Briskorn, Dirk

Article — Published Version

Routing two stacking cranes with predetermined container sequences

Journal of Scheduling

Provided in Cooperation with:
Springer Nature

Suggested Citation: Briskorn, Dirk (2021) : Routing two stacking cranes with predetermined container sequences, Journal of Scheduling, ISSN 1099-1425, Springer US, New York, NY, Vol. 24, Iss. 4, pp. 367-380, https://doi.org/10.1007/s10951-021-00689-4

Journal of Scheduling (2021) 24:367–380
https://doi.org/10.1007/s10951-021-00689-4

Routing two stacking cranes with predetermined container sequences

Dirk Briskorn[1]

Accepted: 28 April 2021 / Published online: 19 June 2021

Abstract
The scheduling of gantry cranes with respect to mutual interference has received considerable attention in recent years. We consider a subproblem which arises when each crane has a sequence of tasks to be assigned. The problem is concerned with resolving the interference between two cranes by determining which crane avoids the other in order to let it complete its next task first. We provide a fairly general problem framework accounting for different crane systems and various side constraints. We assume a cost function for each task that determines the cost of completing the task at a specific point in time. We then distinguish between the objectives to minimize both the total cost and the maximum cost among tasks. A general dynamic programming framework is provided which allows us to solve all problem versions in pseudo-polynomial time. Furthermore, we show that while the general problem aiming for minimum total cost is binary NP-hard, the general problem aiming for minimum maximum cost can be solved in polynomial time. Finally, we address two important special cases of the former, and we show that they can be solved in polynomial time as well.

Keywords Port operations · Container logistics · Gantry crane scheduling · Resolving mutual interference

1 Introduction

Over the past few decades, the automation of seaport container terminal equipment has steadily increased. Automated rail-mounted gantry cranes (RMGs) represent a major success story from this period, and they have allowed terminal designers to increase the number of containers processed and stored with less cost and fewer space requirements.

Typically, in seaport container terminals, containers are temporarily stored in blocks, and different types of devices are available to manage those blocks. Among them, we find RMGs which span the whole storage block in width and move on tracks installed alongside the block. As opposed to straddle carriers and rubber-tired gantry cranes, which are not automated, RMGs are fixed to a certain block within the container storage area. As a consequence, they can handle containers only within the storage area and hence have to hand over the containers to transport devices or receive containers from them at the short side of the block.

✉ Dirk Briskorn
briskorn@uni-wuppertal.de
https://www.prodlog.uni-wuppertal.de/

[1] Lehrstuhl für BWL, Insbesondere Produktion und Logistik, Bergische Universität Wuppertal, Gaußstraße 20, 42119 Wuppertal, Germany

We can find one of four different settings of one to three RMGs managing a single block. First, there might be a single crane managing the block. Second, we can have two identical cranes running on the same pair of rails. Obviously, these cranes cannot pass each other, and thus one is always located closer to the seaside and the other closer to the landside. We refer to cranes in this setting as twin cranes. In a third scenario, we might find two cranes of different sizes running on separate pairs of rails. The smaller crane can move below the larger crane, and thus these cranes can pass each other. However, the smaller crane can be located below the larger crane only if the larger crane's spreader is up. We refer to cranes in this setting as crossover cranes. Finally, there might be three cranes, two identical smaller ones and a larger one. The smaller ones cannot pass each other, but both can pass the larger one. In this paper, the focus is on settings with two cranes.

In addition to resolving the technological issues arising with process automation, sophisticated scheduling techniques are required. While scheduling a single crane might already be a challenging task (see Gharehgozli et al. (2014)), the problem becomes much more than just sequencing of tasks if multiple cranes are considered. First, it must be determined which crane is to conduct which task. Second, because the cranes might interfere with each other, the right

of way must be established. We can imagine a hierarchical approach on three levels to execute this three-part decision. First, we determine the assignment of tasks to cranes, then the task sequences for each crane, and finally the right of way in the case of interference. The subproblem on the third level has been considered in Briskorn and Angeloudis (2016) and Briskorn and Zey (2018). The approaches developed have proved beneficial as workhorses in more involved optimization approaches, tackling the three decision levels in an integrated manner in Briskorn and Zey (2020), Nossack et al. (2018), and Zey et al. (2020). Eilken (2019) generalizes the approach for twin cranes in Briskorn and Angeloudis (2016) and employs it to solve a subproblem as well.

Each of the aforementioned approaches aims for a minimum makespan schedule. While it is easy to motivate makespan minimization, there are many other objective functions that might be interesting. For example, due dates might be associated with tasks representing containers to be handed over to trucks, and a reasonable objective would be to minimize total tardiness of tasks. In this paper, we investigate the subproblem on the third level for two cranes and various objective functions, and we develop algorithms for the respective optimization problems. These algorithms could be employed as subroutines in future research, similarly to those in Briskorn and Angeloudis (2016) and Eilken (2019).

1.1 Literature overview

There is a huge body of scientific literature on optimization of container transport in ports. We refer to the survey papers on port operations in general in Stahlbock and Voß (2008); Steenken et al. (2004) and Vis and de Koster (2003), and in particular on seaside operations in Bierwirth and Meisel (2010, 2015) and Carlo et al. (2015), intermediate storage in container yards in Carlo et al. (2014a), ground container transport in ports in Carlo et al. (2014b), and hinterland railway access in Boysen et al. (2013). It should be noted that there are similarities between scheduling problems for cranes and for other devices; see Boysen et al. (2017) for an overview.

We focus our literature overview on scheduling approaches for multiple cranes working on the same container block, since interference is obviously not an issue for a single crane operating in a container block. When considering more than one crane, potential conflicts between the individual crane schedules must be resolved in order to obtain compatible schedules. While there exist some approaches for scheduling two (or more) cranes on a storage block, few of them explicitly address interference.

Dorndorf and Schneider (2010) propose an algorithm that schedules three cranes working jointly on a block under a multi-criteria objective. Choe et al. (2007, 2012) and Li et al. (2009, 2012) examine the problem of scheduling twin cranes and consider interference. Gharehgozli et al. (2015) provide a heuristic approach tackling the problem of scheduling twin cranes when storage and retrieval operations are given under the objective of minimizing the makespan. Boysen et al. (2015) tackle the same problem using a decomposition approach. Briskorn et al. (2016) schedule two identical cranes with the ability for one crane to hand over a container to the other by setting it down in an intermediate position. The setting is restricted such that containers are exchanged between cranes and other devices only on one side of the block. Jaehn and Kress (2018) and Kress et al. (2019) extend the setting to handovers on both sides. Kovalyov et al. (2018) examine the computational complexity of scheduling twin cranes under various protocols for assigning tasks to cranes.

Problems on the third level of the three-part decision, namely determining the right of way in the case of interference, have mostly been solved heuristically. Only a few papers have focused on analyzing such problems or solving them exactly. Since this is the focus of the paper at hand, we now consider those papers in more detail. With both an assignment of tasks to cranes and processing sequences being given, Briskorn and Angeloudis (2016) determine conflict-free schedules with minimum makespan in strongly polynomial time for two cranes, either twin cranes or crossover cranes. In order to design a dynamic programming (DP) approach, a geometrical model is developed which is transformed into a graph model. Conflict-free schedules can then be achieved by finding the shortest path in this graph. Since the graph happens to be acyclic, this can be done very efficiently. No further side constraints or other objectives are considered. In Nossack et al. (2018), a two-stage decomposition approach is proposed to determine the task assignment and task sequences for two crossover cranes in the first stage, while the right of way is decided in the second stage. Here, the shortest path approach developed in Briskorn and Angeloudis (2016) is employed to determine the right of way for given candidate assignments and sequences. With a restriction to crossover cranes, Nossack et al. (2018) refine the approach by Briskorn and Angeloudis (2016) and speed it up slightly. The approach by Briskorn and Angeloudis (2016) is employed in Zey et al. (2020), where two twin cranes can hand over containers in a dedicated position. A branch-and-bound algorithm is developed where task assignment and task sequences are determined by branching, and the right of way is decided by evaluating leaf nodes of the search tree. The shortest path approach developed in Briskorn and Angeloudis (2016) is refined to account for precedence constraints and explicit consideration of trolley moves and is then used to determine the right of way. Eilken (2019) also provides a branch-and-bound approach for scheduling two twin cranes. However, containers cannot be handed over from one crane to the other in this case. Again, task assignment and task sequences are determined

Table 1 Approaches for right-of-way decisions

	Cranes			Dim.		Operation characteristics						Objective function		
	Twin	CO	Triple	G	T	p_j	w_j	e_j	d_j	$\overline{d}_j$	prec	$C_{\max}$	$\max f_j$	$\sum f_j$
Briskorn and Angeloudis (2016)	✓	✓		✓		✓						✓		
Briskorn and Zey (2018)			✓	✓		✓						✓		
Eilken (2019)	✓			✓	✓	✓		✓	✓		✓	✓		
Nossack et al. (2018)		✓		✓		✓						✓		
Zey et al. (2020)	✓			✓	✓	✓					✓	✓		
This paper	✓	✓		✓	✓	✓	✓	✓	✓	✓		(✓)	✓	✓

CO, crossover cranes; G, gantry; T, trolley; p_j, lift/release durations; w_j, weights representing importance of operations; e_j, earliest operation start time; d_j, hard deadlines for operations; $\overline{d}_j$, soft due dates of operations; prec, precedence constraints; $C_{\max}$, makespan minimization; $\max f_j$, minimization of maximum cost; $\sum f_j$, minimization of total cost

by branching, and the right of way is decided by evaluating leaves of the search tree. For evaluation of leaf nodes, the approach in Briskorn and Angeloudis (2016) is extended by accounting for release dates, deadlines, and precedence constraints and explicitly considering trolley moves. Finally, Briskorn and Zey (2018) extend the approach developed by Briskorn and Angeloudis (2016) to a three-crane setting. Following the same basic idea, a geometrical model is developed which allows for a DP approach. However, the authors do not resolve the computational complexity of the problem. The extended approach is also employed in Briskorn and Zey (2020), where a branch-and-bound algorithm is again developed with task assignment and task sequences determined by branching, and right of way by evaluating leaf nodes of the search tree. For the latter, the approach proposed in Briskorn and Zey (2018) is employed. While all these papers tackle the problem on the third level of the three-part decision, they differ with respect to the crane setting and whether trolley moves and the characteristics of tasks are considered explicitly. Table 1 gives an overview of the features covered in these papers and in the current one.

As shown in Table 1, existing approaches aim only for makespan minimization. In the present paper, a much more general class of objective functions is considered, namely minimizing the maximum cost or minimizing the total cost, where the cost of an operation is determined by an individual function mapping its completion time on a cost value. Note that minimization of maximum cost includes minimization of the makespan. Furthermore, the problems in in Briskorn and Angeloudis (2016) are generalized in the present paper by considering trolley moves explicitly and taking into account various operational characteristics, and the problems in Eilken (2019), Nossack et al. (2018), and Zey et al. (2020) are generalized by considering both twin cranes and crossover cranes. The problem setting in Briskorn and Zey (2018) considers triple cranes but is far more restricted with respect to trolley moves, operation characteristics, and objective function.

1.2 Contribution and outline

The contribution of this paper is threefold. First, we formalize the problem setting, distinguishing between crossover cranes and twin cranes and two types of objective functions. Second, we provide a pseudo-polynomial-time algorithm for the most general problem settings considered. Third, we show that, depending on the type of objective function, the problem is NP-hard or can be solved in polynomial time. Furthermore, we provide strongly polynomial-time algorithms for two important special cases.

We restrict ourselves to a deterministic and offline perspective on the optimization problem, for two reasons. First, it allows for a thorough analysis of the inherent complexity of the problem. As we will see, the computational complexity of the problem is heavily dependent on the objective function under consideration. We hope that these insights and the algorithmic framework we develop will provide a valuable starting point for developing algorithms for more involved problems as well, for example, those addressing the first two levels of the three-part decision. Second, while it is true that container terminals are highly dynamic and stochastic environments which often require frequent revising of decisions, those revisions will most likely also involve the first two levels. Hence, in order to account for the dynamic and stochastic nature, we need an integrated approach for the three-part decision. We can easily imagine how a more involved approach, potentially applied in a rolling horizon scheme, bears the deterministic (sub)problem setting considered in this paper, and this approach might benefit from a component that can efficiently solve the deterministic subproblem. We provide such a component.

The paper proceeds as follows. In Sect. 2, we outline the problem characteristics and formally define the problem settings. Section 3 focuses on the general problem settings. Here, we show that it is binary NP-hard or can be solved in polynomial time depending on the objective function considered. Section 4 highlights important special cases that can

Fig. 1 Gantry cranes at container terminal Altenwerder, Jessen (2020)

be solved in strongly polynomial time. Finally, we conclude the paper in Sect. 5.

2 Problem settings

In this section, we first provide an informal problem description in Sect. 2.1 and then a formal one in Sect. 2.2.

2.1 Problem description and assumptions

We consider a block in the storage area of a container terminal managed by two cranes. Each crane's gantry spans the whole block in width and moves on a pair of rails installed alongside the block. There is a trolley which moves on the horizontal beam of the gantry and from which a spreader can be lowered to pick up or release a container. By moving the gantry, trolley, and spreader, each position in the three-dimensional block can be reached. Furthermore, the gantry can move outside the block to a limited extent to reach automated guided vehicles or trucks for handing over containers to them or receiving containers from them. We refer to the areas for handovers as seaside and landside handover areas, respectively. Figure 1 depicts multiple such blocks with two cranes each.

Interference between cranes depends on the crane setting. For two twin cranes which cannot pass each other (as depicted in Fig. 1), we require that one is closer to the seaside handover area than the other throughout the planning horizon. For two crossover cranes, we require that the positions of their gantries do not overlap when the larger crane's spreader is lowered. That is, both cranes can pass each other and can be present in the same gantry position as long as the larger crane has its spreader up.

For each crane, we have a predetermined sequence of containers given. Each container is associated with an origin position within the block (including positions in handover areas) and a destination position within the block (including positions in handover areas). Moreover, for each container, the time necessary for lifting it up from its origin position and the duration for dropping it off at its destination position is given. These container sequences prescribe the order in which each crane conducts the container transport assigned to it. Also, for each container and both its pickup and its drop-off, we have a release date denoting the first time point that it can be started and a deadline denoting the last time point at which it can be completed. Finally, two cost functions are given for each container, mapping the time a container is lifted and the time a container is set down to a cost value. How these cost values are consolidated to a schedule's objective value depends on the objective function itself.

In this paper, we impose simplifying assumptions, as follows.

- The sequences of containers to be transported are fully known, and the containers' properties are deterministic. The problem setting we consider is very short-term, and thus this restriction seems to be acceptable.
- Interference between cranes depends only on the positions of their gantries. Obviously, while this might not cover all kinds of interference, it fully accounts for physical interference.
- The gantry and the trolley of each crane move either with full speed or not at all, that is, we neglect bounded acceleration. Furthermore, the full gantry speeds for the two cranes are identical.
- The spreader can only be moved while the gantry and trolley stand still. This is a technical prerequisite of many cranes and a safety restriction in most container terminals. However, the gantry and the trolley of a crane can move in parallel.
- The cost functions are non-decreasing in time. As long as release dates are respected, we typically will not suffer from completing lifting or drop-off as early as possible, since it provides more flexibility to other devices exchanging containers with cranes.
- We have a grid of discrete gantry and trolley positions of cranes in the block which needs to be considered in order to pick up or drop off a container. Since blocks are typically discretized into slots able to host one stack of large containers or two stacks of small containers, these discrete positions occur naturally.
- Gantry positions of cranes are with respect to the center of the respective gantry, and gantries do not overlap if their positions differ by a distance of two consecutive gantry positions in the grid.

2.2 Problem definitions

We consider a time horizon of length Θ and storage positions of the yard block arranged in a two-dimensional grid. The slots in the first dimension are passed by the whole gantry, and we refer to them as bays, with $b = 1, \ldots, B$. The slots in the second dimension are passed by a crane's trolley and are referred to as rows, with $r = 1, \ldots, R$. Consequently, each storage position can be identified by a tuple $(b, r) \in \{1, \ldots, B\} \times \{1, \ldots, R\}$. Additionally, we have bay 0 with positions $(0, 1), \ldots, (0, R)$ representing the seaside handover area and bay $B + 1$ with positions $(B + 1, 1), \ldots, (B + 1, R)$ representing the landside handover area. The set of all positions is denoted by $P := \{0, \ldots, B + 1\} \times \{1, \ldots, R\}$.

For each crane $c \in \{1, 2\}$, we have an initial position (b_c^0, r_c^0) and a sequence σ_c of tasks with length n_c. We refer to the kth task in σ_c, $c \in \{1, 2\}$ as $\sigma_c(k)$ and denote the set of all tasks by J. For each task $j \in J$, the duration $p_j \in \mathbb{N}$, the earliest start time $e_j \in \mathbb{N}$, the deadline $d_j \in \mathbb{N}$ for completion, and the position $(b_j, r_j) \in P$ are given. Duration p_j covers the time span the gantry and trolley have to remain in position (b_j, r_j) in order to conduct j, that is, time for lowering the spreader, adjusting it, locking it to a container (unlocking it from the container), and lifting it. Hence, the cranes' moves along the third dimension which are not captured by position set P are covered by task durations. Note that we abstract from containers here and consider each lifting operation and each drop-off operation to be an individual task in σ_c. Since the sequence of tasks is fixed, it can conveniently represent a sequence of containers to be transported from given origin positions to given destination positions.

The cranes move their gantries requiring time $s^g = 1$ per bay and their trolleys requiring time $s^t \in \mathbb{N}^+$ per row. Thus, if no interference occurs, it takes a crane $\max\{|b - b'|, |r - r'| \cdot s^t\}$ time units to move from (b, r) to (b', r').

For the input described above, a routing for crane c, $c \in \{1, 2\}$, is specified by two continuous piecewise linear functions $p_c^g : [0, \Theta] \to [0, B+1]$ and $p_c^t : [0, \Theta] \to [1, R]$ with $p_c^g(0) = b_c^0$, $p_c^t(0) = r_c^0$, $\partial p_c^g / \partial\theta \in \{-1, 0, 1\}$, and $\partial p_c^t / \partial\theta \in \{-1/s^t, 0, 1/s^t\}$ (if differentiable at θ). Functions p_c^g and p_c^t then reflect the position of the gantry and the trolley, respectively, over time. Note that differentials taking values in $\{-1, 0, 1\}$ and $\{-1/s^t, 0, 1/s^t\}$ reflect gantries and trolleys either moving at full speed or not moving at all. We say crane c visits position (b, r) in time interval $[\theta, \theta']$ if $p_c^g(\theta'') = b$ and $p_c^t(\theta'') = r$ for each $\theta'' \in [\theta, \theta']$.

A routing of crane c is sequence-compatible if for each task $j \in \sigma_c$ there is a visit of length p_j in (b_j, r_j) such that

- the visit associated with task $\sigma_c(k)$ has a time interval preceding the one of the visits associated with task $\sigma_c(k+1)$ for each $k = 1, \ldots, n_c - 1$, and
- for each $k = 1, \ldots, n_c$ the time interval of the visit of $\sigma_c(k)$ is in $[e_j, d_j]$.

Table 2 Task properties

j	b_j	r_j	p_j	e_j	d_j
a	5	4	3	4	8
b	6	4	2	0	20
c	5	4	4	0	20
d	8	4	1	0	20

We will say that the end of the time interval of the visit associated with task $j \in J$ is its completion time C_j. For each task $j \in J$, a non-decreasing cost function f_j maps completion time C_j on a cost value $f_j(C_j)$.

A feasible solution is a pair $((p_1^g, p_1^t), (p_2^g, p_2^t))$ of sequence-compatible routings for cranes 1 and 2 which are compatible with each other with respect to interference. Here, we distinguish between the two types of crane settings.

- For a twin crane system, we assume crane 1 to be the crane located towards the seaside. Then, a pair $((p_1^g, p_1^t), (p_2^g, p_2^t))$ of sequence-compatible routings is interference-compatible if $p_1^g(\theta) \leq p_2^g(\theta) - 1$ for each $\theta \in [0, \Theta]$.
- For a crossover crane system, we assume crane 1 to be the larger crane. Then, a pair $((p_1^g, p_1^t), (p_2^g, p_2^t))$ of sequence-compatible routings is interference-compatible if $|p_1^g(\theta) - p_2^g(\theta)| \geq 1$ or θ is not in an interval of a visit associated with a task of crane 1 for each $\theta \in [0, \Theta]$. Hence, interference can occur only during visits associated with tasks of crane 1. During these visits, the gantries are required to maintain a distance of at least one bay.

Example In order to illustrate the concepts introduced so far, we consider a small example concerning twin cranes with $(b_1^0, r_1^0) = (3, 1)$, $(b_2^0, r_2^0) = (8, 4)$, $s^t = 1$, $\sigma_1 = (a, b)$ and $\sigma_1 = (c, d)$, $B = 8$, $R = 5$ and the task properties given in Table 2.

Now, a feasible solution is an interference-compatible pair of routings exemplified in the following. Crane 1 moves from $(b_1^0, r_1^0) = (3, 1)$ to $(5, 4)$, which takes three time units, waits one time unit for $e_a = 4$ and conducts a for three time units, moves to $(3, 4)$, which takes two time units and stays for one time unit, moves to $(4, 4)$, which takes one time unit and stays for one time unit, moves to $(6, 4)$, which takes two time units, and conducts b for two time units. Crane 2 moves from $(b_2^0, r_2^0) = (8, 4)$ to $(7, 4)$ which takes one time unit, waits for one time unit, moves to $(6, 4)$ which takes one time unit and stays for four time units, moves to $(5, 4)$ which takes one time unit and conducts c for four time units, and moves to

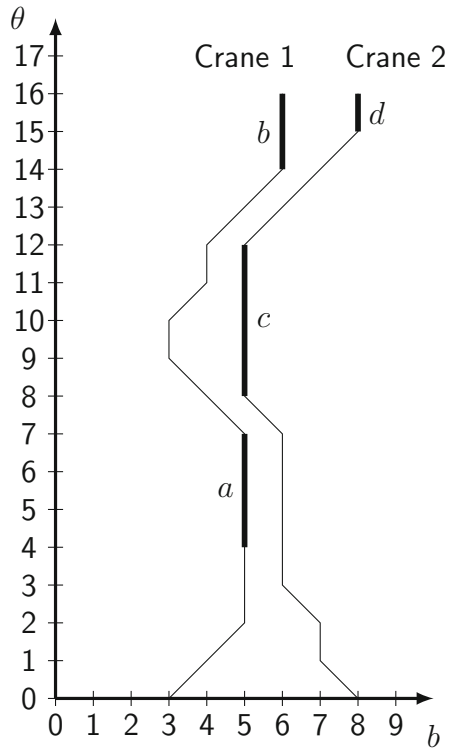

Fig. 2 Example solution

$(8, 4)$ which takes three time units and conducts d for one time unit.

Figure 2 depicts the gantry positions of both cranes over time. The bold parts of the lines represent tasks being conducted. Note that crane 1 needs the first time unit after arriving in bay 5 for adjusting the trolley to row 4. For all other moves, the crane's trolley has reached the row when the gantry reaches the bay. Crane 1 has to wait one more time unit before conducting a due to $e_a = 4$.

For a feasible solution and the implied completion time C_j of each task, we consider two objective functions $\max\{f_j(C_j) \mid j \in J\}$ and $\sum_{j \in J} f_j(C_j)$ which are both to be minimized.

In the following, we address four problem settings (and some special cases), distinguishing between twin cranes and crossover cranes as well as the objectives of minimization of maximum cost among tasks and minimization of total cost. We refer to these problems as Twin-Max, Crossover-Max, Twin-Sum, and Crossover-Sum, respectively. Furthermore, we consider a decision version of each of these problems asking whether there is a solution with an objective value not exceeding a given threshold.

The first question we are going to address is whether the decision versions are members of NP. In order to show this, we will consider a shorter certifier. Instead of functions p_c^g and p_c^t, we consider two sequences of tuples

$$((\theta^g_{c,1}, b^g_{c,1}), \ldots, (\theta^g_{c,n_1^b}, b^g_{c,n_1^b}) \text{ and } ((\theta^t_{c,1}, b^t_{c,1}), \ldots, (\theta^t_{c,n_1^t}, b^t_{c,n_1^t})$$

specifying points of time when the gantry and the trolley of crane c reach a certain bay and row, respectively, and stay there for some positive duration. Since gantry speeds and trolley speeds are given, two consecutive tuples are implied when the first position is left.

Lemma 1 *The decision versions of Twin-Max, Crossover-Max, Twin-Sum, and Crossover-Sum are members of NP.*

Proof Clearly, sequences of tuples as suggested fully specify a routing if the time difference of two consecutive tuples allows us to cover the distance. Hence, we can check sequence compatibility of the routings and also the feasibility of a solution composed of the two routings in a time polynomial in the length of the sequences.

Now, it suffices to argue that the length of the sequences can be bounded to be polynomial in the input size of the problem instance. First, note that the trolley of crane c can move independently from the gantry of c and both the gantry and trolley of crane $3 - c$. Hence, we cannot do anything wrong by moving the trolley of each crane to the row of the next task immediately after completing a task. Thus, we can restrict ourselves to trolley sequences with a length of at most n_c for crane c. Second, note that the gantry of crane c needs to go to a certain bay (and stay there for some time) only if c has a task to conduct in this bay or if c avoids crane $3 - c$ while $3 - c$ conducts a task. Between these bay visits, there is no need for any further visits. Hence, we can restrict ourselves to gantry sequences with a length of at most $n_1 + n_2$. □

While the result in Lemma 1 itself does not come as a surprise, the representation of a routing as a pair of sequences with bounded length offers a particular perspective on solutions of our problem. In the following, we focus on solutions where the planning horizon is separated into time intervals such that

- in the last interval (potentially of length 0), both cranes have completed their respective task sequence;
- in the kth interval, k odd (potentially of length 0), both cranes are processing their task sequences in parallel; and
- in the kth interval, k even, only one crane c is processing its task sequence while $3 - c$ is not and idles or avoids c for c to complete a task.

Here, we say that a crane processes its task sequence if it conducts a task or moves towards its next task's position. As soon as it delays the start of its next task, it does not process its task sequence. Note that we can restrict ourselves

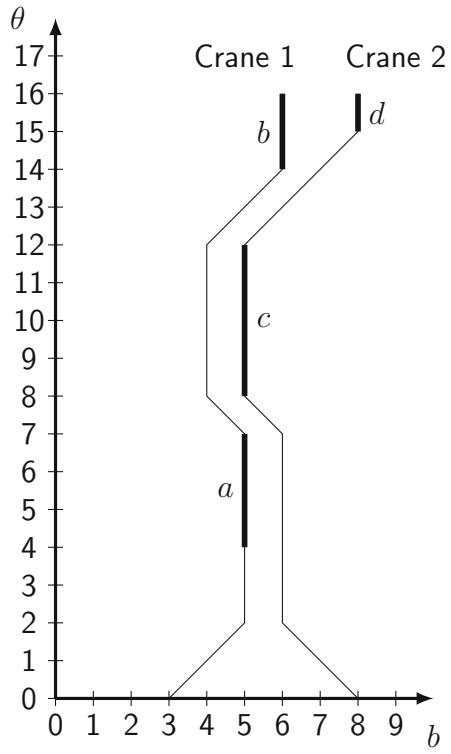

Fig. 3 Example solution without additional visits

to routings where a crane c stops processing its task sequence only if it is prevented from doing so by the other crane $3-c$. Once crane $3-c$ is no longer preventing this action, crane c proceeds. We can assume that crane c is located in a bay next to the one in which crane $3-c$ is located, or even in the same bay (if feasible), at that moment because it minimizes the distance to the next task's bay.

Example (cont.) We take up the example above and consider the gantry positions of both cranes over time as depicted in Fig. 2. Figure 3 depicts gantry positions over time in a different solution, which has the same completion times of tasks as that in Fig. 2 but fits the new perspective, and thus does not incorporate any additional visits.

In the solution sketched in Fig. 3, both cranes process their task sequences for two time units. Then, only crane 1 processes its task sequence for five time units (including adjustment of trolley to row 4 and waiting for $e_a = 4$). Meanwhile, crane 2 waits in the neighboring bay 6. Afterwards, both cranes process their sequences for zero time units before crane 1 avoids crane 2, and only crane 2 processes its sequence for six time units until it reaches bay 6. In the meantime, crane 1 avoids crane 2, moves to the neighboring bay 4, and returns to bay 5 as soon as possible. Then, both cranes complete their sequences in parallel.

From now on we thus assume that at any point of time, each crane c has already completed its task sequence, is processing it, or is prevented from doing so by crane $3-c$. In the latter case, there is a task $j \in \sigma_{3-c}$ such that c is prevented from processing its task sequence up to the completion of j. If we consider twin cranes, we assume that c waits in bay $b_j - 3 + 2c$, that is, in $b_j - 1$ if $c = 1$ and in $b_j + 1$ if $c = 2$. If we consider crossover cranes, we assume that crane 1 waits in b_j, and crane 2 waits either in $b_j - 1$ or in $b_j + 1$ depending on whether crane 2 has been operating in a smaller or in a larger bay before. This perspective is close to the one taken by Briskorn and Angeloudis (2016) and Eilken (2019).

3 General problem settings

In this section, we address the general problem settings. We show NP-hardness of Crossover-Sum in Sect. 3.1 explicitly and conclude NP-hardness of Twin-Sum. In Sect. 3.2, we propose a DP framework that enables us to solve Crossover-Sum and Twin-Sum in pseudo-polynomial time, and Crossover-Max and Twin-Max in polynomial time.

3.1 NP hardness of Crossover-Sum and Twin-Sum

We will focus on the special case of Crossover-Sum with $R = 1$ and prove it to be NP-hard. For the sake of convenience, we thus address positions by bay numbers only in Sect. 3.1. For this special case, Briskorn and Angeloudis (2016) introduce a graphical model representing solutions. We will use this model to clarify the reduction used in our proof and, in order to keep the paper self-contained, briefly describe the model. Let T_1 and T_2 be the time spans required by cranes 1 and 2, respectively, to complete their task sequences, including moving durations between tasks if no interference occurs. The model encompasses a rectangular area with width T_1 and height T_2. Each point (t_1, t_2) in the rectangular area represents a state where cranes 1 and 2 have completed the first t_1 and t_2 time units of their task sequence, respectively.

Note, however, that some points might be infeasible with respect to crane interference. We have a rectangular cluster of infeasible points for each pair comprising a task of crane 1 in a bay b and the presence of crane 2 in b (including entering b, leaving b, and conducting a task in b).

Consider the following example with cranes 1 and 2 initially located in bays 0 and 1, respectively, and task sequences $\sigma_1 = (1, 2, 3)$ and $\sigma_2 = (4, 5, 6)$ with positions $b_1 = 3$, $b_2 = 4$, $b_3 = 6$, $b_4 = 5$, $b_5 = 3$, and $b_6 = 1$ and durations $p_2 = 2$ and $p_1 = p_3 = p_4 = p_5 = p_6 = 1$. Then, we have $T_1 = 10$ and $T_2 = 11$.

Among the three tasks of crane 1, only the first two raise the potential for interference, since crane 2 never moves to bay $b_3 = 6$ according to its task sequences. The two left clusters depicted in Fig. 4 correspond to task $j = 1$ in bay $b_1 = 3$ with duration $p_1 = 1$. Consequently, they have width

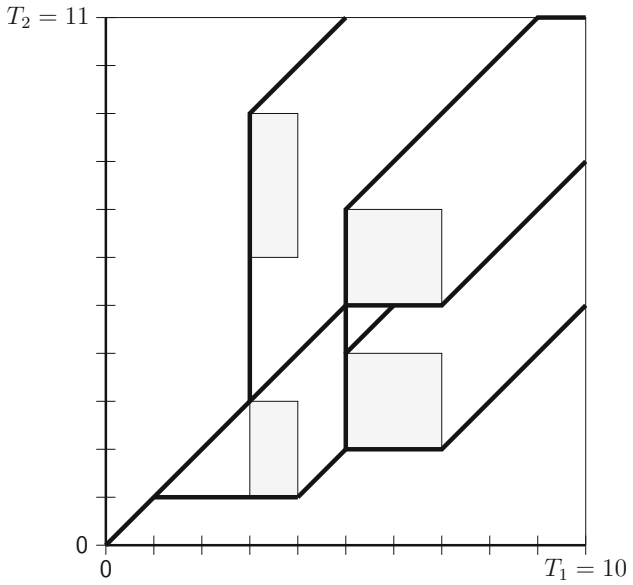

Fig. 4 Example model

$p_1 = 1$ and correspond to crane 2 crossing bay 3 before conducting task 4 and task 5 in bay $b_5 = b_1 = 3$. The right two clusters correspond to task $j = 2$ in bay $b_2 = 4$ with duration $p_2 = 2$, and thus have width $p_2 = 2$ and correspond to crane 2 crossing bay 4 twice.

As shown in Briskorn and Angeloudis (2016), these clusters accurately specify the set of infeasible points. A solution corresponds to a path from the lower left corner of the area to the upper right corner of the area. A path consists of lines that can only run horizontally from left to right, vertically upwards, or diagonally upwards from left to right, and must not cross through obstacles. Horizontal sections, vertical sections, and diagonal sections correspond to time intervals where only crane 1 processes its task sequence, only crane 2 processes its sequence, or both cranes process their sequences in parallel, respectively. In Fig. 4, several such paths are outlined in bold. The bottommost path represents both cranes processing their task sequences for one time unit, then only crane 1 processing its sequence for three time units (including conducting task 1 in bay 3) while crane 2 waits in bay 2, then again both cranes processing their sequences for one time unit, crane 2 waiting in bay 3 while crane 1 conducts task 2 in bay 4, and finally, both cranes completing their task sequences.

We will prove the hardness of Crossover-Sum by reduction from PARTITION, which is known to be binary NP-complete.

PARTITION:

Given m integers $a_1, \ldots, a_m$ with total value $2D$, is there a subset A of these numbers with total value of D?

Lemma 2 *Crossover-Sum is NP-hard.*

Proof We consider a particular cost function $f_j(C_j)$ for each task $j \in J$. Each task j is associated with a due date d'_j and a weight w_j, and we have

$$f_j(C_j) = \begin{cases} 0 & \text{if } C_j \leq d'_j \\ w_j & \text{else} \end{cases},$$

and thus our objective is the classic minimization of the weighted number of tardy tasks.

We start by introducing a structure which represents the decision of whether to have number a_k in subset A by means of Crossover-Sum. We refer to the structure as switch in the following. We consider two pairs of tasks k_1 and k_2 as tasks k'_1 and k'_2 which are consecutive tasks in the sequences of cranes 1 and 2. The positions of these tasks (neglecting rows) are $b_{k_1} = b_{k_2} = b$ and $b_{k'_1} = b_{k'_2} = b + 2$ for a $b \in \{0, \ldots, B-1\}$.

- For the time being, we assume that larger crane 1 reaches b at the same time point as crane 2 reaches bay $b-1$ (we will justify this assumption later). This point of time depends on the routings up to this point. Hence, it is not constant, but we assume it to be bounded from below by $\underline{\theta_k}$ and from above by $\overline{\theta_k}$. Now, the decision must be made as to which crane can go first. We have $p_{k_1} = \overline{\theta_k} - \underline{\theta_k} + a_k$, $d'_{k_1} = \infty$, $w_{k_1} = 0$, $p_{k_2} = a_k - 2$, $d'_{k_2} = \overline{\theta_k} + a_k - 1$, and $w_{k_2} = a_k$. Note that completion of k_2 will be non-tardy if and only if crane 2 goes first, and we are indifferent as to whether to let crane 1 go first with respect to $f_{k_1}(C_{k_1})$, since $w_{k_1} = 0$.
- For the two next tasks k'_1 and k'_2, we have $p_{k'_1} = a_k - 2$, $d'_{k'_1} = 2 \cdot \overline{\theta_k} - \underline{\theta_k} + 3 \cdot a_k$, $w_{k'_1} = \infty$, $p_{k'_2} = d'_{k'_1} - \underline{\theta_k}$, $d'_{k'_2} = \infty$, and $w_{k'_2} = 0$. Again, we must decide which crane can go first. Note that completion of k'_1 will be non-tardy if and only crane 1 conducts k'_1 before crane 2 conducts k'_2, and this does not depend on the priority of cranes with regard to k_1 and k_2. In order to meet a given finite upper bound for the overall objective value, k'_1 indeed needs to be conducted first due to $w_{k'_1} = \infty$.

In a solution meeting a finite upper bound, the decision of whether to give crane 1 or crane 2 priority with respect to k_1 and k_2 is open, but crane 1 necessarily is given priority with respect to k'_1 and k'_2, and crane 2 is waiting in bay $b+1$ when crane 1 completes k'_1. Then, if crane 2 is given priority with respect to k_1 and k_2,

- the total weight of tardy tasks among k_1, k_2, k'_1, and k'_2 is zero, and

- the time span between cranes 1 and 2 reaching bays b and $b-1$, respectively, and crane 1 completing k'_1 is

$$\underbrace{1 + p_{k_2} + 1}_{\text{in } b,\, k_2,\, \text{out of } b} + \underbrace{p_{k_1} + 2 + p_{k'_1}}_{k_1,\, \text{move to } b+2,\, k'_1}$$
$$= a_k + (\overline{\theta_k} - \underline{\theta_k} + a_k) + 2 + (a_k - 2) = 3 \cdot a_k + (\overline{\theta_k} - \underline{\theta_k}).$$

  Here, the left part reflects the length of time that crane 1 is delayed from starting k_1 by crane 2, and the right part reflects the time crane 1 needs to complete k'_1 after that.

However, if crane 1 is given priority with respect to k_1 and k_2,

- the total weight of tardy tasks among k_1, k_2, k'_1, and k'_2 is $w_{k_2} = a_k$, and
- the time span between cranes 1 and 2 reaching bays b and $b-1$, respectively, and crane 1 completing k'_1 is

$$p_{k_1} + 2 + p_{k'_1} = (\overline{\theta_k} - \underline{\theta_k} + a_k) + 2 + (a_k - 2)$$
$$= 2 \cdot a_k + (\overline{\theta_k} - \underline{\theta_k})$$

  Here, crane 1 can start k_1 immediately, and crane 2 can conduct k_2 and move to $b+1$ while crane 1 conducts k'_1.

In summary, the switch lets us decide between two options, where the first implies a total weight of involved tardy tasks that is lower by a_k but implies a partial solution that is longer by a_k. We depict this switch by means of the graphical model in Fig. 5.

We see four paths in Fig. 5 leading below and above the left and the right clusters in which cranes 1 and 2 are given priority, respectively. Only the two black paths leading below the right cluster can lead to a solution with a finite objective value. The upper path among the two is longer by a_k but has k_2 as non-tardy. The two gray paths lead to k'_1 being tardy.

We now show how to combine such switches, one per number in the instance of PARTITION, in order to represent the instance of PARTITION. We argue using the graphical model depicted in Fig. 6.

We have a switch like the one described before for each number in the instance of PARTITION. For each number a_k, we have a switch where we can decide whether to complete k_2 as non-tardy, avoiding the extra weight of a_k of a tardy task and paying the price of a_k in extra length of the routing up to this point. Note that we can decide whether to complete k_2 as non-tardy independently of our decisions regarding $a_1, \ldots, a_{k-1}$, since we can account for all possible resulting routing lengths by having $\overline{\theta_k} - \underline{\theta_k} \geq \sum_{k'=1}^{k-1} a_{k'}$.

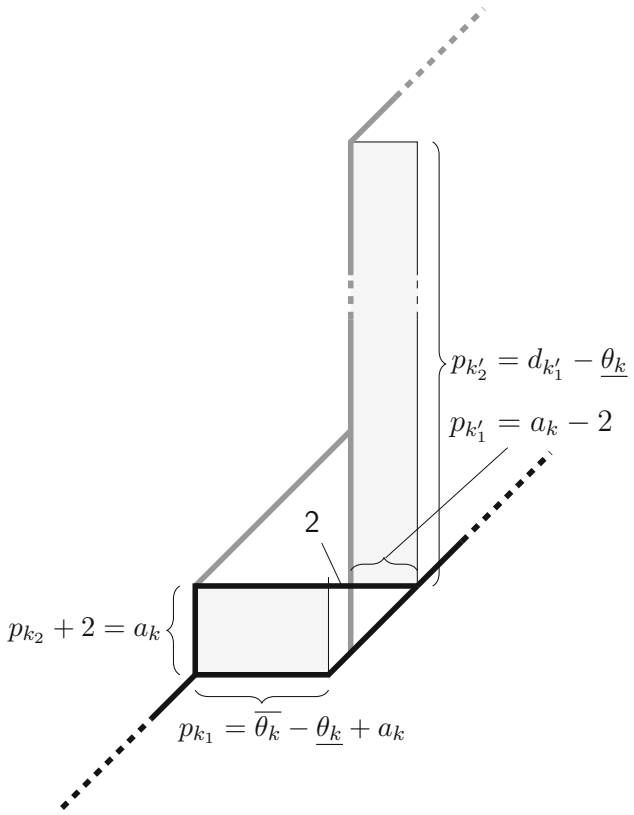

Fig. 5 Switch

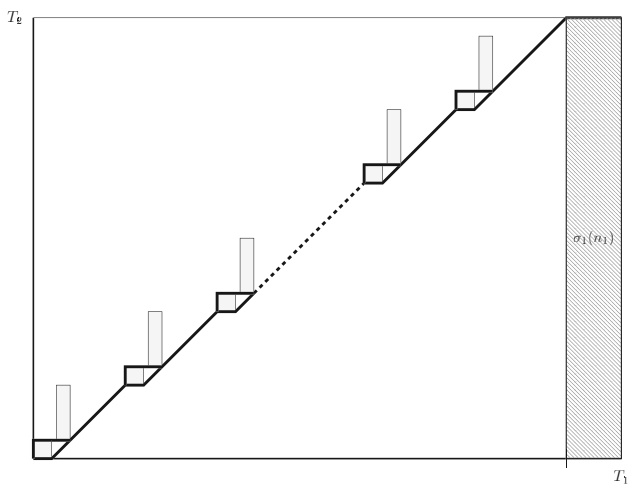

Fig. 6 Model obtained by reduction from the PARTITION instance

It remains to be discussed how switches corresponding to a_k and a_{k+1} can be geared to each other such that cranes 1 and 2 in fact reach $b = b_{(k+1)_1} = b_{(k+1)_2}$ and $b-1$ simultaneously (as we assumed when outlining the switch above). We consider a third task of crane 1 per switch which has the same duration as k'_2, location $b_{k'_1}+1$, weight zero, and due date ∞. Crane 1 then is busy in $b_{k'_1}+1$ for the same time as crane 2 is busy with k'_2 in $b_{k'_2} = b_{k'_1}$, and cranes 1 and 2 can

proceed simultaneously to $b = b_{k'_1} + 2$ and $b - 1 = b_{k'_1} + 1$, respectively, immediately afterwards.

So far, we have used $5m$ tasks in total for designing a switch for each number in the instance of PARTITION. A lower bound LB on the length of feasible routings completing these $5m$ tasks can be given as the duration it takes crane 1 to complete its $3m$ tasks without any delay caused by interference. A feasible routing for these $5m$ tasks having length LB can be visualized as the path in Fig. 6 where crane 1 is given priority at each cluster of infeasible points up to the shaded area. This routing corresponds to a total weight of tardy tasks of $2D$, since each task k_2, $k = 1, \ldots, m$ is tardy. Furthermore, for an arbitrary subset A of numbers in the PARTITION instance, we have a partial routing where k_2 is non-tardy if and only if $k \in A$ with length $LB + \sum_{k \in A} a_k$ and total weight of tardy tasks of $\sum_{k \notin A} a_k$.

Now we add one last task to sequence σ_1 with a weight of $D+1$ which can be non-tardy only if the routing completing the first $5m$ tasks has length of at most $LB + D$; that is, we set its due date to $d'_{\sigma_1(n_1)} = T_1 + D$. The question then is whether there is a solution with a total weight of tardy tasks of at most D. We claim that the answer to this question is yes if and only if the answer to the instance of PARTITION is yes.

First, for an arbitrary feasible solution, consider the subset $A \subset \{1, \ldots, m\}$ with $k \in A$ if and only if k_2 is non-tardy. If the total weight of tardy tasks does not exceed D, then $\sum_{k \in A} a_k \geq D$ (since otherwise the total weight of tardy tasks in $\{k_2 \mid k = 1, \ldots, m\}$ exceeds D) and $\sum_{k \in A} a_k \leq D$ (since otherwise the last task in σ_1 with weight $D + 1$ is tardy). Hence, the answer to the instance of PARTITION is yes if we can achieve a total weight of tardy tasks of at most D.

In Fig. 6, task $\sigma_1(n_1)$ or rather the points corresponding to conducting $\sigma_1(n_1)$ are highlighted as the shaded area. Note that this area does not constitute a cluster of infeasible points. It becomes clear that T_1 is a lower bound for the completion time of $\sigma_1(n_1)$. Since the due date of $\sigma_1(n_1)$ is set to $T_1 + D$, each path with $\sigma_1(n_1)$ being non-tardy can have a total height of vertical segments of at most D.

Second, for a yes instance of PARTITION and corresponding subset A of numbers, the routing with k_2 being non-tardy occurs if and only if $k \in A$ has a total weight of tardy tasks of D. This completes the proof. □

Lemma 3 *Twin-Sum is NP-hard.*

We omit a formal proof in order to reduce the notational burden. However, note that the routing enforced in the reduced instance in the proof of Lemma 2 closely resembles the routing that could be implemented by twin cranes. While crossover cranes are allowed to pass each other, we can see that for each of the considered routings of crossover cranes for the reduced instance there is a routing having the same completion time for each task where crane 1 is located in a larger bay than crane 2 at each time point. This might require crane 1 to avoid crane 2 if crane 2 is given priority (and conducts k_2 before crane 1 conducts k_1), which however does not delay any completion time. Hence, we can use the same task sequences as in the proof of Lemma 2 in a similar proof for Twin-Sum.

3.2 General dynamic programming framework

We develop a DP framework for each of the considered problems. In order to achieve unifying phrasing, we distinguish between different crane settings as little as possible and rely on the structure of solutions as outlined at the end of Sect. 2.2.

We consider a state (k_1, k_2, c, θ) for each $k_c = 1, \ldots, n_c$, $c \in \{1, 2\}$, and $\theta \in \{0, \ldots, \Theta\}$. State (k_1, k_2, c, θ) represents crane c having just completed task $\sigma_c(k_c)$ at time point θ, and crane $3 - c$ having completed $\sigma_{3-c}(k_{3-c})$ last and waiting for crane c (to complete $\sigma_c(k_c)$). Hence, the implied position of crane c is $(b_{\sigma_c(k_c)}, r_{\sigma_c(k_c)})$. Depending on the crane system under consideration, state (k_1, k_2, c, θ) also implies or at least restricts the possible gantry and trolley positions of crane $3-c$ at time θ.

- If we consider twin cranes, then the gantry of crane $3 - c$ is located in bay $b = b_{\sigma_c(k_c)} + 3 - 2c$. The trolley of crane $3 - c$ has a position in $[r_{\sigma_{3-c}(k_{3-c}+1)} - |b - b_{\sigma_{3-c}(k_{3-c}+1)}|, r_{\sigma_{3-c}(k_{3-c}+1)} + |b - b_{\sigma_{3-c}(k_{3-c}+1)}|]$. Note that any trolley position in this interval allows an adjustment of the trolley to the next task's row $r_{\sigma_{3-c}(k_{3-c}+1)}$ while moving the gantry to the next task's bay $b_{\sigma_{3-c}(k_{3-c}+1)}$. Thus, if the trolley is in a position in this interval, then the start of the next task of crane $3 - c$ is delayed by crane c. If, however, the trolley is not in a position in this interval, then while the gantry of crane $3 - c$ is prevented from approaching the next task's bay, crane $3 - c$ can use this delay of the gantry (at least partially) to adjust its trolley.
- If we consider crossover cranes, we distinguish between $c = 1$ and $c = 2$.
  - If $c = 2$, then crane $3 - c = 1$ is located in bay $b_{\sigma_c(k_c)}$, and its trolley has a position in $[r_{\sigma_{3-c}(k_{3-c}+1)} - 1, r_{\sigma_{3-c}(k_{3-c}+1)} + 1]$.
  - If $c = 1$, then crane $3 - c = 2$ is located in bay $b_{\sigma_c(k_c)} - 1$ or $b_{\sigma_c(k_c)} + 1$ if $b_{\sigma_{3-c}(k_{3-c})} < b_{\sigma_c(k_c)}$ or $b_{\sigma_{3-c}(k_{3-c})} > b_{\sigma_c(k_c)}$, respectively. The trolley of crane $3 - c$ has a position in $[r_{\sigma_{3-c}(k_{3-c}+1)} - |b - b_{\sigma_{3-c}(k_{3-c}+1)}|, r_{\sigma_{3-c}(k_{3-c}+1)} + |b - b_{\sigma_{3-c}(k_{3-c}+1)}|]$.

In any case, the trolley position may not be fully defined here, but it is narrowed such that it does not delay conducting $\sigma_{3-c}(k_{3-c}+1)$ any further once c gives way to $3-c$. Note that

the states considered in our DP are similar to those proposed for twin cranes in Eilken (2019).

Furthermore, we have an initial state $(0, 0, 0, 0)$ representing each crane being in its initial position at the beginning of the planning horizon, a state $(k_1, 0, 1, \theta)$ for each $k_1 = 1, \ldots, n_1$ and $\theta \in \{0, \ldots, \Theta\}$ representing crane 2 waiting for crane 1 to complete k_1 before starting its first task, and a state $(0, k_2, 2, \theta)$ for each $k_2 = 1, \ldots, n_2$ and $\theta \in \{0, \ldots, \Theta\}$ representing crane 1 waiting for crane 2 to complete k_2 before starting its first task. Finally, we have a final state $(n_1, n_1, 0, \theta)$ for each $\theta \in \{0, \ldots, \Theta\}$ representing both task sequences to be completed.

There is a transition from (k_1, k_2, c, θ) to $(k'_1, k'_2, c', \theta')$ if

- $k'_{c'} > k_{c'}$ and $k'_{3-c'} \geq k_{3-c'}$ and
- there is an interference-compatible pair of partial routings starting from (k_1, k_2, c, θ) where
  - crane c' processes its task sequence up to completing $\sigma_{c'}(k'_{c'})$ without any delay caused by interference with crane $3 - c'$ and completes $\sigma_{c'}(k'_{c'})$ at θ',
  - crane $3 - c'$ processes its task sequence up to completing $\sigma_{3-c'}(k'_{3-c'})$ without any delay caused by interference with crane c', and
  - crane $3 - c'$ gets into the position implied by $(k'_1, k'_2, c', \theta')$ prior to θ'.

Note that we do not consider a crane to be delayed by interference here if its next task is not delayed. For example, if a crane's gantry is prevented from approaching the next task's bay for some time but still reaches the next task's position before its release date, we do not consider the crane to be delayed by interference.

Furthermore, there is a transition from (k_1, k_2, c, θ) to final state $(n_1, n_1, 0, \theta')$ if there is an interference-compatible pair of partial routings starting from (k_1, k_2, c, θ) where both cranes complete their task sequences without any delay caused by interference.

Hence, states represent situations where both cranes have completed their tasks or where a crane has prevented the other crane from proceeding and the other crane is just about to resume processing its task sequence. Transitions, then, connect two such situations if there is no third such situation in between. Note that a transition corresponds to two consecutive time intervals in a solution, as discussed towards the end of Sect. 2.2.

Note that we can check in $O(|J|)$ time whether a transition from (k_1, k_2, c, θ) to $(k'_1, k'_2, c', \theta')$ exists by scanning through task sequences and considering release dates of tasks, moving times for gantries and trolleys, and deadlines. Note also that since θ is given, we can determine the completion times of all tasks completed during this transition. This allows us to, first, check whether one of these tasks misses its deadline, which means the partial routing is not sequence-compatible, and second, determine the total cost of these tasks and the maximum cost among these tasks, respectively, depending on the objective function considered.

We evaluate each state (k_1, k_2, c, θ) by $F(k_1, k_2, c, \theta)$, reflecting the total cost of tasks $\sigma_1(1)$ to $\sigma_1(k_1)$ and $\sigma_2(1)$ to $\sigma_2(k_2)$ or the maximum cost among these tasks depending on the objective function considered.

Initial state $(0, 0, 0, 0)$ has $F(0, 0, 0, 0) = 0$, while for each of the other states (k_1, k_2, c, θ) the Bellman function is

$$F(k_1, k_2, c, \theta) = \min \big\{ F(k'_1, k'_2, c', \theta') + c^{sum}((k'_1, k'_2, c', \theta'), (k_1, k_2, c, \theta)) \mid (k'_1, k'_2, c', \theta') \in P(k_1, k_2, c, \theta) \big\}$$

for Crossover-Sum and Twin-Sum and

$$F(k_1, k_2, c, \theta) = \min_{(k'_1, k'_2, c', \theta') \in P(k_1, k_2, c, \theta)} \big\{ \max \big\{ F(k'_1, k'_2, c', \theta'), c^{\max}((k'_1, k'_2, c', \theta'), (k_1, k_2, c, \theta)) \big\} \big\}$$

for Crossover-Max and Twin-Max, where

- $P(k_1, k_2, c, \theta)$ is the set of states that have a transition leading to (k_1, k_2, c, θ),
- $c^{sum}((k'_1, k'_2, c', \theta'), (k_1, k_2, c, \theta))$ is the total cost of tasks $\sigma_1(k'_1 + 1)$ to $\sigma_1(k_1)$ and $\sigma_2(k'_2 + 1)$ to $\sigma_2(k_2)$, and
- $c^{\max}((k'_1, k'_2, c', \theta'), (k_1, k_2, c, \theta))$ is the maximum cost among tasks $\sigma_1(k'_1+1)$ to $\sigma_1(k_1)$ and $\sigma_2(k'_2+1)$ to $\sigma_2(k_2)$.

The problem then is to determine

$$\min \{F(n_1, n_2, 0, \theta) \mid \theta \in \{0, \ldots, \Theta\}\}.$$

We have $O(|J|^2 \cdot \Theta)$ states, since we can restrict ourselves to integer values of θ for each state (k_1, k_2, c, θ) because all the time parameters are integers. Furthermore, we have $O(|J|)$ transitions starting from each state. To see this, we can imagine that starting from a state (k_1, k_2, c, θ) we choose c' and $k'_{c'}$ of the state to be reached. Note that this implies that θ' of the state is reached, since the transition represents a partial routing where c' processes its task sequence without any delay by interference with crane $3 - c'$. For chosen c' and $k'_{c'}$, we furthermore can choose $k'_{3-c'}$ to be maximum under the restriction that a transition to $(k'_1, k'_2, c', \theta')$ exists. Summarizing the above, we have a total number of $O(|J|^3 \cdot \Theta)$ transitions. We can determine the total cost of tasks and the maximum cost among tasks associated with the transition in $O(|J|)$ time by scanning through task sequences. This leaves us with an overall effort of $O(|J|^4 \cdot \Theta)$ and leads to the following results.

Lemma 4 *Twin-Max, Crossover-Max, Twin-Sum, and Crossover-Sum can be solved in* $O(|J|^4 \cdot \Theta)$ *time.*

Lemmas 2, 3, and 4 imply the following theorem.

Theorem 1 *Twin-Sum and Crossover-Sum are binary NP-hard.*

Next, we turn our attention to the problem of determining a feasible solution. Obviously, we can use the DP approach developed above with an arbitrary objective function, but we instead refine it in order to derive a more efficient approach. Using $f_j(C_j) = C_j$ for each $j \in J$ and objective function $\max\left\{f_j(C_j) \mid j \in J\right\}$, we obtain $F(k_1, k_2, c, \theta) = \theta$ for each state (k_1, k_2, c, θ). Hence, we can restrict ourselves to a state specification (k_1, k_2, c) for each $k_c = 0, \ldots, n_c$ and $c \in \{0, 1\}$, and let θ be implied by $F(k_1, k_2, c)$. Note that we can then determine and evaluate transitions starting from a state (k_1, k_2, c) only after evaluating (k_1, k_2, c) itself.

We have $O(|J|^2)$ states, $O(|J|)$ transitions starting from each state, and thus $O(|J|^3)$ transitions in total. We can determine whether a particular transition exists in $O(|J|)$ time by scanning through task sequences. This leaves us with an overall effort of $O(|J|^4)$ to find a feasible solution.

Now, it is not hard to see that we can employ this refined DP approach in a binary search scheme (adjusting deadlines in order to bound the maximum cost among tasks from above) which yields the following result.

Theorem 2 *Twin-Max and Crossover-Max can be solved in* $O(|J|^4 \cdot \log(\Theta))$ *and, thus, in polynomial time.*

4 Special cases of Crossover-Sum and Twin-Sum

In this section, we consider special cases of Crossover-Sum and Twin-Sum which can be solved in strongly polynomial time.

4.1 Number of tardy tasks

We consider a particular cost function $f_j(C_j)$ for each task $j \in J$. Each task j is associated with a due date d'_j, and we have

$$f_j(C_j) = \begin{cases} 0 & \text{if } C_j \leq d'_j \\ 1 & \text{else} \end{cases},$$

and thus our objective is the classic minimization of the number of tardy tasks. Obviously, we can use the approach developed in Sect. 3.2 for Twin-Sum and Crossover-Sum which enables us to solve the problem in $O(|J|^4 \cdot \Theta)$ time. However, we will develop a strongly polynomial, and thus more efficient, DP approach in the following.

First, we extend the previous state specifications and explicitly maintain the number of tardy tasks in the state definition here. Hence, we consider a state (k_1, k_2, c, θ, u) for each $k_c = 0, \ldots, n_c$, $c \in \{0, 1\}$, $\theta \in \{0, \ldots, \Theta\}$, and $u = 0, \ldots, |J|$. Here, u represents the number of tardy tasks among the first k_1 and k_2 tasks of cranes 1 and 2, respectively.

We then take up the approach to find a feasible solution proposed in Sect. 3.2 and use $f'_j(C_j) = C_j$ for each $j \in J$ and objective function $\max\left\{f'_j(C_j) \mid j \in J\right\}$. Hence, we consider a state (k_1, k_2, c, u) for each $k_c = 0, \ldots, n_c$, $c \in \{0, 1\}$, and $u = 0, \ldots, |J|$, and let θ be implied by $F(k_1, k_2, c, u)$.

Again, we can determine and evaluate transitions starting from a state (k_1, k_2, c, u); then, only after evaluating (k_1, k_2, c, u) itself, we have $O(|J|^3)$ states and $O(|J|)$ transitions starting from each state, which implies $O(|J|^4)$ transitions in total. For given (k_1, k_2, c, u), c', and $k'_{c'}$, we can determine $k'_{3-c'}$ and the number of tardy tasks implied, and hence the resulting state (k'_1, k'_2, c', u'), in $O(|J|)$ time. This leaves us with an overall effort of $O(|J|^5)$ and leads to the following results.

Theorem 3 *We can find a feasible solution with a minimum number of tardy tasks in* $O(|J|^5)$ *time.*

4.2 Total weighted completion time without release dates and deadlines

In this section, we consider a problem setting where tasks have no (restricting) release dates or deadlines; that is, we have $r_j = 0$ and $d_j = \infty$ for each task $j \in J$. Furthermore, we consider a particular cost function. Each task j is associated with a weight w_j, and we have $f_j(C_j) = w_j C_j$, and thus our objective is the classic minimization of total weighted completion time. Note that a special case of this objective function with $w_{\sigma_1(n_1)} = w_{\sigma_2(n_2)} = 1$ and $w_j = 0$ for each other task j reflects the total time that cranes are busy, and has been considered, for example, in Dorndorf and Schneider (2010) for triple cranes and in Speer and Fischer (2017). Again, the approach developed in Sect. 3.2 is available. However, we will derive a strongly polynomial DP approach in the following.

We will distinguish two parts of the completion time of each task $j \in \sigma_c$, $c \in \{1, 2\}$. The first part is implied by the task sequence σ_c. It accounts for the total driving duration prior to j if no interference occurs between cranes and the total processing time of the preceding tasks and j itself. Note that this part is constant and therefore will not be considered in our DP approach. The second part accounts for the total duration that crane c does not process σ_c before j is completed. We refer to this part of the completion time as the delay of task j. Our DP approach focuses on minimizing the total weighted delay of tasks, which is equivalent to minimizing the total weighted completion time.

We consider a state (k_1, k_2, c) for each $k_c = 0, \ldots, n_c$ and $c \in \{0, 1\}$. Similar to the approaches described earlier, state (k_1, k_2, c) represents crane c having just completed $\sigma_c(k_c)$ and crane $3-c$ having completed $\sigma_{3-c}(k_{3-c})$ last and waiting for crane c (to complete $\sigma_c(k_c)$). Note that, as opposed to the more general approach in Sect. 3.2, the point of time is not specified here. Furthermore, we have an initial state $(0, 0, 0)$, a state $(k_1, 0, 1)$ for each $k_1 = 1, \ldots, n_1$, and a state $(0, k_2, 2)$ for each $k_2 = 1, \ldots, n_2$. Finally, we have a final state $(n_1, n_1, 0)$ representing both cranes having completed their task sequences.

We have a transition from (k_1, k_2, c) to (k'_1, k'_2, c') if

- $k'_{c'} > k_{c'}$ and $k'_{3-c'} \geq k_{3-c'}$, and
- there is an interference-compatible pair of partial routings starting from (k_1, k_2, c) where
  - crane c' processes its task sequence up to completing $\sigma_{c'}(k'_{c'})$ without any delay,
  - crane $3 - c'$ processes its task sequence up to completing $\sigma_{3-c'}(k'_{3-c'})$ without any delay, and
  - crane $3 - c'$ gets into the position implied by $(k'_1, k'_2, c', \theta')$ prior to the completion of $k'_{c'}$.

Note that we can actually decide whether such an interference-compatible pair of partial routings exists without specifying a start time (again, as opposed to the more general approach in Sect. 3.2) since neither release dates nor deadlines need to be considered. Note that for the same reason, delays can only be caused by crane interference.

We evaluate each state (k_1, k_2, c) by $F(k_1, k_2, c)$ representing the total weighted delay of tasks raised so far. Note that the total weighted delay of tasks implied by a transition from (k_1, k_2, c) to (k'_1, k'_2, c') can be easily determined. No delay at all is implied for tasks in $\sigma_{c'}$, since crane c' processes its task sequence without any delay. The same reasoning allows us to deduce the time span T covered by the pair of partial routings represented by the transition. For tasks $\sigma_{3-c'}(1)$ to $\sigma_{3-c'}(k'_{3-c'})$, no delay is implied either, since crane $3 - c'$ has no delay implied by this transition up to completing $\sigma_{3-c'}(k'_{3-c'})$. Let T' be the minimum time span required by crane $3-c'$ to process its task sequence as implied by the transition from (k_1, k_2, c) to (k'_1, k'_2, c'). Then, tasks $\sigma_{3-c'}(k'_{3-c'} + 1)$ to $\sigma_{3-c'}(n_{3-c'})$ are delayed by $T - T'$, and the total weighted delay of tasks implied by the transition is

$$c^{del}((k_1, k_2, c), (k'_1, k'_2, c')) = (T - T') \cdot \sum_{k''=k'_{3-c'}+1}^{n_{3-c'}} w_{\sigma_{3-c'}(k'')}.$$

Initial state $(0, 0, 0)$ has $F(0, 0, 0) = 0$, while for each other state (k_1, k_2, c) the Bellman function is

$$F(k_1, k_2, c) = \min \left\{ F(k'_1, k'_2, c') + c^{del}((k'_1, k'_2, c'), (k_1, k_2, c)) \mid (k'_1, k'_2, c') \in P(k_1, k_2, c) \right\},$$

where $P(k_1, k_2, c)$ is the set of states that have a transition leading to (k_1, k_2, c).

The problem then is to determine $F(n_1, n_2, 0)$. We have $O(|J|^2)$ states, $O(|J|)$ transitions starting from each state, and thus $O(|J|^3)$ transitions in total. We can determine the total weighted delay implied by the transition in $O(|J|)$ time by scanning through task sequences. This leaves us with an overall effort of $O(|J|^4)$ and leads to the following results.

Theorem 4 *We can find a feasible solution with a minimum total weighted completion time in $O(|J|^4)$ if no (restricting) release dates or deadlines have to be considered.*

5 Conclusion and outlook

In this paper, we considered a generalization of a problem setting that has been considered in Briskorn and Angeloudis (2016) and of a subproblem arising in Eilken (2019). These problems were proven beneficial when employed as workhorses in Eilken (2019), Nossack et al. (2018), and Zey et al. (2020), but were limited to the objective of makespan minimization. This motivated the investigation of a fairly general setting in the paper at hand. Here, we consider twin cranes and crossover cranes with predetermined task sequences, release dates and deadlines of tasks, movement of cranes along the three dimensions of a container storage block, and the objectives to minimize total task cost or maximum task cost where a task's cost is determined by a task-dependent non-decreasing function of the task's completion time.

In this paper, we resolve the complexity of the most general problem settings. We show that the problem of minimizing total task cost is binary NP-hard for both crane settings. However, a solution minimizing maximum task cost can be determined in polynomial time. For both structures of objective functions, we provide general DP frameworks. Furthermore, we provide strongly polynomial algorithms for two important special cases. Note that our algorithmic ideas for Crossover-Sum and Twin-Sum essentially rely on a reduction in the problem at hand to the problem of finding the shortest path in an acyclic graph. This path can be determined by DP, as proposed in this paper. However, every off-the-shelf shortest path solver can be applied to solve our problem as well.

For future research, we can think of various directions. First, since the problem is motivated mainly as a subproblem in more involved problem settings, it would be interesting to develop algorithms with even better worst-case run time behavior. Second, the algorithms provided here open new

opportunities to tackle holistic crane scheduling problems with objectives other than makespan minimization. In fact, in our opinion, the most promising way to benefit from our algorithms in terms of increased efficiency in real-world terminals is not to apply them in a post-processing step in an existing terminal control system, but to embed them in an integrated approach to address the three-part decision.

Funding Open access funding enabled and organized by Projekt DEAL.

References

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

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

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

Boysen, N., Briskorn, D., & Meisel, F. (2017). A generalized classification scheme for crane scheduling with interference. *European Journal of Operational Research*, *258*(1), 343–357.

Boysen, N., Fliedner, M., Jaehn, F., & Pesch, E. (2013). A survey on container processing in railway yards. *Transportation Science*, *47*(3), 312–329.

Briskorn, D., & Angeloudis, P. (2016). Scheduling co-operating stacking cranes with predetermined container sequences. *Discrete Applied Mathematics*, *201*, 70–85.

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

Briskorn, D., & Zey, L. (2018). Resolving interferences of triple-crossover-cranes by determining paths in networks. *Naval Research Logistics*, *65*, 477–498.

Briskorn, D., & Zey, L. (2020). Interference aware scheduling of triple-crossover-cranes. *Journal of Scheduling*, *23*, 465–485.

Carlo, H., Vis, I., & Roodbergen, K. (2015). Seaside operations in container terminals: Literature overview, trends, and research directions. *Flexible Services and Manufacturing Journal*, *27*(2–3), 224–262. 9.

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

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

Choe, R., Park, T., Ok, S. M., & Ryu, K. R. (2007). Real-time scheduling for non-crossing stacking cranes in an automated container terminal. In M. A. Orgun, & J. Thornton (Eds.) *Proceedings of AI 2007: Advances in artificial intelligence: 20th Australian joint conference on artificial intelligence, Gold Coast, Australia, December 2–6, 2007* (pp. 625–631). Springer.

Choe, R., Yuan, H., Yang, Y., & Ryu, K. R. (2012). Real-time scheduling of twin stacking cranes in an automated container terminal using a genetic algorithm. In *Proceedings of the 27th Annual ACM symposium on applied computing, SAC '12* (pP. 238–243). ACM.

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

Eilken, A. (2019). A decomposition-based approach to the scheduling of identical automated yard cranes at container terminals. *Journal of Scheduling*, *22*, 517–541.

Gharehgozli, A. H., Laporte, G., Yu, Y., & de Koster, R. (2015). Scheduling twin yard cranes in a container block. *Transportation Science*, *49*(3), 686–705.

Gharehgozli, A. H., Yu, Y., de Koster, R., & Udding, J. T. (2014). An exact method for scheduling a yard crane. *European Journal of Operational Research*, *235*(2), 431–447.

Jaehn, F., & Kress, D. (2018). Scheduling cooperative gantry cranes with seaside and landside jobs. *Discrete Applied Mathematics*, *242*, 53–68.

Jessen, H. (2008). Hamburg container terminal altenwerder (digital image). Retrieved 22 Oct 2020 from https://de.wikipedia.org/wiki/Datei:Hamburg-CTA-AGV-2008.JPG

Kovalyov, M. Y., Pesch, E., & Ryzhikov, A. (2018). A note on scheduling container storage operations of two non-passing stacking cranes. *Networks*, *71*(3), 271–280.

Kress, D., Dornseifer, J., & Jaehn, F. (2019). An exact solution approach for scheduling cooperative gantry cranes. *European Journal of Operational Research*, *273*, 82–101.

Li, W., Goh, M., Wu, Y., Petering, M., de Souza, R., & Wu, Y. (2012). A continuous time model for multiple yard crane scheduling with last minute job arrivals. *International Journal of Production Economics*, *136*(2), 332–343.

Li, W., Wu, Y., Petering, M. E. H., Goh, M., & de Souza, R. (2009). Discrete time model and algorithms for container yard crane scheduling. *European Journal of Operational Research*, *198*, 165–172.

Nossack, J., Briskorn, D., & Pesch, E. (2018). Container dispatching and conflict-free yard crane routing in an automated container terminal. *Transportation Science*, *52*(5), 1059–1076.

Speer, U., & Fischer, K. (2017). Scheduling of different automated yard crane systems at container terminals. *Transportation Science*, *51*, 305–324.

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

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

Vis, I. F. A., & de Koster, R. (2003). Transshipment of containers at a container terminal: An overview. *European Journal of Operational Research*, *147*, 1–16.

Zey, L., Briskorn, D., & Boysen, N. (2020). Twin-crane scheduling during seaside workload peaks with a dedicated handshake area. Working Paper Under review.