader is moved from the vessel to the quay or vice versa. Eq. (25) sums up the time needed for unloading containers, loading containers, and the interconnecting empty spreader movements for the two cases and the trivial case $U = L = 0$.

$$\mathrm{LB}^{\mathrm{UL}} = U \cdot d^{\mathrm{VY}} + L \cdot d^{\mathrm{YV}} + \begin{cases} (m^{\mathrm{DC}} - 1) \cdot d^{\mathrm{VY,YV}}, & \text{if } U = L > 0 \\ m^{\mathrm{DC}} \cdot d^{\mathrm{VY,YV}} + (m^{\mathrm{SC}} - 1) \cdot d^{\mathrm{VY,VY}}, & \text{if } U \neq L \\ 0, & \text{if } U = L = 0. \end{cases} \tag{25}$$

Note that every short empty spreader movement is assessed by $d^{\mathrm{VY,YV}}$ in Eq. (25) and every long empty movement is assessed by $d^{\mathrm{VY,VY}}$. The evaluation implies that (i) an empty spreader movement between slots in a bay takes as long as the waiting for another container at the transshipment point at the quay, (ii) the time needed for moving the empty crane spreader from the vessel to the quay is the same as for moving it in the opposite direction, and (iii) moves of type VY, YV, VB, and BV have identical processing times. The processing times in Table 2 fulfill these conditions.

For our example, with $R^{\mathrm{Ext}} = 2$, we observe $U = 7$ and $L = 4$. Hence, $m^{\mathrm{DC}} = 8$ containers can be processed using double cycling whereas single cycling is unavoidable for the remaining $m^{\mathrm{SC}} = 3$ containers. With these values, the second case of Eq. (25) returns $\mathrm{LB}^{\mathrm{UL}} = 1{,}220\,\mathrm{s}$.

The total lower bound LB on the time required to serve a bay is calculated as

$$\mathrm{LB} = \mathrm{LB}^{\mathrm{R}} + \mathrm{LB}^{\mathrm{UL}}. \tag{26}$$

For the example instance in Fig. 1, we obtain $\mathrm{LB} = 1{,}410\,\mathrm{s}$ (23.5 min).

5 Computational results

In a computational study, we evaluate the quality of the lower bound presented in Sect. 4.3, the quality of solutions generated by the GRASP heuristic, and the improvement potential that is gained from incorporating internal reshuffles into QC operations.

5.1 Generation of test instances

A large set of test instances is generated for the computational study. The instances differ in the size of the bay, where we distinguish *small*, *medium*, and *large* instances representing bays of size $m = n = 10, 15$, and 20 stacks and tiers, respectively. Moreover we distinguish instances by the ratio of import, export, reshuffle and fixed containers in the arrival and departure configurations. The meaning of the ratios is explained at the example in Fig. 1. With five imports, two exports, four reshuffles, two

fixed containers, and a bay capacity of 12 containers, the ratio of import containers in the arrival configuration is $5/12 \approx 42\%$, the ratio of exports in the departure configuration is $2/12 \approx 17\%$, the ratio of reshuffles is $4/12 \approx 33\%$ in both configurations and the ratio of fixed containers is $2/12 \approx 17\%$ in both configurations. For generating test instances, we use various settings of the ratios to define different workload scenarios of bays. More precisely, we distinguish three workload scenarios, namely *high load*, *low import*, and *low export*, which determine the ratio of import containers in the arrival configuration and the ratio of export containers in the departure configuration. In the high-load scenario, 70% of the slots in the arrival configuration hold an import container and 70% of the slots in the departure configuration hold an export container. In the low-import scenario the ratio of imports is reduced to 40% whereas in the low-export scenario the ratio of exports is reduced to 40%. Finally, the ratio of reshuffle containers (RR) in the instances is varied in the range $[0, 2, \ldots 20\%]$. The ratio of fixed containers is set to 10% for all instances. Slots with no container assigned at all remain empty. In total, 99 combinations of instance size (3 values), workload scenario (3 values), and reshuffle ratio (11 values) are available. For every of these combinations, ten instances are generated by randomly assigning the import, export, reshuffle, and fixed containers to the stacks. In total, 990 benchmark instances are generated. The generation process ensures that at most n containers are placed in a stack. The reshuffle containers in a stack are randomly positioned in-between the import containers in the arrival configuration and on top the export containers in the departure configuration. For the tests, we use the processing times of container moves and empty spreader movement given in Table 2.

5.2 Comparison of lower bounds

In Sect. 4.3, we proposed a lower bound to evaluate heuristic solutions for the CSP. While CPLEX is not able to find integer feasible solutions within short runtimes, it can be used to generate another lower bound. In a first test, we compare these two lower bounds. We identify the better one, which should be used to evaluate heuristic solutions. For this test, each of the 990 generated instances is handed over to CPLEX 11 for a limited runtime of one hour. We then compare the lower bound value delivered by CPLEX, denoted as LB^{cplex}, with the lower bound LB of Eq. (26).

Table 4 shows the results of this test. Every row in this table reports results for 110 test instances of same instance size and workload scenario. The fourth column reports the number of instances where the lower bound LB of Eq. (26) delivers a value that is at least as high as the lower bound LB^{cplex} returned by CPLEX. The fifth column reports the average gap between the two lower bounds that is observed for the 110 instances considered in a row. The gap is calculated for an instance as $\text{gap} = (\text{LB} - \text{LB}^{\text{cplex}})/\text{LB}^{\text{cplex}} \times 100$. The last column reports the maximum gap observed.

The results show that for 961 out of the 990 instances, the lower bound LB is at least as good as the lower bound returned by CPLEX. Actually the values returned by Eq. (26) are noticeably higher than those returned by CPLEX. The average gap is about 4% for instances of small size, increases to about 8% for medium sized instances

Table 4 Comparison of lower bounds

Size	Scenario	# Instances	LB $\geq$ LB$^{\text{cplex}}$	Avg. gap (%)	Max. gap (%)
	High load	110	101	3.41	6.45
Small	Low import	110	100	3.53	8.39
	Low export	110	110	4.06	8.12
	High load	110	110	9.18	12.82
Medium	Low import	110	100	7.52	14.89
	Low export	110	110	8.02	15.30
	High load	110	110	10.86	11.82
Large	Low import	110	110	11.48	14.84
	Low export	110	110	11.48	14.79
		$\sum = 990$	$\sum = 961$	$\oslash = 7.73$	$\oslash = 11.94$

and to about 11% for large sized instances. For the complete instance suite, the values of LB are on average 7.73% above the lower bound values returned by CPLEX with a maximum gap of 15.30%. It can be seen that the lower bound presented in Sect. 4.3 is best suitable for evaluating the quality of heuristic solutions. Compared to CPLEX, Eq. (26) delivers a stronger lower bound at negligible computational effort.

5.3 Solution quality of GRASP

In this test, we investigate the quality of solutions of the GRASP heuristic. GRASP has been implemented in Java. It performs 1,000 iterations per test instance and returns the best found solution. Solutions are assessed on the basis of the relative error RE in percent of the objective function value Z against the lower bound LB of Eq. (26), i.e., $\text{RE} = (Z - \text{LB})/\text{LB} \times 100$. Figure 2 reports results for small, medium, and large sized instances as a function of the reshuffle ratio RR of instances. Each data point represents the average relative error ARE of the 30 instances of corresponding size and corresponding reshuffle ratio RR.

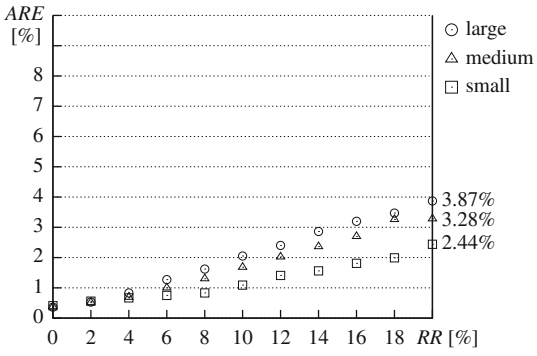


Fig. 2 Solution quality of GRASP

Table 5 Performance measures of the GRASP solutions

Size	Scenario	# Instances	DCR (%)	IRR (%)	ARE (%)	Time (s)
	High load	110	84.52	60.51	1.81	4
Small	Low import	110	72.13	58.46	1.19	4
	Low export	110	74.30	62.15	0.69	3
	High load	110	85.92	65.69	2.20	27
Medium	Low import	110	71.61	63.76	1.79	21
	Low export	110	73.61	68.00	1.25	19
	High load	110	86.50	66.14	2.49	93
Large	Low import	110	72.44	64.95	1.95	79
	Low export	110	73.49	67.84	1.68	68
		$\sum = 990$	$\oslash$ =77.17	$\oslash = 64.17$	$\oslash = 1.67$	$\oslash = 35$

Figure 2 reveals that the ARE of GRASP's solutions grows in the reshuffle ratio RR of instances. Near-optimal solutions are generated for instances with low RRs. For RR = 20%, the ARE grows to 2.44% for small sized instances, 3.28% for medium sized instances, and 3.87% for large sized instances. The results show that GRASP delivers solutions of very good quality for instances of every size. To gain insight into the structure of the solutions generated by GRASP and to evaluate the impact of the load configuration of bays, we report several performance measures for instances of different size and workload scenario in Table 5. The reported double cycling ratio DCR is calculated as the number of loading and unloading moves that are performed in alternating manner in a solution against the total number of loading and unloading moves. The reported internal reshuffling ratio IRR is calculated as the number of internal reshuffle moves in a solution against the total number of reshuffle containers in an instance. The ARE is calculated as described above. Finally, the computation time needed by GRASP on a PC P-IV 2.8 GHz is reported. The measures presented in a row in Table 5 are the average values observed for the 110 test instances of corresponding size and workload scenario.

Table 5 shows that the highest double cycling rates are achieved in the high-load scenario for instances of any size. The balanced numbers of import and export containers in this load scenario enable to load and unload containers in alternating manner most often. On average, about 77% of loading and unloading operations are performed using double cycling. The results for IRR show that, independent of the instance size, the highest internal reshuffling ratio is achieved in the low-export scenario. In such a situation, stacks are comparably early ready for receiving internal reshuffles, because only few export containers have to be loaded. In the low-import scenario, stacks are quickly unloaded but a large number of export containers must be loaded causing external buffering of reshuffles more often. Nevertheless, also in this scenario as well as in the high-load scenario, considerable IRRs are realized in the solutions. As a general observation, on average about 64% of reshuffle containers are internally reshuffled. The high solution quality of the solutions generated by GRASP is proven by the AREs, which stay clearly below 3% for all instance sizes and all workload scenarios. Even for instances of large size and high-load scenario the ARE is only

2.49%. The ARE for the whole test suite is 1.67%. Merely the computation time increases when turning to larger instances. However, even for instances with 20 stacks, 20 tiers, and a high workload, the average computation time is only about one and a half minute per instance. This shows that near-optimal solutions are delivered by the heuristic within runtimes that are reasonably short for practical application.

5.4 Improvement gained from internal reshuffling

In a third test, we evaluate the improvement in the service time of a bay that stems from incorporating internal reshuffling into crane operations planning. As a benchmark, we generate container move sequences where crane double cycling is maximized in a solution but all reshuffles are handled externally. Such a solution is obtained from adopting the stack-based planning approach of Goodchild and Daganzo (2006), see Sect. 2. We first merge the containers of every stack into two precedence-related tasks, one for unloading the stack and one for loading the stack. These tasks are handed over to the rule of Johnson (1954). The rule delivers a processing sequence for the stacks that maximizes double cycling through a largest possible parallelization of unloading tasks and loading tasks. From disaggregating the stack sequence, we obtain a container move sequence, which is a feasible solution to the CSP. In this move sequence a crane performs a maximum number of double cycles but no internal reshuffles. This benchmarking solution is evaluated by Eq. (2). We denote the corresponding bay processing time by Z_{DC}. Let Z_{DC}^{IR} denote the processing time of the solution with double cycling and internal reshuffles as delivered by GRASP for the previous tests. The relative improvement in handling time that results from incorporating internal reshuffles next to double cycling is then calculated as $Imp = (Z_{DC} - Z_{DC}^{IR})/Z_{DC} \times 100$. This improvement has been calculated for each of the 990 test instances. Figure 3 reports the improvement Imp for small, medium, and large sized instances as a function of the reshuffle ratio RR of instances. Each data point represents the average improvement observed for the 30 instances of corresponding size and corresponding reshuffle ratio RR.

The figure illustrates the considerable improvement that is gained from internal reshuffling. As expected, the improvement increases with a growing reshuffle ratio RR. For instances with a low number of reshuffle containers (RR = 2%) the service

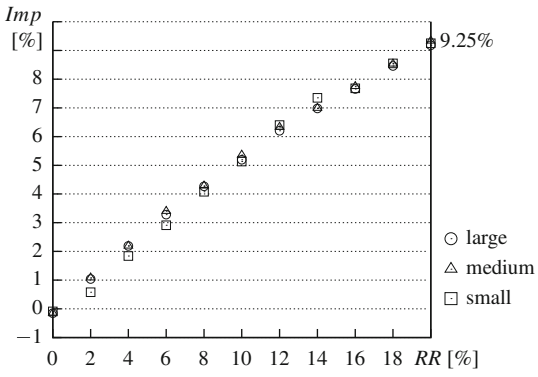


Fig. 3 Improvement gained from internal reshuffling

Table 6 Average improvement observed for different processing times of internal reshuffles

d^{VV} (s)	60	70	80	90	100
Avg. Imp (%)	6.3	5.8	5.4	4.9	4.5

time of the bay is decreased by only 1%. When turning to instances with RR = 20%, the service time of the bay is decreased by more than 9%. The average improvement for the 990 instances in the test suite is 4.9%. Merely for instances without reshuffle containers (RR = 0), the improvement takes a negative value of about −0.1%. Here, Z_{DC} represents the objective value of the optimal solution, because crane double cycling is maximized for an instance without any reshuffle containers. The GRASP heuristic, however, does not find solutions with maximum crane double cycles in every case. The interesting finding in Fig. 3 is that the results are almost identical for instances of small, medium, and large size. The improvement that is gained from internal reshuffling depends on the ratio of reshuffle containers but not on the problem size. The presented planning methodology performs equally well independent of the problem size.

The above results have been determined using the processing times of Table 2 where internal reshuffles take $d^{VV} = 90$ s. Hence, in the previous tests, an internal reshuffle is only slightly faster than loading or unloading of a container, which was assumed to take 100 s. To investigate the impact of the processing time of an internal reshuffle on the reduction of the service times of bays, the computations have been repeated with d^{VV} varied in the range [60, 70, ... 100]. Table 6 shows the average relative improvement Imp for the 990 instances that is observed for these values of d^{VV}. If reshuffles are processed comparably fast ($d^{VV} = 60$) the service times of bays is reduced by 6.3% compared to solutions where no internal reshuffles are performed. Interestingly, even when reshuffles are processed slowly ($d^{VV} = 100$), where internal reshuffles take the same time as unloading and loading moves, an average reduction in the service time of a bay of 4.5% is realized. This improvement stems from saving one container move and its empty spreader movements that comes along with an internal reshuffle. It is concluded that internal reshuffling of containers is inherently advantageous, independent of the actual processing time that is needed for the repositioning.

6 Conclusions

We have provided an approach for planning quay crane operations at a container vessel, where the service of bays is accelerated by crane double cycling and by in-bay repositioning of reshuffle containers. A mathematical optimization model has been formulated for the planning problem. Since the model is too complex for being solved optimal, a GRASP has been designed. Computational tests have been conducted on a large set of test instances. The results show that on average about 77% of loading and unloading operations can be performed in double cycles and that 64% of reshuffle containers in a bay can be reshuffled internally. High quality solutions are generated independent of the problem size, proving that the new approach opens up a considerable potential for accelerating the service of vessels of any size.

Future research could deal with integrated stowage planning and transshipment operations planning. While in nowadays CT practice the departure configurations of

bays are preset for the CSP, they might be derived as a further output of the crane operations planning. In this way, departure configurations may be determined that enable even faster transshipment processes at CTs.

References

Ambrosino D, Sciomachen A (2003) Impact of yard organisation on the master bay planning problem. Marit Econ Logist 5(3):285–300

Ambrosino D, Sciomachen A, Tanfani E (2004) Stowing a containership: the master bay plan problem. Transp Res A 38(2):81–99

Avriel M, Penn M (1993) Exact and approximate solutions of the container ship stowage problem. Comput Ind Eng 25(1–4):271–274

Bendall HB, Stent AF (1996) Hatchcoverless container ships: productivity gains from a new technology. Marit Policy Manage 23(2):187–199

Bierwirth C, Meisel F (2009) A fast heuristic for quay crane scheduling with interference constraints. J Sched 12(4):345–360

Bierwirth C, Meisel F (2010) A survey of berth allocation and quay crane scheduling problems in container terminals. Eur J Oper Res 202(3):615–627

Caserta M, Schwarze S, Voß S (2008) A mathematical formulation for the blocks relocation problem. Working paper, University of Hamburg

Chu C-Y, Huang W-C (2002) Aggregates cranes handling capacity of container terminals: the port of kaohsiung. Marit Policy Manage 29(4):341–350

Daganzo CF (1989) The crane scheduling problem. Transp Res B 23(3):159–175

Goodchild AV, Daganzo CF (2006) Double-cycling strategies for container ships and their effect on ship loading and unloading operations. Transp Sci 40(4):473–483

Goodchild AV, Daganzo CF (2007) Crane double cycling in container ports: planning methods and evaluation. Transp Res B 41(8):875–891

Imai A, Nishimura E, Papadimitriou S, Sasaki K (2002) The containership loading problem. Int J Marit Econ 4(2):126–148

Imai A, Sasaki K, Nishimura E, Papadimitriou S (2006) Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. Eur J Oper Res 171(2):373–389

Johnson SM (1954) Optimal two- and three-stage production schedules with setup times. Nav Res Logist Q 1(1):61–68

Kang J-G, Kim Y-D (2002) Stowage planning in maritime container transportation. J Oper Res Soc 53(4):415–426

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

Kim KH, Kang JS, Ryu KR (2004) A beam search algorithm for the load sequencing of outbound containers in port container terminals. OR Spectr 26(1):93–116

Kim KH, Park YM (2004) A crane scheduling method for port container terminals. Eur J Oper Res 156(3):752–768

Lim A, Rodrigues B, Xu Z (2007) A m-parallel crane scheduling problem with a non-crossing constraint. Nav Res Logist 54(2):115–127

Liu J, Wan Y-W, Wang L (2006) Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. Nav Res Logist 53(1):60–74

Moccia L, Cordeau J-F, Gaudioso M, Laporte G (2006) A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. Nav Res Logist 53(1):45–59

Resende MGC, Ribeiro CC (2003) Greedy randomized adaptive search procedures. In: Glover F, Kochenberger G (eds) Handbook of Metaheuristics. Kluwer, Dordrecht, pp 219–249

Sammarra M, Cordeau J-F, Laporte G, Monaco MF (2007) A tabu search heuristic for the quay crane scheduling problem. J Sched 10(4–5):327–336

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

Zhang H, Kim KH (2009) Maximizing the number of dual-cycle operations of quay cranes in container terminals. Comput Indust Eng 56(3):979–992

Zhu Y, Lim A (2006) Crane scheduling with non-crossing constraint. J Oper Res Soc 57(12):1464–1471