Scheduling cooperative gantry cranes with seaside and landside jobs*

Florian Jaehn[a,b], Dominik Kress[c,*]

[a] *Helmut Schmidt University – University of the Federal Armed Forces Hamburg, Management Science and Operations Research, Holstenhofweg 85, 22043 Hamburg, Germany*

[b] *University of Augsburg, Sustainable Operations and Logistics, Universitätsstr. 16, 86159 Augsburg, Germany*

[c] *University of Siegen, Management Information Science, Kohlbettstr. 15, 57068 Siegen, Germany*

Abstract

We consider the problem of scheduling two identical rail mounted gantry cranes (twin cranes) working within a single storage area (block) at a seaport. The cranes, referred to as seaside crane and landside crane, cannot pass each other. Our focus is on peak times, where the minimization of dwell times of vessels at the berth is typically the major objective of port authorities. We allow the seaside crane to drop inbound containers at intermediate positions where the landside crane takes over and delivers the containers to their target slots. Earlier studies have shown that allowing the cranes to cooperate in this manner is beneficial, at least when there are no containers that are already stored in the block at the beginning of the planning horizon and that have to be delivered to the landside handover point by the landside crane within given time windows. In this paper, we analyze if the positive effect of letting the cranes cooperate persists when these latter jobs are present. This might have a critical impact, because these tasks are performed close to the landside whereas supporting the seaside crane is performed rather close to the seaside. We present complexity results and some general problem insights. Furthermore, we introduce lower bounds and develop heuristic procedures that apply these bounds. The performance of the algorithms is evaluated in computational tests.

Keywords: Crane scheduling, Twin cranes, Port logistics, Container logistics

1. Introduction

The use of standardized loading units that can be handled by different modes of transport has become one of the most time- and cost-effective ways of shipping cargo. Especially *containers* play an important role in modern freight business. The equipment required for handling containers is available almost all over the world. The number of containers handled is especially large at seaports. Large ports handle several ten thousand twenty-foot equivalent units (TEU) per day on average, so that sophisticated logistic processes at those seaports are of great importance for guaranteeing time- and cost-efficient transport.

*Corresponding author

Email addresses: florian.jaehn@hsu-hh.de (Florian Jaehn), dominik.kress@uni-siegen.de (Dominik Kress)

* This is an Accepted Manuscript of an article published by Elsevier in **Discrete Applied Mathematics** on 17 July 2017, available online: https://doi.org/10.1016/j.dam.2017.06.015

The typical movement of an *inbound container*, i.e. a container arriving at the port with a ship, within a seaport can be described as follows. The container is unloaded by a quay crane and it is then passed to a reach stacker, a straddle carrier, an automated guided vehicle, or some similar device. It is then taken to one of dozens of *storage areas*, called blocks, for intermediate storage. A block is usually equipped with *gantry cranes*, which transport the container from a handover point located on the seaside to its designated storage position. When the container is ready to be loaded onto a train or to be picked up by a truck, one of the block's gantry cranes transports the container to a handover point, which is usually located on the other side of the block, i.e. the landside. Then, again, the container is handled by some reach stacker or the like, and it is finally loaded onto a train or a truck. For the sake of completeness, it shall be mentioned that some containers that arrive by ship will be loaded onto another ship, which implies that these containers leave the block on the seaside (*transit containers*). *Outbound containers*, i.e. containers that arrive by train or truck and have to be loaded onto a ship, process the same steps in reverse order.

1.1. Problem Setting and Contribution

Our main focus in this paper lies on scheduling the gantry cranes working within a single block. A common configuration of a block features two identical rail mounted gantry cranes as depicted in Figure 1. This layout is usually referred to as a *twin system* (see, for example, [1]). Based on this layout,

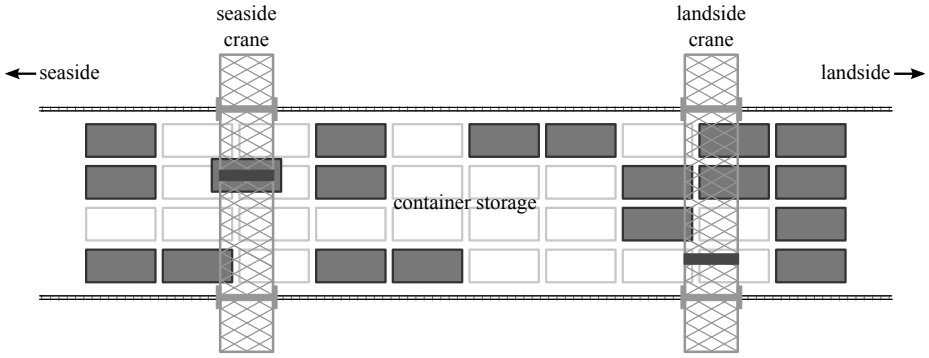


Figure 1: Schematic layout of a twin system

we will refer to the cranes as the *seaside crane* and the *landside crane*, respectively. For a detailed description of the situation at the port of Hamburg, we refer to Speer et al. [2].

Each crane can move a container in three dimensions. First, the *spreader* allows for lifting and dropping the container. Usually, during lifting and dropping, no other movements of the container are possible. Second, each crane can move along *tracks*, allowing for traveling along the long (horizontal) side of the block. Whenever a crane moves in this direction, the spreader can concurrently move along the short (vertical) side of the block. As these two kinds of movements can be performed simultaneously, and as the block is very long but not very wide, spreader movements along the short side are almost never time critical. Therefore, vertical spreader moves will be neglected in our considerations, as it is often done in other approaches for container terminals (e.g. [3, 4]).

The scheduling of twin cranes is usually embedded into higher scheduling tasks. Therefore, we assume that a designated storage position within the block is given for each container, so that capacity

constraints or stacking constraints do not have to be considered. Furthermore, reshuffling of containers is not in the scope of our analysis, especially because the corresponding operations are commonly performed in off-peak times, while we focus on peak times.

The main challenge of a twin-crane configuration is that the cranes cannot pass each other as they share the same tracks, but working areas of both cranes must overlap because containers enter the block on one side and may leave the block on the other side. Thus, when scheduling the container moves to be performed by the cranes, *interference constraints* have to be taken into account.

The workload of the cranes varies significantly over time. It usually reaches its peak when a vessel is to be unloaded at the berth. Accordingly, our considerations will focus on this critical time period. Driven by cost requirements, the minimization of dwell times of vessels at the berth is the major objective of port authorities. Therefore, containers that have to be unloaded from a ship are assigned highest priority and the seaside crane stores containers in the block nonstop. It is a common procedure to let the landside crane perform other duties in the meantime, as, for example, delivering inbound containers to the landside handover point. However, it might be beneficial to let the cranes *cooperate*, i.e. let the landside crane support the seaside crane in storing inbound containers by allowing the seaside crane to drop inbound containers at intermediate positions, where the landside crane takes over and delivers the containers to their target slots. Briskorn et al. [5] raise the question whether this can considerably decrease the time required for storing a set of inbound containers. The authors find: "This [i.e. letting the cranes cooperate] is something for terminal operators to consider, seeing that it is not an uncommon policy in practice to assign only the seaside crane exclusively to stacking containers, while the landside crane is supposed to exclusively handle container transfers to the hinterland. Our study suggests that, at least if no hinterland traffic is currently to be handled, a lot of time can be saved if the landside crane helps out at the seaside during peak times." Even though their study shows that the commonly used procedure in practice bears potential for optimization, they study a scenario which might not be applicable in many cases. The landside crane might still have to deliver some containers that are already stored in the block at the beginning of the planning horizon to the landside handover point, or it may have to store containers that are waiting at the landside handover point in the block. Usually, these container moves should be kept at a minimum, e.g. by assigning corresponding containers to other blocks with less workload. However, even if the landside crane has to perform just a few of these tasks, this might have a critical impact, as these tasks are performed close to the landside, whereas supporting the seaside crane is performed rather close to the seaside, which induces long travel times. In this paper, we therefore want to answer the question of whether it is possible to save significant time during storing inbound containers if the seaside crane is supported by the landside crane, which in turn has to perform some tasks on the landside.

1.2. Related Literature

A general overview on operations research challenges arising at container terminals is given by Steenken et al. [6] and Stahlbock and Voß [7]. When it comes to seaport operations, a major focus lies on berth allocation problems and the scheduling of quay cranes. For these problems, literature reviews are given by Bierwirth and Meisel [8, 9]. However, there are also numerous approaches for optimizing

processes at the storage blocks. A review has recently been given by Carlo et al. [10]. With respect to crane scheduling within the storage area, some very recent approaches can be found in the literature. They are not yet included in [10], so that we outline these papers in this section.

A simple approach for coping with crane interferences is to assign fixed working areas to each crane (see, e.g., [11]). However, if we consider situations with containers entering the storage area on the short side of the block and leaving the block on the other short side, this approach is not applicable because working areas must overlap. In this case, interferences must be prohibited during the scheduling procedure. One of the first approaches to this problem is given by Ng [12], who presents an integer program that includes non-crossing constraints. Dorndorf and Schneider [13] consider a three crane setting, where twin cranes are complemented by a third crane. The latter crane is larger than the twin cranes and can therefore pass them. However, the twin cranes can only pass the larger crane if it is not currently lifting or dropping a container. The authors present a branch and bound method for solving this problem. If inbound containers are exclusively served by the seaside crane, the landside crane may perform other tasks. Ehleiter and Jaehn [3], for example, assume that the landside crane repositions containers within the block, so that they receive more advantageous positions for future operations. However, the landside crane must always give way to the seaside crane.

All of the aforementioned approaches have in common that the cranes do not cooperate in the sense that preemption of inbound container processing is allowed with the landside crane finishing processing the container. Cooperating cranes are analyzed by Briskorn et al. [5], whose setting is very similar to ours. However, there are no containers that are stored in the block at the beginning of the planning horizon and that have to be moved to the landside handover point (landside jobs). The problem is analyzed for two scenarios, one in which the sequence of inbound containers is given and one in which this sequence is part of the optimization. The authors refer to the former problem as the preemptive crane scheduling problem with a given unloading sequence (PCSP-S) and, amongst other heuristics, propose a heuristic algorithm based on the ideas of a bucket brigade. As mentioned above, they conclude that handover operations can significantly reduce the time required for storing all containers in both scenarios.

Note that PCSP-S bears some similarities to scheduling twin robots on a line. These robots have to adhere non-crossing constraints and may allow for handover operations (see, for example, [14, 15]). Finally, note that a test instance generator for crane scheduling tasks at seaports is available online at `www.instances.de/dfg` [16].

1.3. Overview of the Paper

As motivated above, this paper aims to extend PCSP-S to include landside jobs. We refer to the resulting problem as the *preemptive crane scheduling problem with a given unloading sequence and additional landside jobs* (PCSP-SL).

The paper is organized as follows. In Section 2, we will present PCSP-SL in detail and define the notation used throughout the paper. Next, in Section 3, we will analyze the computational complexity of PCSP-SL and present some general problem insights. Section 4 is devoted to solution procedures,

including the introduction of lower bounds, for PCSP-SL. Computational results are presented in Section 5. The paper closes with a conclusion in Section 6.

2. Detailed Problem Description and Notation

Consider a single storage block with two identical rail mounted gantry cranes c, the *seaside crane* ($c = w$) and the *landside crane* ($c = l$), that cannot pass each other. Assume that the *slots* (storage positions) s in the storage block are arranged along a single straight line and are numbered from 0 to $S+1$ with slot 0 being the input/output point (I/O) on the seaside and slot $S+1$ being the I/O point on the landside (see Figure 2). As mentioned in Section 1, assuming the slots to be arranged in this manner is not restrictive for real-world yard settings, as the spreader's vertical movement is typically fast enough to complete its positioning during the crane's horizontal movement. The storage capacity of the slots is assumed to be sufficiently large throughout the planning horizon.

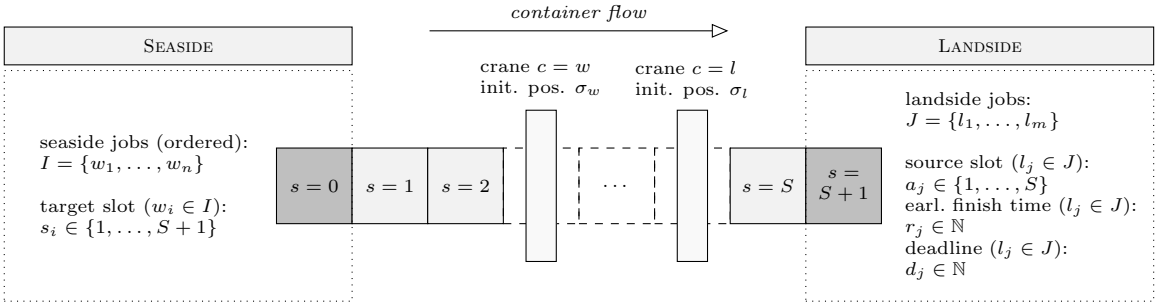


Figure 2: Representation of the storage area and notation used throughout the paper

The time horizon under consideration is divided into a finite number of intervals (time slots) $[t-1, t]$ of equal length, each starting at time instant $t-1$ and ending at time instant t, $t = 1, 2, \ldots$ We will refer to the length of a time interval as a time unit. Starting from a given slot of the storage block, each crane can move to an adjacent slot within one time unit. The number of time units required to lift (also referred to as pick up in the remainder of this paper) or drop a container while not moving between slots is denoted by $p \in \mathbb{N}$.

We restrict ourselves to considering an unidirectional flow of inbound containers from seaside to landside through the storage block (Figure 2), with *seaside jobs* $I = \{w_1, \ldots, w_n\}$ originating at the seaside (slot 0) and *landside jobs* $J = \{l_1, \ldots, l_m\}$ having to be moved to the landside (slot $S+1$). Each seaside job $w_i \in I$ has an associated *target slot* $s_i \in \{1, \ldots, S+1\}$. Similarly, landside jobs $l_j \in J$ originate from a given *source slot* $a_j \in \{1, \ldots, S\}$. We assume that the seaside jobs are ordered with respect to a given *pick-up sequence*, i.e. job $w_i \in I$ needs to be picked up and moved to a slot $s > 0$ by the seaside crane before job $w_j \in I \setminus \{w_i\}$, $j > i$, can be picked up. Additionally, each landside job $l_j \in J$ has an associated earliest time slot $r_j \in \mathbb{N}$ (*earliest finish time*) and a latest time slot $d_j \in \mathbb{N}$ (*deadline*) at the end of which it may be (must be) delivered to the I/O point $S+1$. The earliest finish time r_j and the deadline d_j define the *time window* $[r_j, d_j]$ of landside job $l_j \in J$. Hence, in our problem setting, a crane may start processing a landside job before its earliest finish time at the cost of potentially having to wait at the I/O point, e.g. because the receiving truck has not yet arrived. Preemption of jobs is allowed, meaning that a single container can be lifted and dropped by both cranes before it reaches

its target slot. We assume that each job may be processed by each crane at most once and refer to a preempted job as a *handover container*. Landside jobs may only be processed by the landside crane.

A *crane schedule* is defined by the positions $x_{c,t}$ of both cranes $c \in \{w, l\}$ at all time instants t of the time horizon as well as the cranes' operations in the respective time slots. Concerning these operations, we say that crane c *moves* left (right) in time slot t, if $x_{c,t} = x_{c,t-1} - 1$ ($x_{c,t} = x_{c,t-1} + 1$). The initial locations of the twin cranes are $\sigma_w = x_{w,0} \in \{0, \ldots, S\}$ and $\sigma_l = x_{l,0} \in \{1, \ldots, S+1\}$ with $\sigma_w < \sigma_l$. A crane is *waiting* (also denoted as operation $\circ$) if it is neither moving left (operation $\leftarrow$) or right (operation $\rightarrow$) nor *lifting* (operation $\uparrow$) or *dropping* (operation $\downarrow$) a container. A crane that has lifted (dropped) a container in time slots $t-p+1$ to t and that drops (lifts) this very (the next) container in time slots $t'+1$ to $t'+p$ is referred to as *loaded* (*unloaded*) at time instants $t, \ldots, t'$. Additionally, a crane is referred to as unloaded from time instant 0 until it starts lifting the first container. In all other time instants, the crane is neither considered being loaded nor unloaded. A crane that is loaded (unloaded) at time instants $t-1$ and t, $t > 0$, is said to be loaded (unloaded) in time slot t. Finally, for a given schedule, we denote by $y_{i,t}$, $i \in \{1, \ldots, n\}$ the position (slot) of seaside job w_i in the storage area at time instant t. Analogously, we define $z_{j,t}$, $j \in \{1, \ldots, m\}$, as the position of landside job l_j at time instant t.

The objective is to find a feasible crane schedule that minimizes the *makespan* $C_{\max}$ of seaside container processing (seaside makespan), being defined as the earliest time instant at which all seaside containers have been dropped at their target slots. Landside jobs with deadlines less or equal to $C_{\max}$, i.e. $J^{C_{\max}} := \{l_j \in J | d_j \leq C_{\max}\}$, must be dropped off on time. Additionally, as practical applications do not allow for being mindless of landside jobs being processed after $C_{\max}$, i.e. infeasibility should not occur right after $C_{\max}$, a solution is only considered being feasible if a next landside job l_j, $j \in \arg\min_{l_j \in J \setminus J^{C_{\max}}} d_j$, can still be dropped off on time.

Table 1 accompanies Figure 2 in summarizing the notation used in this paper.

Table 1: Additional notation used throughout the paper

$t = 1, 2, \ldots$	index of time interval (slot) $[t-1, t]$
$p \in \mathbb{N}$	time units required to lift or drop a container
$x_{c,t} \in \begin{cases} \{0, ..., S\} & \text{if } c = w \\ \{1, ..., S+1\} & \text{if } c = l \end{cases}$	position of crane c at time instant t
$y_{i,t} \in \{0, \ldots, S+1\}$	position of job $w_i \in I$ at time instant t
$z_{j,t} \in \{1, \ldots, S+1\}$	position of job $l_j \in J$ at time instant t

3. Problem Insights

We will start our analysis of the problem at hand by analyzing its computational complexity in Section 3.1. Afterwards, in Section 3.2, we will present some general observations based on example instances of PCSP-S with counterintuitive optimal solutions.

3.1. Computational Complexity

If no landside jobs have to be processed by the landside crane, i.e. if $J = \emptyset$, PCSP-SL corresponds to the problem PCSP-S as described by Briskorn et al. [5]. As the authors prove PCSP-S to be NP-hard

in the strong sense, PCSP-SL is NP-hard in the strong sense as well.

Contrary to PCSP-S, where it is easy to find a feasible solution, e.g. by not using handovers at all, this is not the case for PCSP-SL.

Proposition 1 *The problem of deciding whether there exists a feasible solution to an instance of PCSP-SL is NP-hard in the strong sense.*

Proof We will reduce 3-PARTITION, which is defined as follows, to PCSP-SL. Given $3m'+1$ integers $u_1, \ldots, u_{3m'}, B$ with $\sum_{i=1}^{3m'} u_i = mB$ and $\frac{B}{4} < u_i < \frac{B}{2}\ \forall i = 1, \ldots, 3m'$. Is there a partition of set $\{1, \ldots, 3m'\}$ into m' subsets $U_1, \ldots, U'_m$ such that $\sum_{i \in U_j} u_i = B\ \forall j = 1, \ldots, m'$? Note that for every YES-instance of 3-PARTITION, we have $|U_i| = 3\ \forall i = 1, \ldots, m'$. Therefore, 3-PARTITION is still NP-hard if all u_i are assumed to be even (if they are not even, they can simply be multiplied by 2).

Given an instance of 3-PARTITION with all u_i being even, we construct an instance of PCSP-SL, which is described as follows. The number of slots S corresponds to parameter B and each crane is initially located in its respective handover area. The time p required to lift or drop a container is chosen arbitrarily. We artificially create a planning horizon of $m'(B+8p+2)-2p-2+(2p+1)$ time units by assigning $n = \left\lceil \frac{m'(B+8p+2)-2p-2}{2p+2} \right\rceil + 1$ jobs to the seaside crane, each of which is designated for the first slot. Additionally, there are $m = 4m'-1$ landside jobs, which can be split into two classes. Each of the first $3m'$ jobs $j \in \{1, \ldots, 3m'\}$ corresponds to an integer of the 3-PARTITION instance and is initially located at the source slot $a_j = S + 1 - \frac{u_j}{2}$. Its earliest finish time is zero, and it is due at time instant $m'(B+8p+2)-2p-2$. The remaining landside jobs are chosen such that their delivery time is fixed, which is achieved by defining identical earliest finish times and due dates. For each job $j \in \{3m'+1, \ldots, 4m'-1\}$, we set $r_j = d_j = (j-3m')(B+8p+2)$ with source slot S. The resulting instance is summarized as follows.

$S = B$	$p \in \mathbb{N}$	$n = \left\lceil \frac{m'(B+8p+2)-2p-2}{2p+2} \right\rceil + 1$	$m = 4m'-1$
$\sigma_w = 0$	$\sigma_l = S+1$	$s_i = 1\ \forall i = 1, \ldots, n$	
$a_j = S + 1 - \frac{u_j}{2}$	$r_j = 0$	$d_j = m'(B+8p+2)-2p-2$	if $j = 1, \ldots, 3m'$
$a_j = S$	$r_j = d_j = (j-3m')(B+8p+2)$		if $j = 3m'+1, \ldots, 4m'-1$

In this instance, there are no handover containers, because each seaside job's target slot is the first slot. As there is no landside job located in the first slot, the schedule of the seaside crane can easily be determined by successively moving all containers to the first slot. Each corresponding move costs $2p+2$ time units except for the last job, which only takes $2p+1$ time units. Therefore, the makespan is at least $(m'(B+8p+2)-2p-2)+(2p+1)$. This implies that all landside jobs have a deadline less than $C_{\max}$, so that all of them must be processed for obtaining a feasible solution.

Let us first consider the landside jobs $j = 3m'+1, \ldots, 4m'-1$, which can be seen as 'blockers'. As earliest finish time and deadline coincide, their drop-off time in any feasible schedule is given. As the landside crane only processes landside jobs and starts in slot $S+1$, each processing time of a job can be divided into the time for moving to the source slot (always starting from slot $S+1$), the delivery time (from the source slot to $S+1$) and pick-up and drop-off. Thus, each blocker job takes $2+2p$ time units for being processed. Therefore, between any two blockers (or between the start and the first

blocker or between the last blocker and the final deadline of all landside jobs), time periods of length $B + 6p$ emerge. These m' time periods must be used for processing landside jobs $j = 1, \dots 3m'$. In each of these periods, exactly three jobs must be processed. Assume it were four or more jobs. Then pick-up and drop-off would amount to $8p$, and because $u_j > \frac{B}{4}$ for all $j = 1, \dots, 3m'$, travel time would be larger than B, contradicting the length of the period. Now, assume it were less than three jobs. As we have exactly m' time periods, we would not be able to process all landside jobs. Therefore, each period processes exactly three jobs. If we reduce the length $B + 6p$ of the periods by the mandatory time for pick-up and drop-off, we get periods of length B. Thus, there exists a feasible solution, if and only if the corresponding 3-PARTITION instance is a YES-instance. ■

From a practical point of view it has to be noted that, contrary to the theoretical complexity of the problem, finding a feasible solution is usually not very hard. This is rooted in the above mentioned fact that landside jobs appear only sporadically in the planning horizon and that the focus is on the seaside containers.

3.2. General Observations

Recall that we allow each job to be processed at most once by each crane. This assumption is motivated from real world requirements of terminal operators. Even though makespan minimization has highest priority, energy consumption for lifting containers plays an important role. Terminal operators incorporate this aspect by allowing each crane to process each container at most once. From a theoretical point of view, however, one may wonder whether this assumption is actually restrictive. The following Example 1 shows that indeed, we can construct instances of PCSP-S that require a seaside container to be processed more than once by the landside crane in every optimal schedule if it is allowed to do so. In their analysis of PCSP-S, Briskorn et al. [5] do not explicitly address this case, while in their MIP formulation it becomes obvious that they restrict both cranes to handle each container at most once.

Example 1 Let $p = 1$, $\sigma_w = 0$, $\sigma_l = 1$, $S = 8$, $n = 6$, $s_1 = 8$, $s_3 = 4$, $s_2 = s_4 = s_5 = s_6 = 1$, and $m = 0$. First, consider the case of allowing each crane to process each container at most once. The optimal makespan is 25, for example with the seaside crane dropping each container in slot 1 and the landside crane processing w_1 before w_3 without needing to wait in any time slot (see Table 2, where the landside crane's position is indicated by $\hat{l}$). While in the corresponding optimal schedule the seaside crane is finished with dropping containers in slot 1 at time instant 23, the earliest possible time instant for the landside crane to finish processing w_1 and w_3 is 25. If the landside crane were to process w_3 before w_1, the makespan would be at least 29; if the seaside crane were to drop w_1 or w_3 in slot 2 or even further to the right to help the landside crane, it would be processing containers for at least 25 time units. Now, drop the assumption of allowing each container to be processed at most once by each crane. Then we can determine a schedule with an optimal makespan of 23 by making the landside crane process w_1 twice (see Table 2, where the landside crane's position is indicated by l). □

We note that there exist examples for PCSP-SL, in which the landside crane even processes a seaside container arbitrarily often in every optimal solution (not shown here).

Table 2: Optimal schedules with w_1 being processed once ($\hat{l}$) or twice (l)

t	crane pos. (w and $\hat{l}$) at inst. t										crane pos. (w and l) at inst. t										crane oper. in slot t		
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	crane w	crane $\hat{l}$	crane l
0	w	$\hat{l}$	-	-	-	-	-	-	-	-	w	l	-	-	-	-	-	-	-	-	$\circ$	$\circ$	$\circ$
1	w	$\hat{l}$	-	-	-	-	-	-	-	-	w	l	-	-	-	-	-	-	-	-	$\uparrow w_1$	$\circ$	$\circ$
2	-	w	$\hat{l}$	-	-	-	-	-	-	-	-	w	l	-	-	-	-	-	-	-	$\rightarrow$	$\rightarrow$	$\rightarrow$
3	-	w	$\hat{l}$	-	-	-	-	-	-	-	-	w	l	-	-	-	-	-	-	-	$\downarrow w_1$	$\circ$	$\circ$
4	w	$\hat{l}$	-	-	-	-	-	-	-	-	w	l	-	-	-	-	-	-	-	-	$\leftarrow$	$\leftarrow$	$\leftarrow$
5	w	$\hat{l}$	-	-	-	-	-	-	-	-	w	l	-	-	-	-	-	-	-	-	$\uparrow w_2$	$\uparrow w_1$	$\uparrow w_1$
6	-	w	$\hat{l}$	-	-	-	-	-	-	-	-	w	l	-	-	-	-	-	-	-	$\rightarrow$	$\rightarrow$	$\rightarrow$
7	-	w	-	$\hat{l}$	-	-	-	-	-	-	-	w	-	l	-	-	-	-	-	-	$\downarrow w_2$	$\rightarrow$	$\rightarrow$
8	w	-	-	-	$\hat{l}$	-	-	-	-	-	w	-	-	-	l	-	-	-	-	-	$\leftarrow$	$\rightarrow$	$\rightarrow$
9	w	-	-	-	-	$\hat{l}$	-	-	-	-	w	-	-	-	l	-	-	-	-	-	$\uparrow w_3$	$\rightarrow$	$\downarrow w_1$
10	-	w	-	-	-	-	$\hat{l}$	-	-	-	-	w	-	l	-	-	-	-	-	-	$\rightarrow$	$\rightarrow$	$\leftarrow$
11	-	w	-	-	-	-	-	$\hat{l}$	-	-	-	w	l	-	-	-	-	-	-	-	$\downarrow w_3$	$\rightarrow$	$\leftarrow$
12	w	-	-	-	-	-	-	-	$\hat{l}$	-	w	l	-	-	-	-	-	-	-	-	$\leftarrow$	$\rightarrow$	$\leftarrow$
13	w	-	-	-	-	-	-	-	$\hat{l}$	-	w	l	-	-	-	-	-	-	-	-	$\uparrow w_4$	$\downarrow w_1$	$\uparrow w_3$
14	-	w	-	-	-	-	-	$\hat{l}$	-	-	-	w	l	-	-	-	-	-	-	-	$\rightarrow$	$\leftarrow$	$\rightarrow$
15	-	w	-	-	-	-	$\hat{l}$	-	-	-	-	w	-	l	-	-	-	-	-	-	$\downarrow w_4$	$\leftarrow$	$\rightarrow$
16	w	-	-	-	-	$\hat{l}$	-	-	-	-	w	-	-	-	l	-	-	-	-	-	$\leftarrow$	$\leftarrow$	$\rightarrow$
17	w	-	-	-	$\hat{l}$	-	-	-	-	-	w	-	-	-	l	-	-	-	-	-	$\uparrow w_5$	$\leftarrow$	$\downarrow w_3$
18	-	w	-	$\hat{l}$	-	-	-	-	-	-	-	w	-	-	l	-	-	-	-	-	$\rightarrow$	$\leftarrow$	$\uparrow w_1$
19	-	w	$\hat{l}$	-	-	-	-	-	-	-	-	w	-	-	-	l	-	-	-	-	$\downarrow w_5$	$\leftarrow$	$\rightarrow$
20	w	$\hat{l}$	-	-	-	-	-	-	-	-	w	-	-	-	-	-	l	-	-	-	$\leftarrow$	$\leftarrow$	$\rightarrow$
21	w	$\hat{l}$	-	-	-	-	-	-	-	-	w	-	-	-	-	-	-	l	-	-	$\uparrow w_6$	$\uparrow w_3$	$\rightarrow$
22	-	w	$\hat{l}$	-	-	-	-	-	-	-	-	w	-	-	-	-	-	-	l	-	$\rightarrow$	$\rightarrow$	$\rightarrow$
23	-	w	-	$\hat{l}$	-	-	-	-	-	-	-	w	-	-	-	-	-	-	l	-	$\downarrow w_6$	$\rightarrow$	$\downarrow w_1$
24	-	w	-	-	$\hat{l}$	-	-	-	-	-											$(\circ)$	$\rightarrow$	
25	-	w	-	-	$\hat{l}$	-	-	-	-	-											$(\circ)$	$\downarrow w_3$	

Next, one may wonder if there exist instances of PCSP-S with the landside crane moving left while being loaded with a seaside container in every optimal solution. When presenting a MIP for the variant of problem PCSP-S in which the sequence of containers to be handled by the seaside crane is not given but part of the decision, Briskorn et al. [5] state that "it makes no sense for [the landside crane] to carry a container past its destination slot". However, as they later apply this model with an additional constraint on the sequence for solving PCSP-S, they might not have solved their instances to optimality as the following Example 2 shows.

Example 2 Let $p = 6$, $\sigma_w = 0$, $\sigma_l = 33$, $S = 32$, $n = 6$, $s_1 = 28$, $s_2 = s_3 = 6$, $s_4 = s_5 = s_6 = 1$, $m = 0$. Consider a feasible schedule with makespan 97 as presented in Table 3. Note that the landside crane moves left while being loaded in time slot 51. We will now show that this schedule is the only optimal schedule.

Table 3: Optimal schedule with landside crane moving left while being loaded

t	crane pos. at inst. t													crane oper. in slot t	
	0	1	2	3	4	5	6	7	8	...	28	...	33	crane w	crane l
00	w	-	-	-	-	-	-	-	-	-	-	-	l	$\circ$	$\circ$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	$(\uparrow w_1, \rightarrow)$	$(\leftarrow)$
09	-	-	-	w	-	-	-	-	-	l	-	-	-	$\rightarrow$	$\leftarrow$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	$(\downarrow w_1)$	$(\leftarrow)$
15	-	-	-	w	-	-	-	-	-	l	-	-	-	$\downarrow w_1$	$\leftarrow$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	$(\leftarrow)$	$(\leftarrow)$

18	w	-	-	-	-	-	-	-	-	l	-	-	-	$\leftarrow$	$\leftarrow$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	$(\uparrow w_2)$	$(\leftarrow)$
25	-	w	-	-	-	-	-	-	l	-	-	-	-	$\rightarrow$	$\leftarrow$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	$(\downarrow w_2)$	$(\leftarrow)$
31	-	w	l	-	-	-	-	-	-	-	-	-	-	$\downarrow w_2$	$\leftarrow$
32	w	l	-	-	-	-	-	-	-	-	-	-	-	$\leftarrow$	$\leftarrow$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	$(\uparrow w_3)$	$(\uparrow w_2)$
39	-	w	l	-	-	-	-	-	-	-	-	-	-	$\rightarrow$	$\rightarrow$
40	-	-	w	l	-	-	-	-	-	-	-	-	-	$\rightarrow$	$\rightarrow$
41	-	-	-	w	l	-	-	-	-	-	-	-	-	$\rightarrow$	$\rightarrow$
42	-	-	-	-	w	l	-	-	-	-	-	-	-	$\rightarrow$	$\rightarrow$
43	-	-	-	-	-	w	l	-	-	-	-	-	-	$\rightarrow$	$\rightarrow$
44	-	-	-	-	-	-	w	l	-	-	-	-	-	$\rightarrow$	$\rightarrow$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	$(\downarrow w_3)$	$(\circ)$
50	-	-	-	-	-	-	w	l	-	-	-	-	-	$\downarrow w_3$	$\circ$
51	-	-	-	-	-	w	l	-	-	-	-	-	-	$\leftarrow$	$\leftarrow$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	$(\leftarrow, \uparrow w_4)$	$(\downarrow w_2)$
57	w	-	-	-	-	-	l	-	-	-	-	-	-	$\uparrow w_4$	$\downarrow w_2$
58	w	-	-	-	-	l	-	-	-	-	-	-	-	$\uparrow w_4$	$\leftarrow$
59	w	-	-	-	l	-	-	-	-	-	-	-	-	$\uparrow w_4$	$\leftarrow$
60	w	-	-	l	-	-	-	-	-	-	-	-	-	$\uparrow w_4$	$\leftarrow$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	$(\uparrow w_4, \rightarrow, \downarrow w_4)$	$(\uparrow w_1)$
66	-	w	-	l	-	-	-	-	-	-	-	-	-	$\downarrow w_4$	$\uparrow w_1$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	$(\downarrow w_4, \leftarrow, \uparrow w_5, \rightarrow, \downarrow w_5, \leftarrow, \uparrow w_6)$	$(\rightarrow)$
91	-	w	-	-	-	-	-	-	-	-	l	-	-	$\rightarrow$	$\rightarrow$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	$(\downarrow w_6)$	$(\downarrow w_1)$
97	-	w	-	-	-	-	-	-	-	-	l	-	-	$\downarrow w_6$	$\downarrow w_1$

First, observe that w_1 is necessarily handed over in any optimal solution as otherwise the seaside crane would be processing w_1 and return to slot 0 for $2p + 2s_1 = 68$ time units and would then have to at least lift and drop every other container, which takes $2p(n-1) = 60$ time units. Hence, the makespan would be larger than 97. Similarly, note that the seaside crane cannot process w_2 and w_3 without handing at least one of them over. This is obvious when noting that in the given schedule the seaside crane never waits for $t > 0$. Now, assume that the landside crane drops both w_2 and w_3 in their target slots. Then it will be processing containers for at least $[(\sigma_l - s_2 + 1) + 2p + 1] + [2p + 2] + [s_3 - 1 + 2p + s_1 - 1] = 99 > 97$ time units. Here, the first two terms correspond to picking up and processing w_2 and w_3 from slot $s_2 - 1 = 5$. The third term refers to picking up and processing w_1 from slot 1. Note that if we assume this latter slot to be located further to the right, the seaside crane's processing time would be larger than 97. Hence, we can conclude that exactly one of the containers w_2 and w_3 will be handed over. Now, assume that w_3 is a handover container. Then the earliest possible time instant for the landside crane to drop w_3 in s_3 is $[2(2p + 2)] + [2p + 2s_2] + [2p + s_3 - 1] = 69$, because the seaside crane will necessarily have to process w_2 alone (second term). Additionally, it will have to process w_1, which induces a makespan lager than 97. Summing up, w_1 and w_2 are handover containers in every optimal

schedule, while the other containers are handled by the seaside crane alone. Recall that in the schedule in Table 3 the seaside crane never waits for $t > 0$. Additionally, it drops its last container at time instant 97, which is equal to the makespan of the schedule. Hence, we can conclude that the sum of the handover slots of containers w_1 and w_2 is at most 4 in any optimal schedule. Moreover, it is easy to conclude from this latter fact that the landside crane will process w_2 before w_1. Finally, note that this results in the seaside crane having to wait in at least two time slots $t_1, t_2 > 0$, if w_2 were set down before w_3 for any pair of potential handover slots for containers w_1 and w_2. Hence, the schedule in Table 3 necessarily represents the only optimal solution except for similar solutions in which the landside crane makes unnecessary moves to the right instead of waiting in slot $s_2 + 1$. □

4. Solution Procedures

We will propose heuristic procedures for solving PCSP-SL. The two heuristics to be presented are extensions of the bucket brigade principle, which is shown to perform well for PCSP-S by Briskorn et al. [5]. Contrary to their principle, our heuristics make use of some lower bounds, so that we start with presenting these bounds in Section 4.1, before presenting details on the heuristics in Section 4.2.

4.1. Lower Bounds

This section introduces a lower bound for PCSP-SL, which obviously holds for PCSP-S as well. For ease of description, assume that $\sigma_w = 0$. Furthermore, consider an optimal solution with objective function value t^* and let LB be a lower bound on t^* (e.g. $LB = 0$), which will iteratively be improved.

We start our deliberations with a lower bound $LB^{(dist)}$ on the total distance the cranes must cover in any optimal solution until time instant t^*. Obviously, all seaside containers must be moved, so that the cranes move at least $\sum_{i=1}^{n} s_i$ time units. Landside containers with a deadline less or equal to LB must be moved as well, which takes $\sum_{\{j\in\{1,\ldots,m\}|d_j\leq LB\}} (S + 1 - a_j)$ time units. Additionally, most container moves force the cranes to move left unloaded. If we assumed that the cranes have to end up in their initial positions, then the distance of all container moves would have to be covered twice. Because the cranes cannot end up arbitrarily far from their initial positions, this second coverage of container moves has to be performed at least partially. If we exclude unnecessary unloaded moves, the rightmost slot that the seaside crane may end up in is s_n, whereas the landside crane might end up in slot $S + 1$, which is $S + 1 - \sigma_l$ slots away from its initial position. Therefore, unloaded crane moves sum up to $\left(\sum_{i=1}^{n} s_i + \sum_{\{j\in\{1,\ldots,m\}|d_j\leq LB\}} (S + 1 - a_j)\right) - s_n - (S + 1 - \sigma_l)$. In total, we obtain

$$LB^{(dist)} := 2\left(\sum_{i=1}^{n} s_i + \sum_{\{j\in\{1,\ldots,m\}|d_j\leq LB\}} (S + 1 - a_j)\right) - s_n - (S + 1 - \sigma_l).$$

Obviously, $\lceil LB^{(dist)}/2 \rceil$ is a lower bound on t^* and it could be used as LB to iteratively update $LB^{(dist)}$ and LB by means of updating the landside containers to be processed before LB. However, the resulting bound might be very weak, especially for large values of p. Let us therefore approximate the number of lifts and drops that appear no later than t^* in any optimal solution. Obviously, we have at

least $2(n + |\{j \in \{1, \ldots, m\} | d_j \leq LB\}|)$ regular lifts and drops plus a lift and a drop for each handover container. For a given number h of handover containers, we therefore obtain the lower bound

$$LB(h) := \left\lceil \frac{LB^{(dist)} + 2p(n + |\{j \in \{1, \ldots, m\} | d_j \leq LB\}| + h)}{2} \right\rceil.$$

Note that, for a given LB, $LB(h+1) = LB(h) + p$. Hence, $LB(h)$ is increasing in h and $LB(0)$ can be used as a general lower bound. However, it might be possible to show that the seaside crane's workload in a solution without handover containers (more generally: with h handover containers) is relatively large and that an additional handover container improves the solution. For a given h, we can easily bound the processing time of the seaside crane, $LB^{(sea)}$, from below by assuming that there are no crane interferences and that the h containers with the rightmost target slots are handed over in slot 1. That means that the seaside crane performs $2n$ lifts and drops, moves h containers to slot one, and moves $n - h$ containers to their target slot. Formally, we get

$$LB^{(sea)}(h) := 2np + 2\left(h + \min_{I' \subseteq I, |I'| = n-h} \sum_{i \in I'} s_i\right) - s_n.$$

Note that $LB^{(sea)}(h)$ can easily be computed iteratively. If $LB^{(sea)}(h) > LB(h)$, then we know that $LB(h)$ is loose. Furthermore, if $LB^{(sea)}(h) > LB(h+1)$, then $LB(h+1)$ is a general lower bound. We may therefore iteratively increase h until either $LB^{(sea)}(h) \leq LB(h)$ or $LB^{(sea)}(h) \leq LB(h+1)$. In the former case, $LB(h)$ is a lower bound. If solely the latter case holds true, $LB^{(sea)}(h)$ is a lower bound. Additionally, if $s_n \neq S + 1$ in an instance of PCSP-SL, then there exists an optimal solution with at most $n - 1$ handover containers. We summarize these ideas in the following algorithm.

Algorithm 1:

0. Initialization: Set $LB := 0$, $LB^* := 0$, and $h := 0$.

1. Bound on h: If $h < n - 1$ and $LB(h+1) < LB^{(sea)}(h)$, then $h := h + 1$ and go to Step 1.
If $LB(h) < LB^{(sea)}(h)$, then $LB := LB^{(sea)}(h)$, else $LB := LB(h)$

2. Iteration: If $LB^* \geq LB$, then stop. Else, set $LB^* := LB$, $h := 0$ and go to Step 1.

The best bound is stored in LB^*. If the current bound LB is better than LB^* in Step 2, the procedure needs to be repeated as more landside containers might be considered when calculating $LB(h)$.

The calculation of the lower bound can be adapted such that it can be applied to partial solutions, in which the movements of the cranes are fixed until a certain time instant. In such a case, the bound can slightly be tightened, e.g. by considering initial movements of the cranes to their first pick-up slot.

4.2. Heuristic Procedures

The bucket brigade algorithm for the PCSP-S is based on some straightforward rules: If the seaside crane is unloaded, it moves to slot 0 and picks up the next container. It then either delivers the container

to its target slot or, if the landside crane obstructs the passage, drops the container for handover. The landside crane always moves left if unloaded until it meets the seaside crane. In this case, a handover container is generated, which is then moved to its corresponding target slot by the landside crane. Details are presented in Briskorn et al. [5]. We will refer to this set of rules with the term *bucket brigade*.

As mentioned before, this simple procedure works very effectively for PCSP-S, so that we extend it for PCSP-SL. This implies that we need to find rules for the landside crane serving the landside containers. The set of landside containers that must still be served within the planning horizon, J^{LB^*}, is dynamically updated in the course of our algorithms, using the lower bound described above. Additionally, the landside crane cannot always process handover containers immediately after they have been dropped off by the seaside crane. These containers and their current positions are stored in the set H.

The first extension of the bucket brigade principle (Algorithm 2) gives priority to landside containers once they are available. More specifically, once the landside crane has dropped off a container, the algorithm checks whether there is a landside container that can be served by the landside crane without having to wait in slot $S+1$. If so, the landside crane serves this container. Of course, due to not waiting for landside containers, this strategy may result in an infeasible solution. In this case, the algorithm uses backtracking, so that the landside crane interrupts the bucket brigade earlier and therefore gives even higher priority to landside containers.

Algorithm 2:

0. Initialization: Determine LB^* using Algorithm 1, set $J^{LB^*} := \{l_j \in J \mid d_j \leq LB^*\}$ (set of landside containers that must still be served by the landside crane) and (in case of its existence) insert an additional element l_j with $j \in \arg\min_{l_j \in J \setminus J^{LB^*}} d_j$ into J^{LB^*}. Furthermore, set $t = 0$ and $H := \emptyset$ (set of handover containers waiting for landside crane handling).

1. Landside crane decision: If $J^{next} := \{l_j \in J^{LB^*} | t + |x_{l,t} - a_j| + 2p + S + 1 - a_j \geq r_j\}$ is not empty (a landside container can be served by the landside crane without waiting in slot $S+1$), then go to Step 2 and pass over $l_{j^{next}}$ with $j^{next} \in \arg\min_{l_j \in J^{next}} d_j$ as next landside container. Else, go to Step 3.

2. Landside container: Let t^{wait} be the time that the landside crane must wait when directly processing $l_{j^{next}}$ due to the seaside crane obstructing the passage. Determine $t^{j^{next}} := t + |x_{l,t} - a_{j^{next}}| + 2p + S + 1 - a_{j^{next}} + t^{wait}$. If $t^{j^{next}} > d_{j^{next}}$ (the time window cannot be met), go to Step 5. Else, the landside crane serves $l_{j^{next}}$ until drop-off and the seaside crane continues bucket brigade. Update t, H, LB^*, and J^{LB^*}. Go to Step 1.

3. Handover container: If $H = \emptyset$, then go to Step 4. Else, the landside crane serves a seaside container $w_{\hat{i}}$ until drop-off, where $\hat{i} \in \arg\min_{w_i \in H_r} y_{i,t}$ if $H_r := \{w_i | w_i \in H, y_{i,t} \geq x_{l,t}\} \neq \emptyset$, or $\hat{i} \in \arg\max_{w_i \in H} y_{i,t}$ if $H_r = \emptyset$. The seaside crane continues bucket brigade. Update t, H, LB^*, and J^{LB^*}. Go to Step 1.

4. Bucket brigade: If all seaside containers have been served by the seaside crane, then stop. Else,

continue bucket brigade with both cranes until the landside crane drops off the next container or until the seaside crane drops off w_n in its target slot while the landside crane is unloaded. Update t, H, LB^*, and J^{LB^*}. Go to Step 1.

5. Backtracking: Turn back to the latest time instant $t' < t$ in which the landside crane has just dropped off a container (or, if no such instant exists, to the beginning of the planning horizon) and is about to serve a seaside container. If no such instant exists, then stop; No feasible solution is found. Else, set $t = t'$, update H, LB^*, and J^{LB^*}, go to Step 2 and pass over $l_{j^{next}}$ with $j^{next} \in \arg\min_{l_j \in J^{LB^*}} d_j$ as next landside container.

In Step 2 of Algorithm 2, t^{wait} is non-zero if $x_{w,t} \geq a_{j^{next}}$ and the seaside crane is in the process of dropping a container with the remaining time needed until final drop-off being no smaller than $x_{l,t} - x_{w,t}$. An analogous situation may arise when the seaside crane is loaded and the cranes are located close to each other. Similarly, at the end of Step 2, the remaining time to process $l_{j^{next}}$ such that it does not miss its deadline might be so small that there is no time for the landside crane to wait for the seaside crane to drop a container within the regular bucket brigade mode. In this case, we slightly modify the bucket brigade rules to let the seaside crane drop its container further to the left or wait in slot 0, so that it does not block the landside crane's passage.

Algorithm 2 is likely to result in a feasible solution because landside containers have high priority. This, however, might result in solutions in which the landside crane often moves unloaded between landside and seaside. We therefore propose a second variant of the bucket brigade principle, which bundles landside containers for sequential processing (Algorithm 3). To do so, we first focus on a landside container with earliest deadline among the containers not yet served. Under the assumption that this container will be dropped off right at its deadline, we add landside containers to the bundle, whose earliest finish times allow the landside crane to process the bundle without waiting in slot $S+1$ and without processing seaside containers in between. We then define a time window in which the containers of the bundle can be processed without waiting in slot $S+1$. The bundle and its time window are potentially updated, whenever the landside crane has dropped off a container. The algorithm then checks whether the bundle can be processed within the time window. If this is not the case, it either continues bucket brigade or uses backtracking.

Algorithm 3:

0. Initialization: Determine LB^* using Algorithm 1, set $J^{LB^*} := \{l_j \in J \mid d_j \leq LB^*\}$ and (in case of its existence) insert an additional element l_j with $j \in \arg\min_{l_j \in J \setminus J^{LB^*}} d_j$ into J^{LB^*}. Furthermore, set $t = 0$ and $H := \emptyset$.

1. Bundling: Let $j^{\min} \in \arg\min_{l_j \in J^{LB^*}} d_j$. If no such container exists, go to Step 4.

1.1 Initialization: Let $J^{bdl} := \{l_{j^{\min}}\}$, $t^{\max} := d_{j^{\min}} - 2p - (S + 1 - a_{j^{\min}})$ (latest possible time for starting pick-up of $l_{j^{\min}}$ for on time drop-off), and $t' := d_{j^{\min}}$.

1.2 Iteration: Determine a landside container $l_{j'}$ with earliest finish time not larger than t' or with the smallest earliest finish time as follows. If $J' := \{l_j \in J^{LB^*} \setminus J^{bdl} | r_j \leq t'\} \neq \emptyset$ then

$j' \in \arg\min_{l_j \in J'} d_j$, else $j' \in \arg\min_{l_j \in J^{LB^*} \setminus J^{bdl}} r_j$. Update $t' := d_{j^{\min}} + \sum_{l_j \in J^{bdl} \setminus \{l_{j^{\min}}\}} (2p + 2(S + 1 - a_j)) + 2p + 2(S + 1 - a_{j'})$. If the landside crane starts processing J^{bdl} at $t^{\max}$ and can then process $l_{j'}$ without waiting, i.e. $t' \geq r_{j'}$, then set $J^{bdl} := J^{bdl} \cup \{l_{j'}\}$ and go to Step 1.2. Else, go to Step 1.3.

1.3 Time window: Determine the earliest time $t^{\min}$ for starting to lift $l_{j^{\min}}$, so that all containers in J^{bdl} can be processed without waiting. If starting to process J^{bdl} at $t^{\max}$ results in a deadline violation, adjust $t^{\max}$ accordingly. If $t^{\max} < t^{\min}$ then stop; No feasible solution is found.

2. Landside crane decision: Let t^{wait} be the time that the landside crane must wait when directly processing $l_{j^{min}}$ due to the seaside crane obstructing the passage. If $t^{\min} > t + |x_{l,t} - a_{j^{\min}}| + t^{wait}$, then go to Step 4. Else, if $t^{\min} \leq t + |x_{l,t} - a_{j^{\min}}| + t^{wait} \leq t^{\max}$, go to Step 3. Else, go to Step 6.

3. Landside container: The landside crane serves J^{bdl} until final drop-off. The seaside crane continues bucket brigade. Set $J^{bdl} := \emptyset$. Update t, H, LB^*, and J^{LB^*} and go to Step 1.

4. Handover container: If $H = \emptyset$, then go to Step 5. Else, the landside crane serves a seaside container $w_{\hat{i}}$ until drop-off, where $\hat{i} \in \arg\min_{w_i \in H_r} y_{i,t}$ if $H_r := \{w_i | w_i \in H, y_{i,t} \geq x_{l,t}\} \neq \emptyset$, or $\hat{i} \in \arg\max_{w_i \in H} y_{i,t}$ if $H_r = \emptyset$. The seaside crane continues bucket brigade. Update t, H, LB^*, and J^{LB^*}. If J^{LB^*} has changed, set $J^{bdl} = \emptyset$ and go to Step 1. Else, go to Step 2.

5. Bucket brigade: If all seaside containers have been served by the seaside crane, then stop. Else, continue bucket brigade with both cranes until the landside crane drops off the next container or until the seaside crane drops off w_n in its target slot while the landside crane is unloaded. Update t, H, LB^*, and J^{LB^*}. If J^{LB^*} has changed, set $J^{bdl} = \emptyset$ and go to Step 1. Else, go to Step 2.

6. Backtracking: Turn back to the latest time instant $t' < t$ in which the landside crane has just dropped off a container (or, if no such instant exists, to the beginning of the planning horizon) and is about to serve a seaside container. If no such instant exists, then stop; No feasible solution is found. Else, set $t = t'$, update H, LB^*, J^{LB^*}, J^{bdl}, $t^{\min}$, and $t^{\max}$. Update $l_{j^{min}}$ and determine t^{wait} as above. If $t + |x_{l,t} - a_{j^{\min}}| + t^{wait} > t^{\max}$, then go to Step 6. Else, go to Step 3.

Note that in Step 2 of Algorithm 3, t^{wait} is defined as in Algorithm 2. Additionally, we apply the modified bucket brigade rules mentioned above in Step 3 of Algorithm 3, so that the seaside crane does not block the landside crane's passage once it has started processing a bundle of landside containers. Moreover, note that if backtracking is triggered in Step 2 of Algorithm 3, waiting in slot $S + 1$ during serving the corresponding bundle in Step 3 might be necessary.

Finally, note that in Algorithms 2 and 3 the landside crane might finish processing landside containers after all seaside containers have reached their target slots. This has to be taken into account when determining the objective function value of the solution returned by the algorithms.

5. Computational Study

Based on a computational study that we conducted on realistic data, we will now empirically evaluate the effect caused by allowing the gantry cranes to cooperate when considering the presence of both, seaside and landside jobs. We will first answer some auxiliary questions, dealing with the appropriateness of the proposed heuristics:

Q1: Which of the two proposed heuristics performs better in terms of solution quality and the percentage of instances for which a feasible solution is found?

Q2: Are the runtimes of the heuristics in ranges that allow their use in real-life scenarios?

Q3: How big is the gap between the (initial) lower bound and the solutions determined by the heuristics?

Answers to the first two questions, Q1 and Q2, indicate which algorithm is suited for practical applications. An answer to Q3 provides insights into the informative value of the conclusions to be drawn on our main research question Q4:

Q4: Is it possible to save a significant amount of time by allowing the gantry cranes to cooperate in real-world problem settings?

If, for example, the gap is fairly large and the heuristics demonstrate that a large amount of time can be saved by allowing the cranes to cooperate when compared to a non-cooperative setting, the research question can positively be answered. On the other hand, a large gap and no time savings do not allow for any positive or negative answer to the research question.

In order to obtain more managerial insights, the following question will also be analyzed:

Q5: What is the critical ratio of landside jobs to seaside jobs, above which no further improvement of the seaside makespan is elicited by allowing the cranes to cooperate?

The answer to this question allows for determining scenarios in which cooperation usually is beneficial.

We implemented the algorithms in C++ (Microsoft Visual Studio 2010). The computational tests were performed on a PC with an Intel® Core™ i7 CPU running at 3.4 GHz and 16 GB of memory, running Windows 8.1 64bit.

5.1. Instance Generation

We used the test data generator introduced by Briskorn et al. [16] to generate a testbed of instances. These instances are publicly available at `www.instances.de/dfg` using the project name *PCSP-SL_Data*. They are based on real-world data from the Europe Container Terminals (ECT) at the port of Rotterdam, where a block is typically about 40 TEU long, which corresponds to $S = 40$ (refer to [5, 17] for details). Our assumptions are in analogy to Briskorn et al. [5]. This includes the assumption of all data being available at the beginning of the planning horizon. When considering the landside jobs of large instances, this assumption might not be realistic. However, we decided to keep it for two reasons. First, this allows a comparison with the existing approaches from the literature. Second, there is a fast development of information systems, and terminal operators already seek for obtaining early

information on truck arrivals, e.g. through GPS signals. Moreover, we assume that cranes move along the block at a speed of around 3 meters per second, so that a time interval of PCSP-SL corresponds to roughly two seconds of real time. Furthermore, based on the real-world data, $p = 20$ is a reasonable assumption and varying the number of seaside containers between a few hundred and 1,000 closely mimics real-world settings. As mentioned in Section 1, the number of landside jobs is usually significantly smaller than the number of seaside jobs. However, in order to be able to answer Q5, we additionally generated instances with a large amount of landside jobs.

Table 4 depicts the parameter values that our testbed is based upon. The new parameters δ_r and

Table 4: Parameters of test instances

fixed					variable		
S	p	σ_w	σ_l	δ_r	n	m	δ_d
40	20	0	20	0.7	500, 1000, 2000	$\lfloor \frac{n}{20} \rfloor$, $\lfloor \frac{n}{10} \rfloor$, $\lfloor \frac{n}{4} \rfloor$, $\lfloor \frac{n}{2} \rfloor$, $\lfloor \frac{2n}{3} \rfloor$, $\lfloor \frac{3n}{4} \rfloor$	1.3, 1.6

δ_d were applied to generate time windows for the landside jobs (see Table 5). Here, small values δ_d on average result in small time windows. We generated 20 test instances for each combination of the parameter values of Table 4, which results in a total of 720 test instances. The remaining parameters of each instance were randomly drawn from uniform distributions on the intervals depicted in Table 5, where $\bar{C}_{sea} = 2pn + Sn - \frac{S}{2}$ is the expected seaside makespan when assuming that there exist no landside containers and that preemption of jobs is not allowed. Within the project PCSP-SL_Data, the data of

Table 5: Intervals of uniform distributions for generating the container data

$s_i, w_i \in I$	$a_j, l_j \in J$	$r_j, l_j \in J$	$d_j, l_j \in J$
$[1, S]$	$[1, S]$	$[0, \lfloor \delta_r \cdot \bar{C}_{sea} \rfloor]$	$[r_j, \lfloor \delta_d \cdot \delta_r \cdot \bar{C}_{sea} \rfloor]$

all instances of a specific parameter combination n, m, and δ_d is stored in a CSV file "$n\_m\_\delta_d.csv$", e.g. "500_25_1.3.csv" for $n = 500$, $m = 25$, and $\delta_d = 1.3$.

5.2. Computational Results

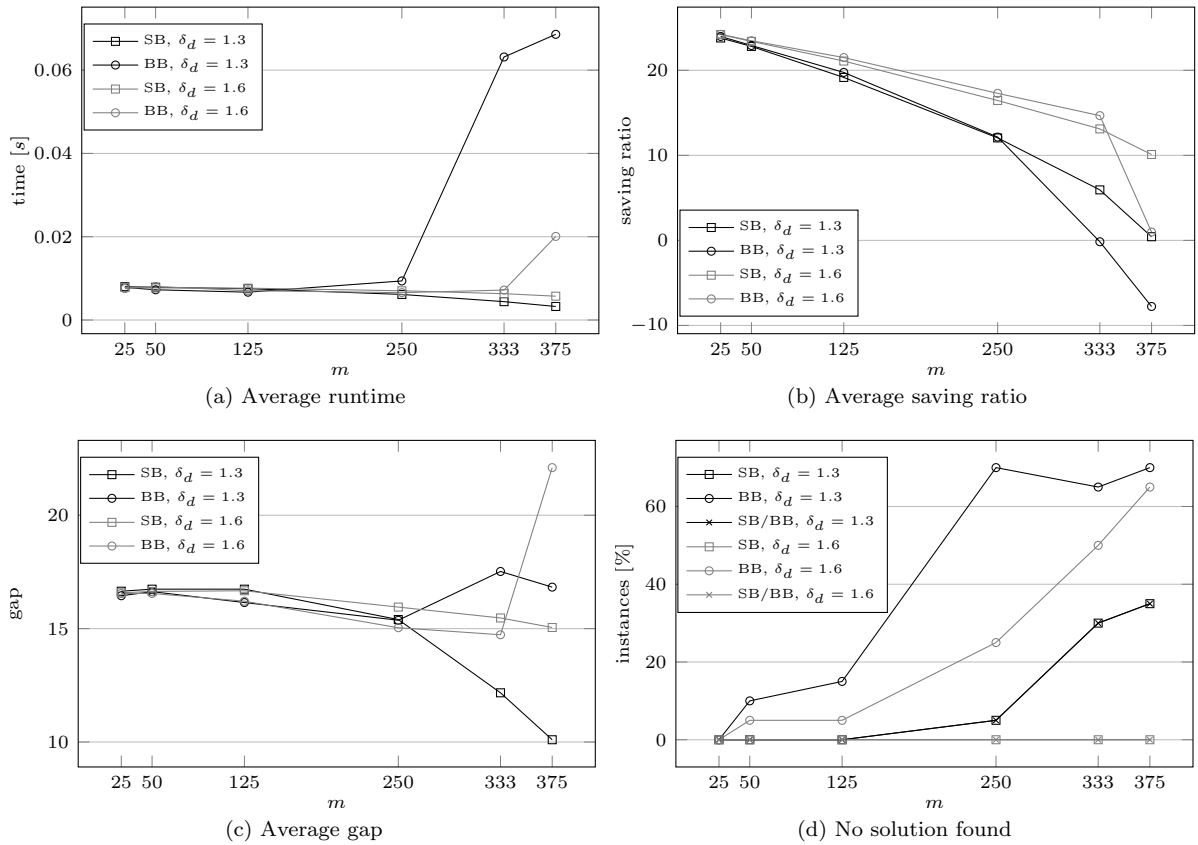
Consider an instance I of PCSP-SL with $\sigma_w = 0$ and let $C_{sea}(I) = 2pn + 2\sum_{i=1}^{n} s_i - s_n$ be the optimal seaside makespan when assuming that there exist no landside containers and that preemption of jobs is not allowed. Furthermore, denote by $C_{max}(sol_I)$ the seaside makespan of a solution sol_I of I. Then we define the *saving ratio* as a measure for the effect of allowing cooperation and the quality of this solution:

$$sav(sol_I) := 100 \cdot \frac{C_{sea}(I) - C_{max}(sol_I)}{C_{sea}(I)}.$$

Similarly, the *gap* of a solution sol_I with respect to the lower bound $LB^*(I)$ determined by Algorithm 1 is defined as follows:

$$gap(sol_I) := 100 \cdot \frac{C_{max}(sol_I) - LB^*(I)}{C_{max}(sol_I)}.$$

Figure 3 presents the computational results for the small instances with $n = 500$ seaside jobs. Here, SB (simple bucket brigade) refers to Algorithm 2, while BB (bundling bucket brigade) refers to Algorithm 3. Figure 3a plots the average runtimes of the heuristics over the number of landside jobs.



(a) Average runtime

(b) Average saving ratio

(c) Average gap

(d) No solution found

Figure 3: Results for the small instances, $n = 500$

Each data point corresponds to the average runtime of those test instances of the specific instance set for which a feasible solution was found. The black (gray) curves correspond to instances with small (large) time windows, i.e. $\delta_d = 1.3$ ($\delta_d = 1.6$). Similarly, Figures 3b and 3c depict results on the average saving ratios and the average gaps as defined above. Again, this relates to the instances for which a feasible solution was found. Finally, Figure 3d illustrates the percentage of instances of each set for which no feasible solution was found with the respective algorithms. It includes curves (SB/BB) that relate to the number of instances for which no feasible solution was found by any of the algorithms.

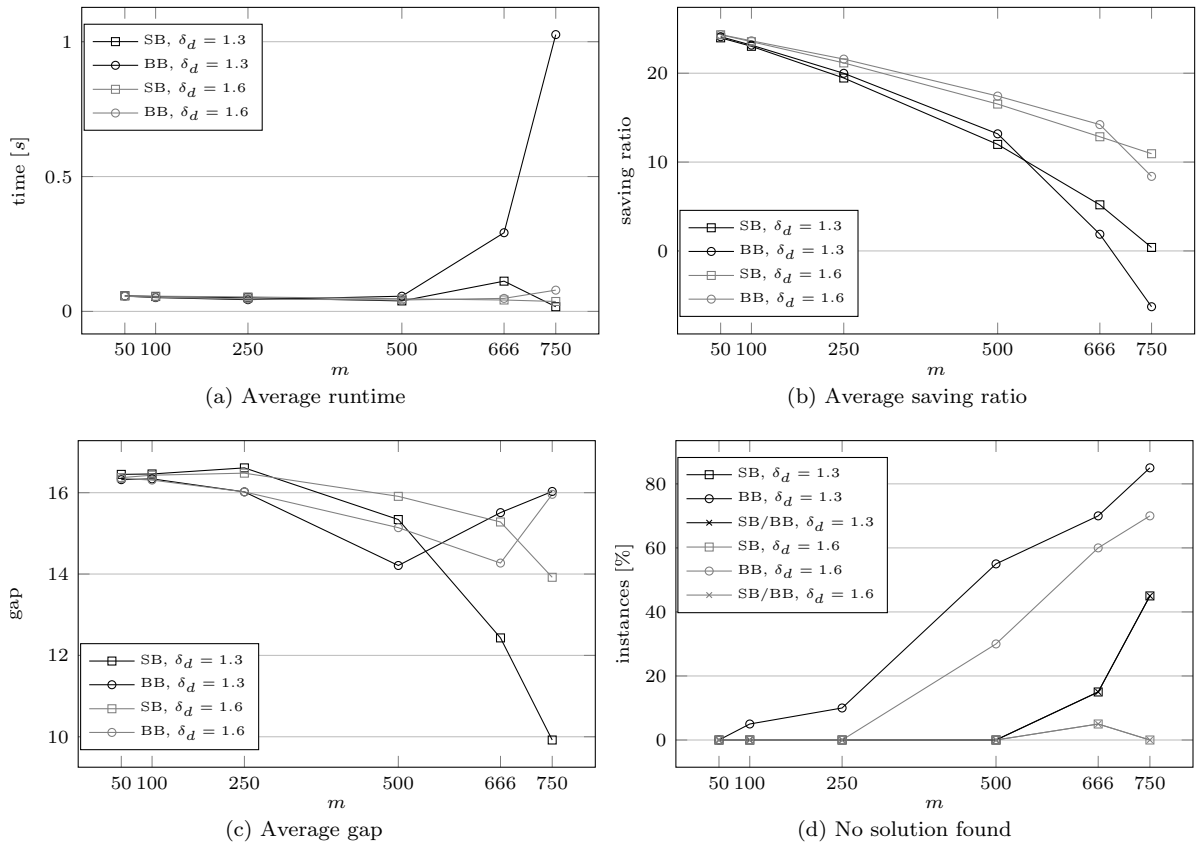
Figures 4 and 5 depict the computational results for the medium ($n = 1000$) and large ($n = 2000$) instances, respectively.

The numerical values that correspond to Figures 3–5 are presented in Table A.6 in the appendix. Table A.7 in the appendix presents some complementary results. Most important, it shows that our method of constructing time windows (Table 5) is such that the landside jobs are indeed relevant when constructing feasible solutions.

5.3. Evaluation of the Results

We can now answer the research questions posed above by evaluating the computational results.

Concerning Q1, Figures 3b, 4b, and 5b indicate that BB on average performs better than SB in terms of solution quality for small ratios m/n, while the opposite is true for large ratios m/n. The intersection point of the corresponding curves shifts towards larger ratios m/n when increasing the time



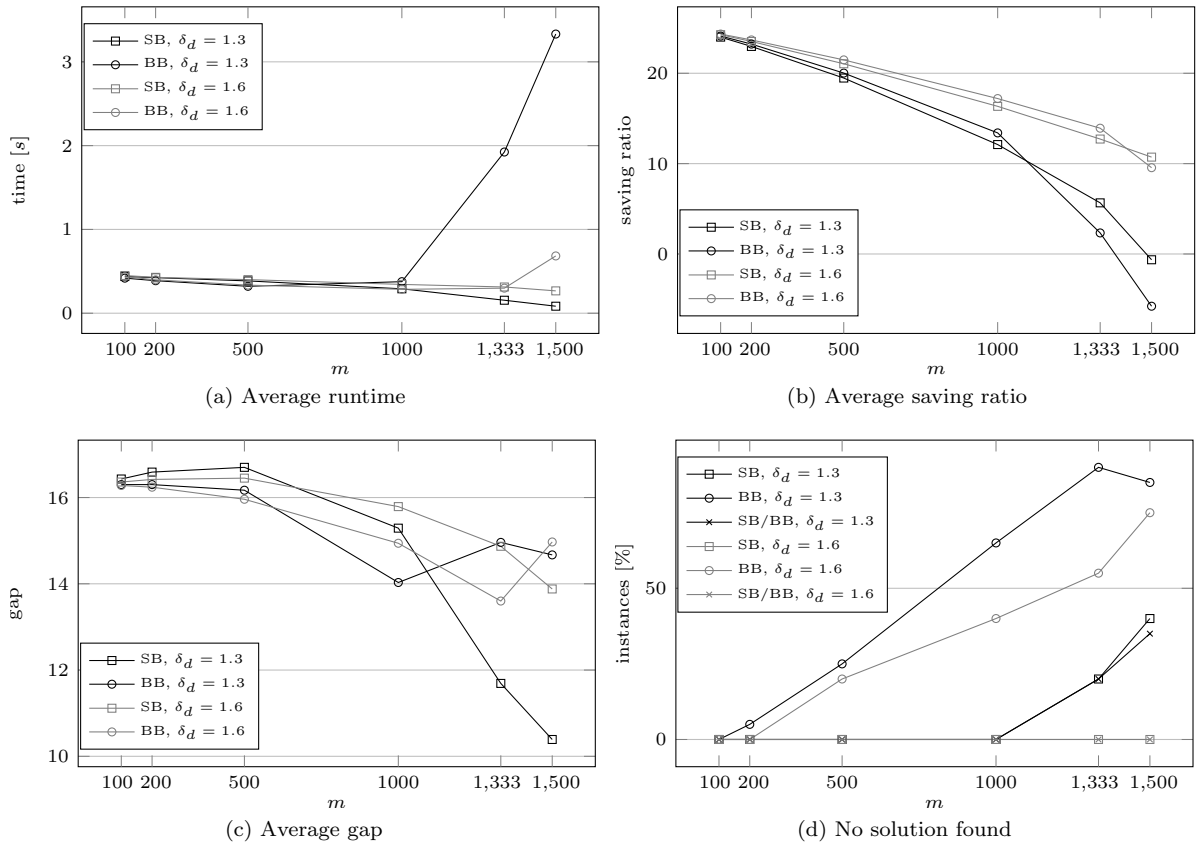
(a) Average runtime

(b) Average saving ratio

(c) Average gap

(d) No solution found

Figure 4: Results for the medium instances, $n = 1000$



(a) Average runtime

(b) Average saving ratio

(c) Average gap

(d) No solution found

Figure 5: Results for the large instances, $n = 2000$

windows of the landside jobs. Additionally, Figures 3d, 4d, and 5d show that SB clearly outperforms BB with respect to the percentage of instances for which a feasible solution is found. Note, however, that there are sporadic instances, for which BB finds a feasible solution while SB does not (see Figure 5d). These results confirm the suitability of the basic principles that the algorithms are based upon (see Section 4.2), i.e. the assumption that bundling reduces the ability of a bucket brigade framework to find feasible solutions, while it improves the quality of potential solutions by reducing unloaded moves of the landside crane. Finally, note that the curves that represent the percentage of instances for which no feasible is found (Figures 3d, 4d, and 5d) tend to increase when increasing the number of landside containers. Similarly, this percentage is nondecreasing when time windows become smaller. Both effects are indicated by Proposition 1.

The average runtimes of the heuristics range from only a few milliseconds to about 3.5 seconds (see Figures 3a, 4a, and 5a). Moreover, we encountered a maximum runtime of about 5 seconds when only considering the instance-algorithm combinations that resulted in a feasible solution (see Table A.7 in the appendix). The number of calls of the backtracking procedures of the heuristics may be rather large, so that the maximum time needed to terminate without having found a feasible solution was about 16 seconds (Table A.7). Nevertheless, these results allow for positively answering Q2 by concluding that both heuristics are applicable in real-life scenarios.

Figures 3b, 4b, and 5b suggest that allowing the gantry cranes to cooperate in the presence of landside containers offers the possibility to significantly reduce the seaside makespan when compared to solely assigning the seaside crane to the processing of the seaside containers. This holds true as long as the amount of landside containers is sufficiently small. More specifically, our computational results indicate that saving ratios larger than 10 are possible if $m \leq \lfloor 0.5 \cdot n \rfloor$ even for relatively small time windows. When additionally noting that real-world problem settings will hardly ever feature this many landside containers, and when observing that the corresponding average gaps (Figures 3c, 4c, and 5c) are fairly large (Q3), we can conclude by positively answering our main research question Q4. The finding of Briskorn et al. [5] (see Section 1) therefore remains true when landside containers are present: Cooperation "is something for terminal operators to consider, seeing that it is not an uncommon policy in practice to assign only the seaside crane exclusively to stacking containers, while the landside crane is supposed to exclusively handle container transfers to the hinterland."

Regarding Q5, we conclude that the critical ratio of landside jobs to seaside jobs is sufficiently large to not play an important role in realistic problem settings arising at seaports. There are two relevant practical insights. First, as to be expected, tight time windows reduce this ratio and result in less savings elicited by cooperation. Second, the proposed heuristics may actually result in negative saving ratios. This is due to waiting operations when generating handover containers and, in case of the bundling procedure (Algorithm 3), because of not allowing the seaside crane to block the landside crane's passage when the latter is processing a bundle (see Section 4.2).

6. Conclusion and Outlook

We considered a problem arising in a seaport storage area in which two rail mounted gantry cranes can perform container moves, but where interference constraints between these cranes have to be respected. Once a ship is berthed, highest priority is given to quickly unloading this ship and to putting the containers into the storage area. However, due to the interference constraints, only the seaside crane can pick up the containers from the input/output point. Yet, one may wonder whether significant time savings can be achieved if the landside crane supports the seaside crane by means of handover containers. Under the assumption that the landside crane does not have to perform other jobs, this question is positively answered by Briskorn et al. [5]. In this paper, we answered this question for the case of the landside crane having to perform some jobs on the landside of the block. To do so, we analyzed the problem complexity and gave some counterintuitive examples of optimal solutions. For example, there are instances in which the landside crane moves left while being loaded in any optimal solution, although the overall container flow is exclusively to the right. A lower bound and two heuristic algorithms were presented. In a computational study, we were able to show that the heuristics deliver promising results if we consider realistic instances. Our main research question was then positively answered: Compared to the situation in which seaside containers are exclusively served by the seaside crane, cooperation might reduce the makespan by more than 20%.

Our main focus was the situation in which the sequence of the seaside containers is given. This assumption is not unrealistic as the unloading sequence from a ship is usually highly restricted by physical constraints such as stacking orders or the stability of the ship. Yet, there might be at least some degrees of freedom when determining the sequence of the seaside containers. Future research could analyze how much this freedom could result in time savings. The promising results obtained in this paper, especially the rather small gap between the heuristic procedures and the lower bound, suggests to analyze the problem at hand on a higher problem level. For example, positioning of containers within a block, the assignment of containers to blocks, the scheduling of quay cranes, etc. could be considered in a combined approach.

Acknowledgement

This work has been supported by the German Science Foundation (DFG) through the grant "Scheduling mechanisms for rail mounted gantries with regard to crane interdependencies" (JA 2311/2-1, PE 514/22-1).

References

[1] N. Kemme, Effects of storage block layout and automated yard crane systems on the performance of seaport container terminals, OR Spectrum 34 (3) (2012) 563–591.

[2] U. Speer, G. John, K. Fischer, Scheduling yard cranes considering crane interference, in: J. W. Böse, H. Hu, C. Jahn, X. Shi, R. Stahlbock, S. Voß (Eds.), Computational Logistics, Springer, Berlin, 321–340, 2011.

[3] A. Ehleiter, F. Jaehn, Housekeeping: foresightful container repositioning, International Journal of Production Economics 179 (2016) 203–211.

[4] M. Y. Kovalyov, E. Pesch, A. Ryzhikov, Scheduling Container Storage Operations of Two Non-passing Stacking Cranes, Working Paper, 2016.

[5] D. Briskorn, S. Emde, N. Boysen, Cooperative twin-crane scheduling, Discrete Applied Mathematics 211 (2016) 40–57.

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

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

[8] C. Bierwirth, F. Meisel, A survey of berth allocation and quay crane scheduling problems in container terminals, European Journal of Operational Research 202 (3) (2010) 615–627.

[9] C. Bierwirth, F. Meisel, A follow-up survey of berth allocation and quay crane scheduling problems in container terminals, European Journal of Operational Research 244 (3) (2015) 675–689.

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

[11] G. Froyland, T. Koch, N. Megow, E. Duane, H. Wren, Optimizing the landside operation of a container terminal, OR Spectrum 30 (1) (2008) 53–75.

[12] W. C. Ng, Crane scheduling in container yards with inter-crane interference, European Journal of Operational Research 164 (1) (2005) 64–78.

[13] U. Dorndorf, F. Schneider, Scheduling automated triple cross-over stacking cranes in a container yard, OR Spectrum 32 (3) (2010) 617–632.

[14] G. Erdoğan, M. Battarra, G. Laporte, Scheduling twin robots on a line, Naval Research Logistics 61 (2) (2014) 119–130.

[15] N. Boysen, D. Briskorn, S. Emde, A decomposition heuristic for the twin robots scheduling problem, Naval Research Logistics 62 (1) (2015) 16–22.

[16] D. Briskorn, F. Jaehn, A. Wiehl, A test suite for scheduling algorithms for cranes in transshipment terminals, Submitted, 2017.

[17] Y. A. Saanen, M. V. Valkengoed, Comparison of three automated stacking alternatives by means of simulation, in: Proceedings of the 37th Conference on Winter Simulation, ACM, 1567–1576, 2005.

Appendix A. Computational Results: Numerical Values

Table A.6: Computational results

instance set	avg. time [s][1]		avg. sav. rat.[1]		avg. gap[1]		no solution [%]		
n, m, δ_d	SB	BB	SB	BB	SB	BB	SB	BB	SB/BB
500, 25, 1.3	0.008	0.008	23.8	23.98	16.65	16.45	0	0	0
500, 50, 1.3	0.008	0.007	22.81	22.92	16.74	16.63	0	10	0
500, 125, 1.3	0.008	0.007	19.16	19.75	16.74	16.15	0	15	0
500, 250, 1.3	0.006	0.009	12.05	12.15	15.4	15.37	5	70	5
500, 333, 1.3	0.004	0.063	5.94	-0.16	12.17	17.52	30	65	30
500, 375, 1.3	0.003	0.069	0.43	-7.77	10.1	16.83	35	70	35
1000, 50, 1.3	0.058	0.057	23.98	24.11	16.45	16.32	0	0	0
1000, 100, 1.3	0.055	0.051	23.03	23.16	16.46	16.34	0	5	0
1000, 250, 1.3	0.051	0.043	19.46	19.99	16.61	16.02	0	10	0
1000, 500, 1.3	0.039	0.056	11.99	13.18	15.34	14.21	0	55	0
1000, 666, 1.3	0.112	0.292	5.19	1.88	12.43	15.51	15	70	15
1000, 750, 1.3	0.017	1.026	0.39	-6.28	9.92	16.03	45	85	45
2000, 100, 1.3	0.438	0.419	24	24.12	16.43	16.3	0	0	0
2000, 200, 1.3	0.424	0.388	22.97	23.23	16.59	16.3	0	5	0
2000, 500, 1.3	0.385	0.321	19.47	20.02	16.7	16.17	0	25	0
2000, 1000, 1.3	0.291	0.376	12.11	13.4	15.29	14.03	0	65	0
2000, 1333, 1.3	0.155	1.924	5.66	2.33	11.69	14.96	20	90	20
2000, 1500, 1.3	0.084	3.332	-0.64	-5.78	10.39	14.67	40	85	35
500, 25, 1.6	0.008	0.008	24.2	24.23	16.58	16.55	0	0	0
500, 50, 1.6	0.008	0.008	23.4	23.48	16.64	16.55	0	5	0
500, 125, 1.6	0.008	0.007	21.09	21.5	16.67	16.21	0	5	0
500, 250, 1.6	0.007	0.007	16.45	17.29	15.95	15.04	0	25	0
500, 333, 1.6	0.006	0.007	13.11	14.67	15.47	14.73	0	50	0
500, 375, 1.6	0.006	0.02	10.1	0.98	15.05	22.1	0	65	0
1000, 50, 1.6	0.058	0.056	24.33	24.35	16.37	16.35	0	0	0
1000, 100, 1.6	0.056	0.053	23.55	23.65	16.43	16.31	0	0	0
1000, 250, 1.6	0.054	0.046	21.16	21.59	16.48	16.02	0	0	0
1000, 500, 1.6	0.047	0.042	16.53	17.43	15.91	15.14	0	30	0
1000, 666, 1.6	0.042	0.048	12.85	14.22	15.28	14.27	5	60	5
1000, 750, 1.6	0.037	0.079	10.94	8.39	13.92	15.96	0	70	0
2000, 100, 1.6	0.449	0.436	24.27	24.34	16.36	16.28	0	0	0
2000, 200, 1.6	0.428	0.398	23.52	23.69	16.42	16.24	0	0	0
2000, 500, 1.6	0.401	0.335	21.05	21.48	16.45	15.96	0	20	0
2000, 1000, 1.6	0.344	0.285	16.34	17.2	15.79	14.94	0	40	0
2000, 1333, 1.6	0.313	0.299	12.72	13.92	14.87	13.6	0	55	0
2000, 1500, 1.6	0.267	0.683	10.72	9.55	13.88	14.97	0	75	0

[1] Considering only those instances of the set for which a feasible solution was found.

Table A.7: Additional computational results

instance set	max. time [s][1]		max. time [s][2]		J_{compl} [%][3]	
n, m, δ_d	SB	BB	SB	BB	SB	BB
500, 25, 1.3	0.01	0.01			77.2	76.8
500, 50, 1.3	0.01	0.01		0.01	75.7	75.11
500, 125, 1.3	0.01	0.01		0.01	82.64	81.22
500, 250, 1.3	0.01	0.01	9.35	0.02	96.17	94.87
500, 333, 1.3	0.01	0.14	0.01	0.04	100	100
500, 375, 1.3	0.004	0.19	0.003	0.03	100	100
1000, 50, 1.3	0.07	0.09			72.9	72.6
1000, 100, 1.3	0.06	0.06		0.01	74.35	74.26
1000, 250, 1.3	0.06	0.08		0.01	80.86	80.18
1000, 500, 1.3	0.04	0.08		0.05	96.11	94.24
1000, 666, 1.3	0.89	0.83	0.21	0.3	100	100
1000, 750, 1.3	0.04	2.21	0.01	0.3	100	100
2000, 100, 1.3	0.46	0.44			72.5	72.15
2000, 200, 1.3	0.44	0.41		0.29	72.58	72
2000, 500, 1.3	0.42	0.34		0.24	80.68	79.63
2000, 1000, 1.3	0.36	0.69		0.38	96.1	93.34
2000, 1333, 1.3	0.52	2.08	0.73	15.61	100	100
2000, 1500, 1.3	0.22	4.92	0.11	2.58	100	100
500, 25, 1.6	0.01	0.01			51.2	51
500, 50, 1.6	0.01	0.02		0.01	52.1	51.68
500, 125, 1.6	0.01	0.01		0.01	55.24	54.78

500, 250, 1.6	0.01	0.01		0.01	61.62	60.27
500, 333, 1.6	0.01	0.01		0.01	65.62	61.65
500, 375, 1.6	0.01	0.05		0.02	71.35	82.51
1000, 50, 1.6	0.06	0.06			51.2	51.3
1000, 100, 1.6	0.06	0.06			52.45	52.35
1000, 250, 1.6	0.06	0.05			54.86	54.24
1000, 500, 1.6	0.05	0.06		0.04	61.56	60.09
1000, 666, 1.6	0.05	0.06	0.06	0.04	67.01	64.43
1000, 750, 1.6	0.04	0.2		0.3	69.91	72.07
2000, 100, 1.6	0.48	0.47			53.1	53.05
2000, 200, 1.6	0.45	0.41			53.45	53.3
2000, 500, 1.6	0.42	0.35		0.3	56.78	56.43
2000, 1000, 1.6	0.37	0.35		0.22	63.43	62.35
2000, 1333, 1.6	0.35	0.4		0.25	68	66.73
2000, 1500, 1.6	0.3	1.11		0.8	70.93	71.96

[1] Considering only those instances of the set for which a feasible solution was found.

[2] Considering only those instances of the set for which no feasible solution was found.

[3] J_{compl}: Average percentage of landside jobs that are dropped in slot $S + 1$ in the feasible solutions determined by the algorithm.