Scheduling co-operating stacking cranes with predetermined container sequences *

Dirk Briskorn
Lehrstuhl für Produktion und Logistik
Bergische Universität Wuppertal, Rainer-Gruenter-Str. 21, D-42119 Wuppertal, Germany
briskorn@uni-wuppertal.de

Panagiotis Angeloudis
Port Operations Research and Technology Centre
Department of Civil and Environmental Engineering, Imperial College London, UK
pa01@ic.ac.uk

April 2015

Abstract

Crane scheduling in container terminals is known as a difficult optimization problem that has become even more challenging in recent years with the proliferation of multi-gantry automated stacking cranes. In this paper we present an efficient algorithm solving a subproblem arising in this context, namely deciding the priority of cranes after transportation tasks have been assigned. We tackle this problem for both, twin crane setting and crossover crane setting, and develop graphical models and strongly polynomial algorithms accordingly. A series of experiments is carried out where it is shown that the method can produce optimum solutions within exceptionally small run times.

Keywords: Automated Stacking Cranes; Scheduling; Container Terminals; efficient algorithm, shortest path representation.

1 Introduction

Automation has been a key agenda item for port operators and equipment manufacturers over the last 20 years. Seen as the key to significant performance improvements and cost savings, there have been many efforts to automate several terminal operation aspects. Automated Stacking Cranes (ASCs) represent a major success story from this period and have allowed

*One of the authors would like to express their gratitude to the UK Engineering and Physical Sciences Research Council for financing parts of their work.

terminal designers to increase the number of containers processed and stored with less cost and space requirements.

Typically containers are stored in blocks. Each block may be maintained by one or multiple gantry cranes. These 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, ASCs are fixed to a certain block within the container storage area. That is, they manage containers only within the storage area and, consequently, have to hand over the containers to transport devices or receive containers from them (typically automated guided vehicles or ship-to-shore cranes on the seaside and trucks on the land side).

Since some or all types of vehicles deployed in the terminal may be unable to lift containers independently it would be essential to fix a time in the planning horizon where both, vehicles and gantries, would meet at the same place to exchange containers. In practice this requires the presence of a sophisticated scheduling technique that is able to determine a series of crane movements that are coordinated with other terminal activities. While several scheduling algorithms for ASCs have been developed in the past, see Dorndorf and Schneider [4] and Vis and Carlo [13] for example, they often overlook the potential problems that arise from the presence of two or more gantries in the same stack. The study discussed in this paper is part of a greater effort to comprehensively address crane scheduling.

Scheduling problems for ASCs typically require several decision components to be made. Usually, a set of transport jobs is given for the planning horizon under consideration. A transport job corresponds to a single container to be picked up at the origin, moved, and released at the destination. The origin is given naturally by the current position of the container and we assume the destination to be predetermined. The decision components, then, can be described as follows.

1. We need to decide which job is done by which crane in case there is more than one.
2. Given an assignment of jobs to cranes we need to decide the sequence of jobs for each crane.
3. Given the above decisions the actual point of time of each operation has to be determined. This involves, in case there are multiple cranes, resolving conflicts between cranes' operations. Typically this means that whenever two cranes cannot execute certain operations in parallel we have to decide which crane gets the right of way. The type of operations which can be executed in parallel depends on the crane design.

Throughout this paper we focus on the third decision assuming that the first two decisions have been made already. We focus on two different settings with two cranes each serving a single container block, namely crossover cranes and twin cranes. On two opposing sides of the block there are dedicated handover areas for exchanging containers with other transport devices. Figure 1 depicts both settings looking from above. A pair of crossover cranes consists of a larger crane (Crane 1) and a smaller crane (Crane 2) using different tracks. This allows both cranes to pass each other, i. e. both cranes may serve both ends of the block. However, while the larger crane's spreader is releasing or lifting a container in a certain row the smaller crane can neither travel across this row nor perform a release or lift in the same row. In twin configurations the cranes have similar or identical gantry, use the same tracks, and, therefore, cannot pass each other. As a result, the two cranes are destined to exclusively serve the

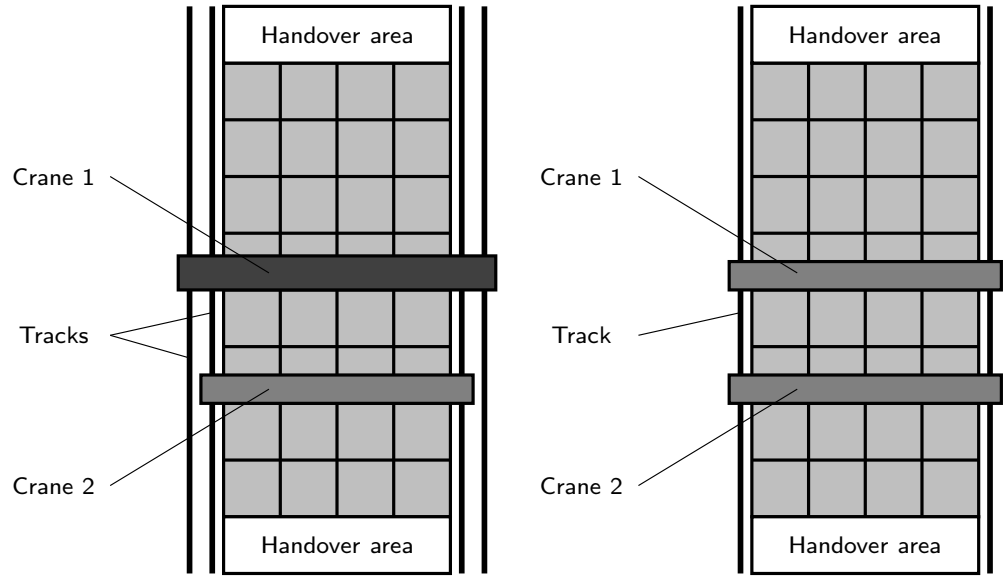

Figure 1: Crossover cranes (left) and twin cranes (right)

opposing ends of the block, with careful planning required when they need to operate in the middle sections.

This rather narrow problem setting can be motivated easily. First, since the aspect of avoiding collisions is rarely considered in the literature on a detailled level we may take an arbitrary schedule provided by one of the existing approaches and reoptimize the priorities without modifying the sequences of operations assigned to both cranes. Furthermore, the algorithm can be used as a component of a metaheuristic approach for holistic crane scheduling (i. e. that also addresses the allocation of jobs to cranes), with the methodology presented here being responsible for deciding movement priorities and accurately resolving any potential conflicts. A natural representation scheme used in a metaheuristic would be to have a sequence of containers for each crane implying the sequence of transport jobs as processed by cranes. Clearly, these sequences do not fully specify a schedule which is implementable since the third decision component is not represented. There are two options at hand. First, we may extend the representation scheme used in the metaheuristic framework in order to represent priorities, as well. It is likely that we end up with a representation scheme having several sections differing in their semantics. Moreover, redundancy can hardly be avoided. The second option is to design an efficient module deciding about priorities in traffic and to employ it to "interpret" individuals specified by sequences of operations. Then, each individual corresponds to many schedules and the module allows us to find the optimum schedule among those represented. This reduces the size of the search space (in comparison to the first option), reduces redundancy, and supports a representation scheme with a uniform structure.

The contribution of this paper is to twofold. First, we give a graphical represenation of an optimization problem with respect to the third decision component. This representation gives valuable insights and enables us to provide our second contribution: strongly polynomial algorithms solving the optimization problem for both, crossover cranes and twin cranes, to optimality.

The paper proceeds as follows. In Section 2 we outline related literature and in Section 3 we formally define the problem. Section 4 provides a graphical model representing the problem at

hand providing a reduction of our problem to a shortest path problem in a plane with obstacles. The latter problem in turn is reduced to the shortest path problem in a graph in Section 5 where we also show efficiency of the resulting approach. Finally, we conclude the paper in Section 6.

2 Previous work

A general overview about operations in a container terminal and corresponding optimization approaches is given in Steenken et al. [11] and updated in Stahlbock and Voß [10]. Vis [12] gives an overview focussing on design and control issues.

The relative novelty of ASCs is reflected in the academic literature, which contains only a few existing studies on this topic. While in the past, several methods were developed for scheduling single (mostly non-automated) stacking cranes (see, e. g., Daganzo [3] and Ng and Mak [9]), the interaction between multiple gantries necessitates a novel scheduling technique that determines conflict-free schedules. Ng [8] was among the first to investigate the optimal scheduling of multi-gantry cranes, and presents an Integer Programming model that can be used to determine the sequence of a set of container moves. Collisions between gantries are avoided through the enforcement of allowable movement ranges at any point in time. The approach presented in this paper focuses on twin crane configurations, and a dynamic programming-based heuristic is presented accordingly.

A different algorithm for similar problem instances is also discussed by Li et al. [7]. An initial Mixed Integer Programming model that can cater for inter-crane interference is presented that is less computationally demanding than the one in Ng [8]. Furthermore, rolling-horizon adaptations and other heuristics that can reduce computational times to seconds are presented.

A later study by Lee et al. [6] focuses on scheduling of multiple transtainer cranes. These are mostly encountered in Far Eastern ports and involve a buffer that runs along the length of the block. As a result, vehicles are able to approach gantries from the side, and crane travel is significantly reduced. Vis and Carlo [13] provide a model formulation for ASC blocks with two crane gantries, that includes considerations for access restrictions. To obtain a lower-bound solution to this problem, they transform the two-crane problem into a single-crane problem and adapt a dynamic programming algorithm that was previously developed by Vis and Roodbergen [14] focusing on optimal sequences of storage and retrieval requests for single straddle carriers. Furthermore, a heuristic based on Simulated Annealing that can more accurately accommodate the operational characteristics of such ASC configurations is provided.

A scheduling algorithm for triple cross-over stacking cranes is discussed by Dorndorf and Schneider [4]. As a subproblem the triple crane version of the problem we consider here is tackled. Avoiding collisions for a given sequence of operations is accomplished by employing a branch & bound algorithm. For the instance sizes tested run times are encouraging. However, run times can be expected to increase exponentially in the number of containers considered and the run time complexity of the problem remains open. It should be emphasized that to the best of our knowledge Dorndorf and Schneider [4] provides the only approach so far that allows for an exact solution of the type of problem we are focussing on here. This is remarkable since as **?**] report that crane interferences influence durations of container transports significantly.

3 Problem definitions

We consider a continuous time horizon. We refer to the time interval $[t-1, t]$ with $t \in \mathbb{N}$ as period t in the following. There is a set of containers. Each of these containers has to be transported exactly once. The origin and destination rows of a container j are denoted as o_j and d_j, respectively, and predetermined. For each container j the duration p_j^l of lifting it in o_j and the duration p_j^r of releasing it in d_j is a multiple of periods and is assumed to be deterministic and known in advance. For the purposes of this study, these durations include horizontal spreader movements within rows. Each container is assigned to exactly one crane and for each crane we have a sequence of those containers being assigned to it. The sequence implies the order in which the crane must process the containers. Each container j has to be transported from o_j to d_j by the crane it is assigned to. Both cranes can travel with the same speed of one row per time unit.

For the input described above, a schedule for crane c, $c \in \{1, 2\}$, can be specified as a sequence of activities for each period. These can be:

- movement from row r to row $r+1$ with $0 \leq r < R+1$,
- movement from row r to row $r-1$ with $0 < r \leq R+1$,
- lifting a container j in row o_j,
- releasing a container j in row d_j, or
- waiting in row r.

A schedule for a crane c will be deemed feasible if

- after arriving at, waiting in or operating in row r the next activity starts from or takes place in r,
- crane c will lift or release a container j only if j has been assigned to c,
- a container j is released only after it has been lifted,
- each activity between the lift and release of container j is either waiting or moving,
- during the lift and the release of container j which takes exactly p_j^l and p_j^r periods, respectively, c is located in o_j and d_j, respectively, and
- a container j is transported only after all its predecessors (in the sequence of the crane) have been transported and before any of its successors is transported.

The length $l(\sigma^c)$ of a schedule σ^c for crane c is equal to the number of periods it covers. A feasible schedule σ is a pair (σ^1, σ^2) where σ^1 and σ^2 are feasible schedules for cranes 1 and 2 respectively that do not violate any movement restrictions. These restrictions depend on the specific type of cranes and are detailed in Sections 3.1 to 3.2. The goal of our methodology therefore becomes to find a feasible schedule $\sigma = (\sigma^1, \sigma^2)$ that minimizes the overall makespan $\max\{l(\sigma^1), l(\sigma^2)\}$.

3.1 Crossover cranes

We assume that cranes 1 and 2 is the larger and the smaller crane, respectively. The movement restrictions related to cranes impeding each other for this case are as follows. If crane 1 lifts a container or releases a container in row r during period t, then crane 2 cannot

- move from r to $r+1$, from r to $r-1$, from $r+1$ to r, or from $r-1$ to r in t,
- wait in r in t, or
- lift or release a container in r in t.

3.2 Twin cranes

We assume that cranes 1 and 2 serve rows 0 and $R+1$, respectively. The restrictions on simultaneous movements of the cranes require that in each period

- crane 1 stays in row r and crane 2 stays in row r' with $r' \geq r+1$,
- crane 1 stays in row r and crane 2 moves from r' to r'' with $\min\{r', r''\} \geq r+1$,
- crane 2 stays in row r and crane 1 moves from r' to r'' with $\max\{r', r''\} \leq r-1$, or
- crane 1 moves from r to r' and crane 2 moves from r'' to r''' with $r'' \geq r+1$ and $r''' \geq r'+1$.

Note that these rules allow crane 1 to move from r to $r+1$ while crane 2 is moving from $r+1$ to $r+2$ in the same period. It would be possible to modify the model to prohibit such simultaneous movements without many complications.

4 Graphical models

This section outlines graphical models that represent the problems specified in Section 3. These models provide insights into the problems' structures and allow a reduction to the shortest path problem. They considerably rest upon on an established approach for the graphical representation of the job shop problem with M machines and two jobs. Akers [1], Brucker [2] and Hardgrave and Nemhauser [5] have discussed representations of the job-shop problem using geometric models, with the overall aim being to find the minimum makespan schedule.

4.1 Crossover cranes

In this section we develop a graphical model for the problem to schedule crossover cranes. We introduce the complete model in two steps. In the first step, we abstain from allowing detours for cranes. That is, if two consecutive container lift and release activities of crane c take place in rows r and r', then each move of c must reduce its distance to r'. This allows us to develop the framework in which we embed detours in the second step.

4.1.1 Schedules without detours

First, we determine non-delay schedules for both cranes. A non-delay schedule of crane c is a feasible schedule for crane c where c does not wait and does not do any detours. The makespan C_1 and C_2 for the non-delay schedule of crane 1 and 2, respectively, equals the sum of total operation time and total travel time of crane 1 and 2 (assuming that there are no detours). Therefore, since the sequence of containers to be transported by c is fixed for each point of time we can determine the exact position of c according to the non-delay schedule.

Note that if both non-delay schedules are not in conflict with each other then we have found an optimum solution as the schedule which consists of both non-delay schedules.

Let us consider an example where $R = 6$ and crane 1 starts in row 0, picks up a container in row 2, delivers it to row 4, picks up a second container in row 3, and delivers it to row 2. Both, picking up and releasing, take one period for the first container and take two periods for the second container. Afterwards, crane 1 returns to row 0. Crane 2 starts in row 7, picks up a container in row 2 (one period) and releases it in row 4 (two periods), and, finally, returns to row 7. Figure 2 represents the non-delay schedules by outlining the position of both cranes over time. Note that the non-delay schedules are in conflict since crane 1 operates in row 3 while crane 2 is passing through row 3.

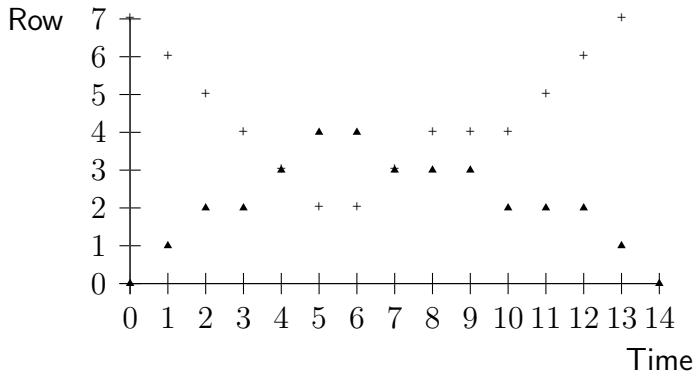

Figure 2: Movement graph for non-delay schedules of crossover cranes

Now, we develop a graphical model which is semantically identical to the model used for the job shop problem outlined in Hardgrave and Nemhauser [5]. The model encompasses a rectangular area with width equal to C_1 and height equal to C_2. Each point (t_1, t_2) in the rectangular area represents a state where crane 1 and 2 has completed the first t_1 and t_2 time units of its non-delay schedule, respectively. Note that if the value of t_c is not an integer then the location of crane c may not be clearly defined to be one of the rows of the block under consideration. If crane c moves from row r to $r + 1$ in period t then we say that it is located in row $r + t_c - (t - 1)$ for each $t - 1 \leq t_c \leq t$ in the following. Similarly, if c moves from row r to $r - 1$ during period t then we say that it is located in row $r - (t_c - (t - 1))$ for each $t - 1 \leq t_c \leq t$ in the following. This gives an intuitive interpretation of the position of both cranes in each point of time.

Note, however, that not each point is feasible with respect to the movement restrictions. Clearly, an infeasible point has to correspond to crane 1 operating in a row r (and, thus, having its spreader lowered) and crane 2 operating in row r, entering r, or leaving r (which

cannot happen in parallel). We derive an obstacle for each pair of an operation of crane 1 in a row r and each entering of r and leaving of r by crane 2 (possibly with an operation in between). For crane 1 operating in r in periods t_1^1 to t_1^2 and crane 2 entering r in period t_2^1, leaving r in period $t_2^2 > t_2^1$, and operating in r in periods $t_2^1 + 1, \ldots, t_2^2 - 1$ if $t_2^2 > t_2^1 + 1$ we define an obstacle covering all points (t_1, t_2) with $t_1^1 - 1 < t_1 < t_1^2$ and $t_2^1 - 1 < t_2 < t_2^2$.

Pursueing the example in Figure 2 we obtain the graphical model depicted in Figure 3. E. g., point $(5, 6)$ represents the state where crane 1 is just about to release its first container in row 4 and crane 2 has just picked up its container but has not left row 2 yet. The right-most obstacle corresponds to crane 1 releasing its second container in row 2 while crane 2 is moving into row 2, picking up its container, or leaving row 2. The number denoted in each obstacle refers to the corresponding row r where the obstruction potentially takes place.

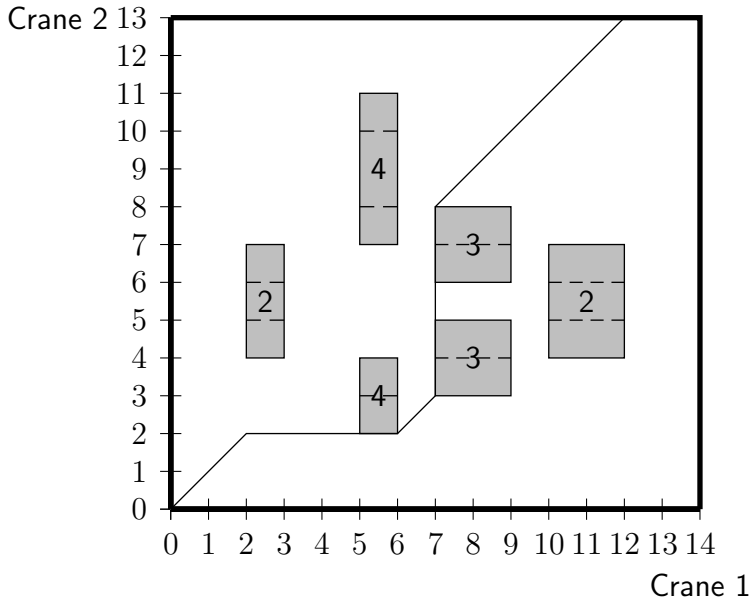

Figure 3: Obstacle graph for a crossover crane configuration

We can identify columns and lines of obstacles. Obstacles in the same column correspond to a specific operation of crane 1. Obstacles in the same line correspond to crane 2 moving to a certain row r, possibly operating in r, and leaving r. Figure 3 shows four columns corresponding to crane 1 operating in rows 2, 4, 3, and, again, 2. Furthermore, we have five lines corresponding to crane 2 moving through row 4, moving through row 3, operating in row 2, moving through row 3 again, and operating in row 4. Obstacles are divided into three parts. The lower, middle and upper parts correspond to crane 2 moving to row r, operating in r and leaving r, respectively. Note that the middle part has height zero if crane 2 is moving through the row but not operating in it.

It is not hard to see that the obstacles derived as described above accurately specify the set of infeasible points. In terms of the graphical model, finding a schedule would be to determine a path from the lower left corner of the area to the upper right corner of the area. A feasible path consists of lines that can only run horizontally, vertically, or diagonally and must not cross through obstacles. Horizontal sections, vertical sections, and diagonal sections correspond to time intervals where only crane 1 processes along its non-delay schedule, only crane 2 processes along its non-delay schedule, and both cranes processe along its non-delay

schedule in parallel. For the purposes of this study, the lengths of a horizontal segment between (t_1, t_2) and $(t_1 + x, t_2)$, a vertical segment between (t_1, t_2) and $(t_1, t_2 + x)$, and a diagonal segment between (t_1, t_2) and $(t_1 + x, t_2 + x)$ are considered to be equal to x. Thus, the length of each segment represents the timespan necessary for both cranes to carry out the corresponding activities. Now, the shortest path from $(0, 0)$ to (C_1, C_2) not cutting through obstacles represents the minimum makespan schedule.

A feasible path is depicted in Figure 3 which runs through points $(0, 0)$, $(2, 2)$, $(6, 2)$, $(7, 3)$, $(7, 8)$, $(12, 13)$, and $(14, 13)$. This path represents a schedule as follows. For two periods both cranes proceed along their non-delay schedules. Then, crane 2 waits for crane 1 to release the first container in row 4. Then, crane 1 waits for crane 2 to pass through row 3 twice. Afterwards, both cranes proceed along their non-delay schedules in parallel. The length of this path is 19.

It remains to show how a shortest path can be found. It is obvious that there may be an infinite number of paths with the required properties. Thus, reducing this set to a finite set of candidate paths is substantial. We pick up ideas presented in Brucker [2] and Hardgrave and Nemhauser [5] and adapt them to our model in order to determine finite sets of horizontal, vertical and diagonal sections such that the overall shortest path can be constructed as a concatenation of a subset of these sections.

We say that an obstacle o is concealed by another obstacle o' with respect to a diagonal if

- o and o' are neither in the same line nor in the same column,
- o and o' overlap with respect to their positions regarding the time axis of crane 2,
- o's position is larger than o''s position with respect to the time axis of crane 2, and
- the diagonal lies left of both, o and o', at the time coordinate of crane 2 where o and o' overlap.

Furthermore, we say that an obstacle o' conceals o indirectly with respect to a diagonal d if there is an obstacle o'' such that o' conceals o'' (indirectly) with respect to d and o'' conceals o (indirectly) with respect to d. In Figure 3, the lower obstacle labelled 3 conceals the right obstacle labelled 2 and the lower obstacle labelled 4 conceals the lower obstacle labelled 3, respectively, with respect to the diagonal starting at $(0, 0)$. Furthermore, the lower obstacle labelled 4 conceals the right obstacle labelled 2 indirectly with respect to this diagonal.

Note that obstacles concealing each other may imply that a path leading below a certain obstacle o in line k and column l must lead below an other obstacle o' in line k', $k' < k$, and column l', $l' < l$. For example, in the setting according to Figure 3 there is no path leading below the right obstacle labelled 2 but above one of the concealing obstacles.

The approach for finding all necessary sections is specified in the following.

1. Set $(s_1, s_2) \leftarrow (0, 0)$
2. Starting at (s_1, s_2) move diagonally until the top or right edge of the outer rectangle or an obstacle o, say at (t_1, t_2), is reached.
3. If an obstacle is reached, then branch.

(a) Move horizontally from the diagonal to the lower right corner (q_1, q_2) of each obstacle which lies right of the diagonal, is not concealed with respect to the diagonal, and has $s_2 \leq q_2 \leq t_2$. Also, if (t^1, t^2) is at the bottom edge of o, move horizontally along the bottom edge of the obstacle until the lower right corner of o is reached. Go to step 2 using each of these corners as a new starting point.

(b) Move vertically from the diagonal to the upper left corner (q_1, q_2) of the lowest obstacle in each column which lies above the diagonal and has $s_1 \leq q_1 \leq t_1$. Also, if (t^1, t^2) is at the left edge of o, move vertically along the left edge of the rectangle until the upper left corner of o is reached. Go to step 2 using each of the corners as a new starting point.

4. If the top or right edge of the outer rectangle is reached then move along that edge to the finish point.

The basic idea of this approach is quite intuitive. Diagonal sections support a short schedule since both cranes can proceed in parallel. Hence, we construct a diagonal section whenever it is possible. If the diagonal section encounters an obstacle we have to provide a path around it. For this, there are two options: The path may lead left of and above the obstacle or below and right of the obstacle. As soon as possible, then, a new diagonal section is started.

It remains to prove that the set of paths constructed by the approach presented above indeed contains the shortest path.

Theorem 1. *The set of paths constructed by the modified approach contains at least one shortest path.*

Proof. The proof is carried out by induction on the number h of times an obstacle is encountered by a diagonal during construction of the network by our method.

If $h = 0$ (no obstacle is encountered while constructing the network based on a diagonal starting at $d = (d^1, d^2)$), then the modified approach yields a single path only. This path obviously is a shortest path.

Now let us assume that the statement holds for $h-1$. If obstacles are encountered by diagonals h times while constructing the network, consider a network based on a diagonal starting at $d = (d_1, d_2)$. Let o be the obstacle encountered by the diagonal starting at d.

It is obvious that we find the shortest paths from d to all lower right corners of obstacles and to all upper left corners of obstacles connected to the diagonal. Furthermore, we find the shortest paths from these corners to C since obstacles are encountered less than h times in the corresponding networks and, thus, the inductive hypothesis applies. Thus, if there is a shortest path passing through one of these corners, then the network starting at d contains the shortest path from d to C.

We show that if the shortest path passes below and right of o, then we can assume that it passes through one of these corners. The case where the shortest path passes left of and above o follows analogically. Consider a shortest path passing below and right of o and not passing through any of these corners. In the following we consider obstacles which are the right-most obstacles in their respective line the optimum path passes on the right and the bottom-most obstacles in their respective column the optimum path passes below. Among these obstacles,

consider obstacle o' which has the lowest lower right corner $l = (l_1, l_2)$ with $l_2 \geq d_2$. Note that our algorithm connects the diagonal with l.

The optimum path has to go through (l_1, y) for $y < l_2$ and (x, l_2) for $x > l_1$. Both, the shortest path from d to l and the shortest path from d to (l_1, y), must have length of at least $l_1 - d_1$ since $l_1 - d_1 \geq l_2 - d_2 > x - d_2$. Note that the path from d to l found by our method has this minimum length. Now consider the path from (l_1, y) to (x, l_2) and the horizontal connection from l to (x, l_2). Note that the horizontal connection does not cut through any obstacles by the choice of o'. Clearly, the shortest path from (l_1, y) to (x, l_2) cannot be shorter as the horizontal connection. Therefore, we can construct a path from d to (x, l_2) through l which is not longer than the connection from d to (x, l_2) on the shortest path.

Summarizing, we find (i) a shortest path from d to each of the connected lower right corners and a network containing the shortest path from each of these corners to C according to the inductive hypothesis, (ii) a shortest path from d to each of the connected upper left corners and a network containing the shortest path from each of these corners to C according to the inductive hypothesis, and (iii) we can assume that there is a shortest path from d to C passing through one of the connected corners. Thus, our network contains a shortest path if obstacles are encountered h times. This completes the proof. □

The network constructed for the example discussed before is depicted in Figure 4.

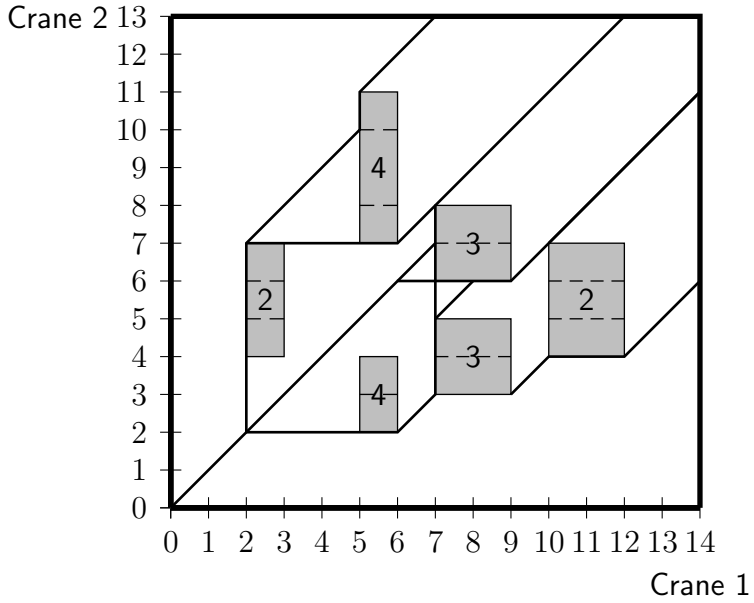

Figure 4: Obstacle graph for a crossover crane configuration

4.1.2 Schedules with detours

Theorem 1 tells us that we can reduce the crossover crane version of the problem without detours to the problem to find the shortest path in a network. A different approach is required for the case where a crane is allowed to make detours. Note that while a detour increases the total travel distance of one crane, it may allow us to determine a schedule with a shorter makespan due to reduced waiting time of the other crane. To show this, we consider the

example depicted in Figure 5. We may say that crane 1 has already picked up a container in row 1 and is about to start to row 5 to deliver it. Afterwards, it picks up a container in row 4, delivers it to row 3, and, finally, moves to row 8. Crane 2 has picked up a container in row 6 and is about to start to row 5 to deliver it. Afterwards, it will pick up a container in row 4, release it in row 3, and move to row 2.

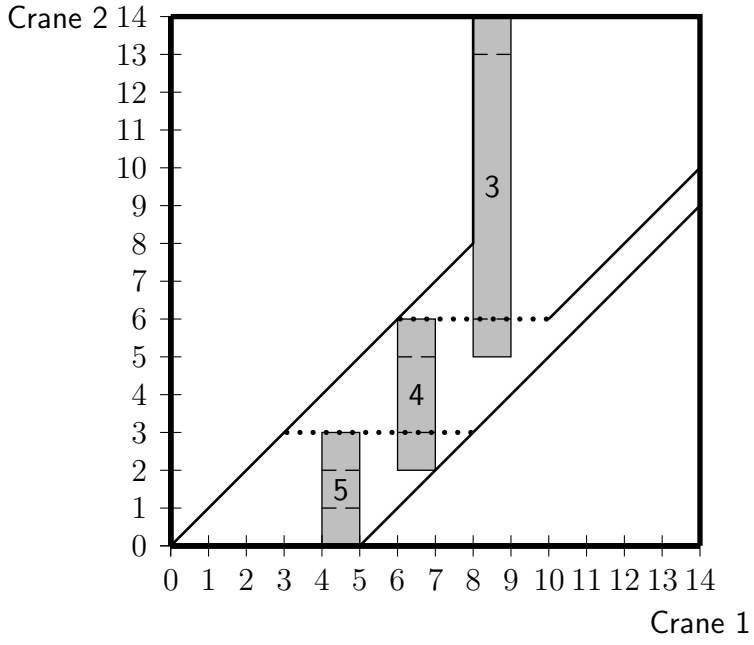

Figure 5: Obstacle graph for crossover crane with detours

In addition to the two paths considered in Section 4.1.1, Figure 5 includes a third path p leading from $(0, 0)$ through $(3, 3)$, $(8, 3)$, and $(14, 9)$ to (C_1, C_2) and a fourth path p' leading from $(0, 0)$ through $(6, 6)$, $(10, 6)$, and $(14, 10)$ to (C_1, C_2). Path p corresponds to Cranes 1 and 2 simultaneously moving to row 4 in the first 3 periods. Afterwards, Crane 2 does not start its operation in row 4 but rather makes a detour (by moving to row 5, e. g.) that allows Crane 1 to operate in row 4 first. Crane 2 returns to row 4 as soon as Crane 1 has finished its operation in row 4. Then, both cranes proceed according to their schedules in parallel. Crane 1 finishes after 14 periods. At this point of time, Crane 2 still has 5 unprocessed periods of its schedule. Since Crane 2 is not delayed anymore it concludes its operations after 19 periods. According to path p' Cranes 1 and 2 simultaneously process their respective schedules for the first 6 periods. Then, Crane 2 makes a detour (by moving to row 4, e. g.) in order to let Crane 1 operate in row 3 first. Crane 2 returns to row 3 as soon as Crane 1 has finished its operation in row 3. Afterwards, both process their schedules in parallel, Crane 1 finishes after 14 periods and Crane 2 finishes after 18 periods. Thus, p' is shorter than the paths considered in Section 4.1.1 which have lengths 20 and 19 which proves that detours may enable us to shorten the makespan.

Note that representing a detour by a dotted line connecting points s and t does not imply that points in between s and t are actually reached. We only know that both, s and t, are reached and that the timespan covered by the detour equals distance of s and t, that is $|s_1 - t_1| + |s_2 - t_2|$.

In the following we restrict the type of detours we have to consider without losing optimality of the resulting schedule.

Lemma 1. *There is an optimum schedule such that*

1. *cranes deviate from their prescribed row sequence only in order to reduce waiting time of the other crane,*
2. *crane 1 does not deviate from its row sequence,*
3. *crane 2 starts a detour only in a row where both cranes are going to operate next,*
4. *crane 2 returns to row r where it started its detour as soon as possible, that is, it starts entering r right after crane 1 finished an operation in r, and*
5. *crane 2 starts a detour in row r only if both cranes approach r from the same row (either $r-1$ or $r+1$), and crane 1 has operated in $r-1$ since crane 2 left $r-1$.*

Proof. The first statement of the lemma follows from the fact that a crane detour increases its total travel distance and therefore its makespan. Consequently, we consider a detour only if it implies a reduction on the makespan of the other crane. Since neither travel times nor operation times can be influenced we can only reduce the makespan by reducing the total waiting time.

Then, the second part of the lemma follows immediately since we can avoid delaying crane 2 by simply not having crane 1 waiting.

Crane 2 may delay crane 1 only in a row where crane 1 operates since both cranes do interfere only if crane 1's spreader is lowered. Note that if crane 2 is not about to operate in the same row, then it can move out of crane 1's way following its prescribed row sequence. Hence, the third statement of the lemma follows.

Fourth, crane 2 has to return to the row r where it started its detour, naturally, since r is its next operation's row due to the third part of this lemma. Then, crane 2 may reach r one period after crane 1 has finished operating in r since Crane 2 may wait in a neighboring row.

The fifth part of the lemma can be justified as follows. Assume that cranes 1 and 2 approach r from different rows. Then, crane 2 can simply wait in the neighboring row of r it passes until crane 1 finished its operation in row r. Crane 2 would reach r one period after crane 1 finished operating in r (which is as early as possible) without making a detour. Hence, assume that cranes 1 and 2 approach r from the same row, let us say $r-1$ w. l. o. g., but crane 1 did not operate in $r-1$ before. Then, Crane 2 can simply wait in $r-1$ until crane 1 finished its operation in row r. Again, crane 2 would reach r not later than before without making a detour. Note that the fifth part of the lemma exactly specifies two overlapping obstacles of different lines. □

In terms of our graphical model this means we can restrict ourselves to detours that

- start at a diagonal d which encounters an obstacle o in line k (according to the first part of Lemma 1),
- run horizontally (according to the second part of Lemma 1),

- cut either through o or through obstacle o' in line k', $k' \leq k$, concealing o (indirectly) with respect to d if o' is concealed with respect to d itself (according to the fifth part of Lemma 1),
- cut through an obstacle o' one period above its lower boundary (according to the third part of Lemma 1), and
- end one period right of a obstacle in line k' concealing o (according to the fourth part of Lemma 1).

Next, we apply two more restrictions.

Lemma 2. *Assume that a diagonal d encounters an obstacle o in line k. There is an optimum schedule such that if a detour starts at d and cuts through obstacles in line k', $k' < k$, then it will cut through each obstacle in line k concealing o with respect to d.*

Proof. Consider an optimum schedule represented by a path p and a detour being part of p and starting at (s^1, s^2) from diagonal d which encounters obstacle o in line k. Assume that this detour cuts through at least one obstacle in line k' but it does not cut through all obstacles in k' concealing o with respect to d. Furthermore, assume that this detour is the first detour of this kind in p. Clearly, either (i) p must lead to the upper left corner u of o or (ii) at least one more detour cutting obstacles concealing o with respect to d is used in p.

(i) We can find a path p' which is identical to p except for the connection from (s^1, s^2) to u. We can simply follow d until it encounters o and, then, go vertically to u. This connection cannot be longer than the one in p and, thus, p' cannot be longer than p. Note that the number of detours not cutting all concealing obstacles in a line in p' is smaller than in p.

(i) Let (t^1, t^2) be the starting point of the detour being used next. We can find a path p' which is identical to p except for the connection from (s^1, s^2) to (t^1, t^2). We can follow d until $(s^1 + (t^2 - s^2), t^2)$ and start a detour. This detour will lead through (t^1, t^2). Note that p' cannot be longer than p and the number of detours not cutting all concealing obstacles in a line in p' is smaller than in p.

By repeating the step described above we can find a schedule as described in Lemma 2. This completes the proof. □

Lemma 3. *There is an optimum schedule where a detour starting from diagonal d cuts only through obstacle o encountered by d.*

Proof. According to Lemma 2, each path containing a detour starting at a diagonal d encountering obstacle o must either pass o below or cuts o. We shall show that the shortest path using a detour starting at (s^1, s^2) and passing o below cannot be shorter than the shortest path using a detour starting at d and cutting through o.

Consider the point (t^1, t^2) where a detour cutting through o ends. Clearly, each path passing below o must pass through (t^1, t'^2), $t'^2 \leq t^2$, since $(t^1 - 1, t^2 - 1)$ is the lower right corner of o. Note that both connections, from (s^1, s^2) to (t^1, t'^2) and from (s^1, s^2) to (t^1, t^2), have length of $t^1 - s^1$. Now, with similar arguments as before it is easy to see that reaching (t^1, t^2) is dominant to reaching (t^1, t'^2) at the same point of time (distance to $(0, 0)$) since the schedule of Crane 2 has been processed further. This completes the proof. □

We outline Lemma 3 using Figure 5, again. The upper detour dominates over the lower one since $(10, 6)$ is reached after 10 periods with the upper detour as $(10, 5)$ is reached after 10 periods following the lower detour.

Lemma 3 provides insights that enable us to modify the method for network generation proposed in Section 4.1.1 for scenarios with detours. We therefore introduce the following step to the approach outlined at the end of Section 4.1.1.

3. (c) If o is concealed with respect to the the diagonal reaching o, then move horizontally from $(t^1 - (t^2 - (l^2 + 1)), l^2 + 1)$ to $(l^1 + 1, l^2 + 1)$ where (l^1, l^2) is the lower right corner of o.

Summarizing, we can reduce the crossover crane version of the problem defined in Section 3 to the problem to find the shortest path in the network constructed by our method.

4.2 Twin cranes without detour movements

In this section we develop a graphical model for the problem to schedule twin cranes. Again, we introduce the model in two steps, first abstaining from detours and introducign them in the second step.

4.2.1 Schedules without detours

The idea is similar to the one presented in in previous sections. Note that in line with the interpretation of point (t_1, t_2) outlined in Section 4.1 there is a row r_1 and r_2 implies as position of crane 1 and 2, respectively.

Clearly, crane 1 operating in r_1 is in conflict to crane 2 operating in r_2 if $r_1 > r_2 - 1$. In the following we consider an obstacle corresponding to each pair of operations of cranes 1 and 2 being in conflict. Such an obstacle covers the time coordinates where both operations are in conflict, those where the operation of one crane is in conflict with movements of the other crane related to its operation, and those where movements of both cranes related to these operations are in conflict with each other. In the following we specify such an obstacle under the assumption that both cranes enter the area of conflict as late as possible before starting the operations and both cranes leave this area as fast as possible after completing the operation. More specifically, we assume that crane 1 and 2 starts in row 0 and $R + 1$, respectively, immediately starts the operation when reaching the respective row, and leaves it towards row 0 and $R + 1$, respectively, as soon as finishing the operation. Note that this assumption is not necessarily justified by the non-delay schedules.

We now formally specify the obstacle corresponding to cranes 1 and 2 operating in rows r_1 and r_2, $r_1 > r_2 - 1$, in periods t_1^1 to t_1^2 and t_2^1 to t_2^2, respectively. We consider an area C that corresponds to the operations themselves neglecting movements. Area C is well defined as a rectangle with a corner at $(t_1^1 - 1, t_2^1 - 1)$, $(t_1^2, t_2^1 - 1)$, $(t_1^1 - 1, t_2^2)$, and (t_1^2, t_2^2). The obstacle as a whole consists of an area C and a circlet containing points which do not belong to C and have a shortest Euclidian distance to a state in C of less than $r_1 - r_2 + 1$.

In order to provide an intuitive understanding we consider an example where Crane 1 operates in row 8 in periods 9 and 10, while Crane 2 operates in row 5 in periods 7, 8 and 9. We

assume here that cranes 1 and 2 start in rows 0 and 11 (respectively) at time 0 and return there afterwards. Hence, given the rows and times of operation the obstacle will be as small as possible. The corresponding obstacle is depicted in Figure 6.

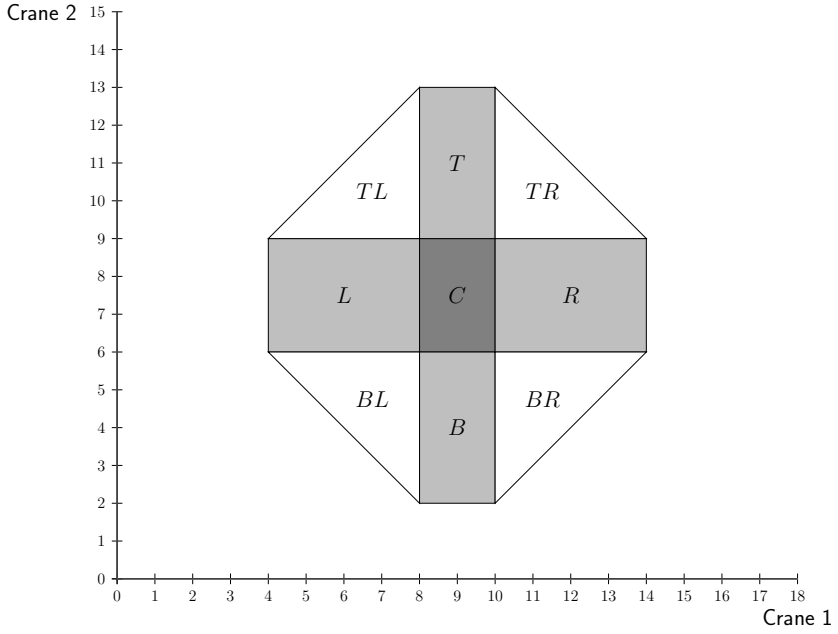

Figure 6: Obstacle graph for twin cranes

Each obstacle is composed by nine parts. The dark-grey shaded area labeled C covers the time coordinates where both cranes are operating. The remaining parts correspond to movements being in conflict with operations or other movements and form the circlet mentioned above. The light-grey shaded areas correspond to one of both cranes operating while the other one is moving. Here, B corresponds to crane 2 passing the row crane 1 is operating in and approaching the row where it operates next while T corresponds to crane 2 passing the row crane 1 is operating in after leaving the row of its own operation. Accordingly, L and R can be interpreted by switching cranes 1 and 2 in the description above. The specification of the light-grey shaded areas can be easily derived from the above. Heights of L and R and widths of T and B correspond to the duration of operations crane 2 and crane 1, respectively. Widths of L and R and heights of T and B equals the duration of travel in the conflicting area, that is $r_1 - r_2 + 1$. Finally, white areas correspond to movements of crane 1 being in conflict with movements of crane 2. The area TL corresponds to crane 2 moving towards row $R + 1$ after operating and crane 1 moving towards row 8 in order to operate. Since we assume that cranes 1 and 2 can simultaneously move from r to $r + 1$ and from $r + 1$ to $r + 2$, respectively, we obtain the straight line from $(4, 9)$ to $(8, 13)$. We can reason similarly for BR, BL, and TR.

From the description above it is clear that each time coordinate covered by an obstacle corresponds indeed to infeasible positions of cranes 1 and 2. The reasoning can be stated as follows. Starting from infeasible positions of operations r_1 and r_2 total movement of both cranes must bridge $r_1 - r_2 + 1$ rows in order to reach positions r'_1 and r'_2 with $r'_1 \leq r'_2 - 1$. We will now show that, moreover, there is no infeasible time coordinate of cranes 1 and 2 not covered by an obstacle corresponding to a pair of conflicting operations of cranes 1 and 2.

Lemma 4. *Each infeasible time coordinate of cranes 1 and 2 is covered by at least one obstacle corresponding to a pair of conflicting operations of cranes 1 and 2.*

Proof. We distinguish between infeasible points, i. e. time coordinates, corresponding to (i) both cranes operating, (ii) one crane operating and the other crane moving, and (iii) both cranes moving. Consider a specific infeasible time coordinate (t^1, t^2).

(i) Clearly, each infeasible time coordinate is covered by area C of an obstacle.

(ii) We restrict ourselves to crane 1 operating and crane 2 moving. For the other case the reasoning is analogue.

Let us assume that crane 1 is operating in r_1 at time t_1 of its non-delay schedule and crane 2 is moving from the last operation's row r_2^1 to the next operation's row r_2^2 at time t_2 of its non-delay schedule. First, if $r_2^1 < r_2^2$, then $r_2^1 < r_1$ and crane 2 is strictly moving from row r_2^1 towards row $R+1$ at time t_2. Thus, (t_1, t_2) must be covered by the obstacle corresponding to the operations of crane 1 in r_1 and crane 2 in r_2^1. Second, if $r_2^1 > r_2^2$, then $r_2^2 < r_1$ and crane 2 is strictly moving towards row r_2^2 at time t_2. Thus, (t_1, t_2) must be covered by the obstacle corresponding to the operations of crane 1 in r_1 and crane 2 in r_2^2.

(iii) Let crane c, $c \in \{1, 2\}$, be moving from the last operation's row r_c^1 to the next operation's row r_c^2 in (t_1, t_2). In each of the following cases a similar reasoning as in (ii) applies.

- If $r_1^1 < r_1^2$ and $r_2^1 < r_2^2$, then (t_1, t_2) must be covered by the obstacle corresponding to the operations of crane 1 in r_1^2 and crane 2 in r_2^1.
- If $r_1^1 < r_1^2$ and $r_2^1 > r_2^2$, then (t_1, t_2) must be covered by the obstacle corresponding to the operations of crane 1 in r_1^2 and crane 2 in r_2^2.
- If $r_1^1 > r_1^2$ and $r_2^1 < r_2^2$, then (t_1, t_2) must be covered by the obstacle corresponding to the operations of crane 1 in r_1^1 and crane 2 in r_2^1.
- If $r_1^1 > r_1^2$ and $r_2^1 > r_2^2$, then (t_1, t_2) must be covered by the obstacle corresponding to the operations of crane 1 in r_1^1 and crane 2 in r_2^2.

This completes the proof. □

At this point we can specify the set of infeasible states as the union of interior points of obstacles corresponding to pairs of conflicting operations. We outline an example in Figure 7 and Figure 8.

We have five obstacles in Figure 8 corresponding to pairs of conflicting operations of cranes 1 and 2 that are carried out in rows 7 and 6, 7 and 5, 9 and 8, 9 and 6, and 9 and 5. As it can be seen the obstacles here may overlap. Also, obstacles in the same line (or column) may differ with respect to their height or width.

In analogy to the strategy developed in Section 4.1, we propose to design a path passing through the bottom border of each obstacle on the right of the diagonal that is not concealed with respect to the diagonal whose bottom border is higher than the diagonal's starting point. The term concealed in this case suggests that the horizontal connection from the diagonal to the obstacle would lead through an other obstacle. In the example in Figure 8 we can see that the diagonal starting in $(0, 0)$ is connected to the bottom border of four obstacles. Consequently, a path is established passing through the left border of each obstacle above

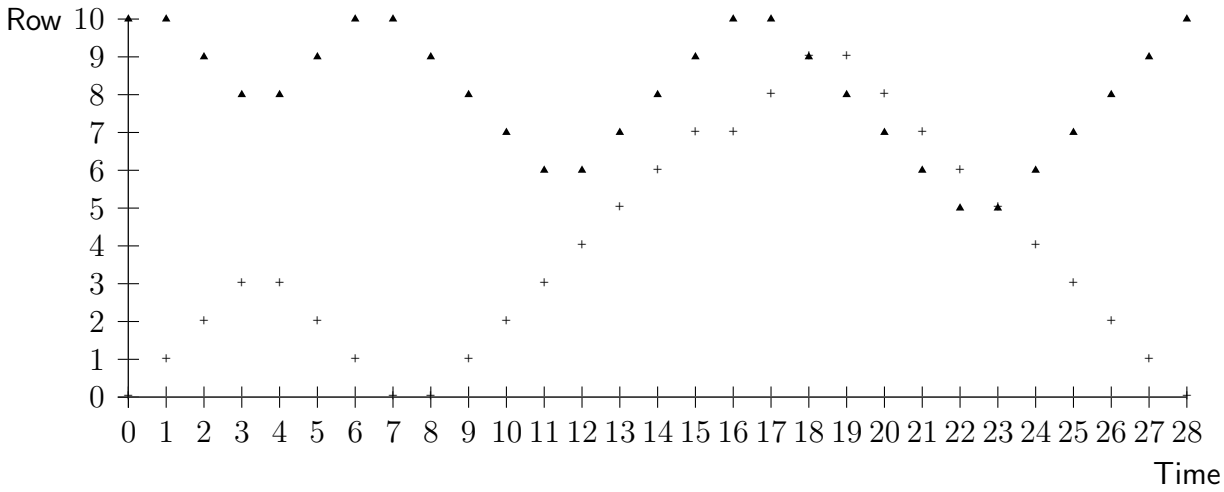

Figure 7: Movement graph for non-delay schedule of twin cranes

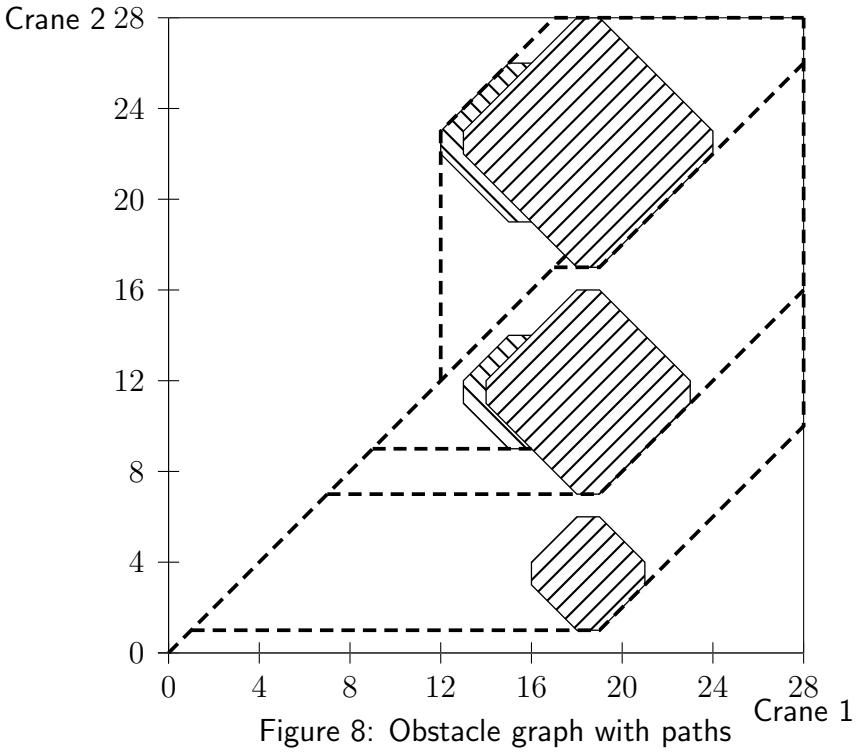

Figure 8: Obstacle graph with paths

the diagonal and not concealed with respect the diagonal whose left border is right of the starting point of the diagonal. Note that the diagonal starting in $(0, 0)$ is connected to one obstacle only since the diagonal passes above three obstacle and one the remaining obstacles is concealed with respect to the diagonal.

What remains to be shown is that it is sufficient to consider horizontal connections passing through the bottom borders of obstacles and vertical connections passing through left borders only. This can be shown easily by applying reasoning similar to the ones applied in Section 4.1.1. For the sake of shortness we do not go into detail here.

4.2.2 Schedules with detours

In this section we modify the graphical model developed in Section 4.2.1 in order to cover crane schedules including detours. First, we shall outline that indeed detours may improve schedules in terms of makespan minimization. We outline an example in Figure 9 and Figure 10.

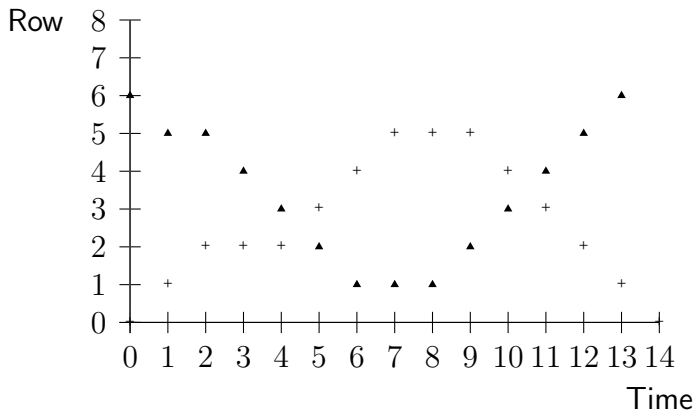

Figure 9: Movement graph for a twin crane non-delay schedule

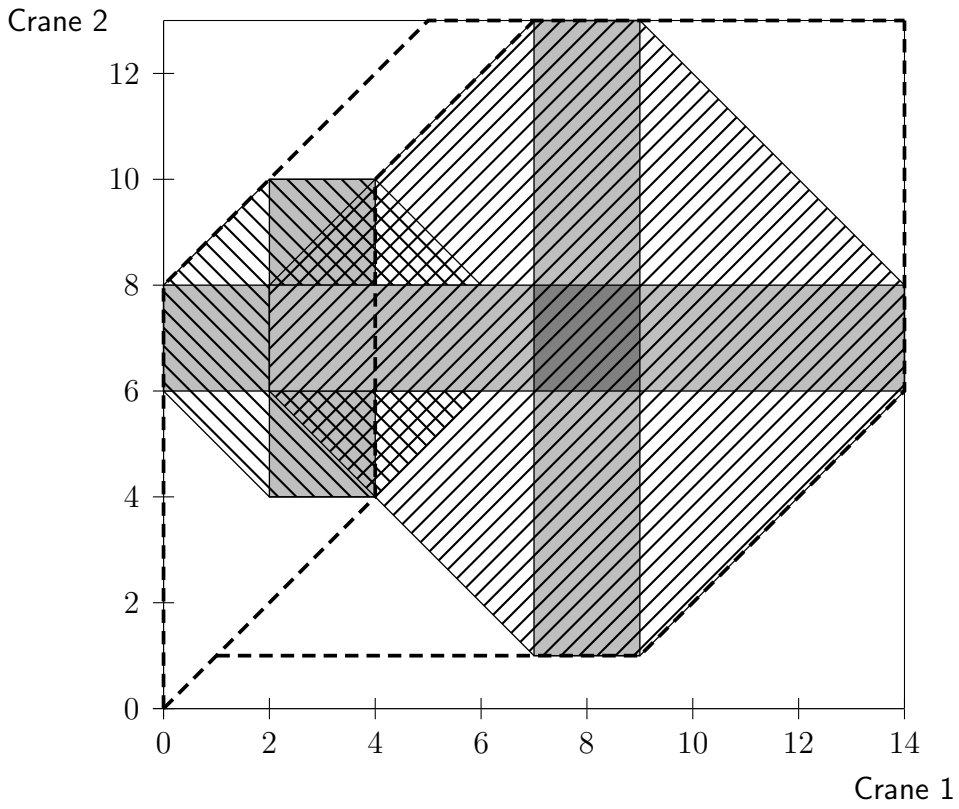

Figure 10: Obstacles graph with paths for the above example

The strategy outlined in Section 4.2.1 provides two options: The path moving below both overlapping obstacles has a length of 21 and the path that leads to the left of the obstacles

has a length 22. The third path depicted in Figure 10 corresponds to crane 1 operating in row 2 first, then dodging Crane 2 that operates in row 1, returning to row 2 as soon as crane 2 leaves, and, finally, moving to row 5 and returning to row 0. This path has length 20 and, thus, is the shortest path among these three.

In the following we state several properties of optimum schedules. We do so for Crane 1 being the crane making a detour. Properties for Crane 2 making a detour are strictly analog.

Lemma 5. *There is an optimum schedule such that*

1. *Crane 1 makes at most one detour between each pair of consecutive operations.*
2. *Crane 1 starts a detour between a pair of consecutive operations in rows r_1^1 and r_1^2*
   (a) *immediately before starting the second operation if $r_1^2 \leq r_1^1$ or*
   (b) *immediately after completing the first operation if $r_1^1 < r_1^2$.*
3. *Crane 1 returns to row r it started the detour in immediately after crane 2 completes an operation in row r.*

Proof. The first statement of the lemma is deduced as follows: Assume that crane 1 makes more than one detour in between two consecutive operations. Let r be the smallest row among those travelled on these detours. It can be seen easily that a single detour to r suffices in order to give way to crane 2.

The second part of the lemma can be justified as follows. We can restrict ourselves to detours starting either immediately after an operation has been completed or immediately before an operation is started since otherwise the crane runs into one direction and turns around at some point. Now, first, assume that crane 1 starts a detour immediately after completing the first operation and we have $r_1^2 \leq r_1^1$. Clearly, it can move to row r_1^2 before starting its detour since moving from r_1^1 to r_1^2 means giving way to crane 2. Second, assume that crane 1 starts a detour immediately before starting the second operation if $r_1^1 < r_1^2$. Then, the crane runs into one direction and turns around at some point which as mentioned before is not efficient. Instead it could either wait in r_1^1 and not make a detour at all if it does not impede crane 2 in row r_1^1 or start its detour from row r_1^1.

The third part of the lemma is easy to see since we assume that both cranes move with same speed. Thus, in the same period crane 2 leaves the row where crane 1 started its detour crane 1 can return to this row. □

We can represent detours according to Lemma 5 by an extension of the graphical model developed in Section 4.2.1.

- If a path passes below the lower right corner of part B of an obstacle corresponding to crane 1 operating in r_1^1 and the next operation of crane 1 takes place in row r_1^2, $r_1^2 \geq r_1^1$, we connect this path vertically to the upper right corner of T of this obstacle.
- If a path passes below the lower left corner of part B of an obstacle corresponding to crane 1 operating in r_1^1 and the previous operation of crane 1 takes place in row r_1^2, $r_1^2 \geq r_1^1$, we connect this path vertically to the upper left corner of T of this obstacle.

- If a path passes left of the upper left corner of part L of an obstacle corresponding to crane 2 operating in r_2^1 and the next operation of crane 2 takes place in row r_2^2, $r_2^2 \leq r_2^1$, we connect this path horizontally to the upper right corner of R of this obstacle.
- If a path passes left of the lower left corner of part L of an obstacle corresponding to crane 2 operating in r_2^1 and the previous operation of crane 2 takes place in row r_2^2, $r_2^2 \geq r_2^1$, we connect this path horizontally to the lower right corner of R of this obstacle.

Note that in the instructions above only paths as constructed in Section 4.2.1 are considered, that is we do not connect a section corresponding to a detour of crane 1 with a section corresponding to a detour of crane 2.

5 Algorithm Implementation and Computational Results

Following the basic idea provided in Brucker [2] we can represent the problem to find the shortest path in the networks constructed in Sections 4.1 to 4.2 as the problem to find a shortest path in a directed acyclic graph (DAG). We first give a generic description of the graph corresponding to each of the graphical models and discuss complexity issues afterwards.

We consider a weighted directed graph $G = (V, E, w)$ where

- $V = \{i \mid i$ is a starting point of a diagonal or $(C_1, C_2)\})$
- $E = \{(i, j) \mid i \in V, j \in V, i$ is directly connected to j in the network$\}$
- $w_e = w_e^{di} + w_e^{hv} + w_e^{de}$ where w_e^{di}, w_e^{hv}, and w_e^{de} is the length of the diagonal section, the length of the horizontal/vertical section, and the length of the detour section which e consists of (note that each of the lengths may equal zero)

First, we see that G is acyclic since $j^1 > i^1$ or $j^2 > i^2$ (possibly both) holds for each $e = ((i^1, i^2), (j^1, j^2)) \in E$. Thus, the shortest path in G can be found in $O(|E|)$ time. Clearly, we have $O(n^2)$ obstacles in each model. There are up to four starting points of diagonals associated to each obstacle. Thus, we have $|V| \in O(n^2)$. Each starting point of a diagonal section is connected to $O(n^2)$ other points. Hence, we have $|E| \in O(n^4)$ and we can solve the problems in $O(n^4)$ time. Note that, remarkably, the run time complexity does not depend on R. We conclude this analysis with the following theorem.

Theorem 2. *The minimum makespan schedule for both, crossover cranes and twin cranes, can be found in $O(n^4)$ time.*

The graph models have been therefore shown to provide strongly polynomial solution methods for the crane scheduling problems considered. To test the algorithm, we developed a software tool called CraneGraphs using C#, that translates job sequences into a schedule of movements. The tool proceeds as follows for a given pair of sequences of transport jobs:

- Using the sequences of jobs, construct a non-delay schedules for both cranes.

- Potential collisions between the two gantries are identified in accordance to the crane configuration that is in place.
- Create graph G for the crane configuration under consideration.
- Use Dynamic Programming to find a shortest path in G (recall that G is acyclic).
- Edges in the shortest path are translated into the corresponding segments in the graphical model which are in turn translated into a set of detours and movement delays that are injected into the original non-delay schedule.

We tested our approach on a moderately modern desktop computer (with a 3 GHz Intel Core 2 X9650 CPU and 8 GB of RAM). A series of computational experiments were carried out to test the performance of the algorithm on both ASC configurations mentioned in this paper. Figure 11 depicts a stylized block for describing our test set.

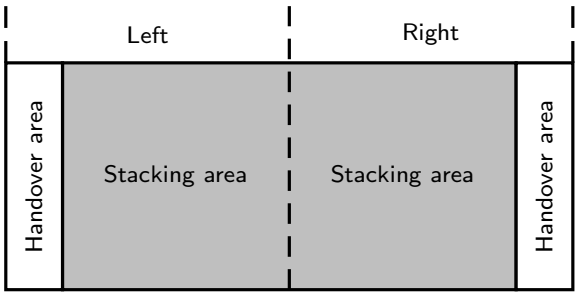

Figure 11: Test scenario

In our testbed we consider a block of 30 rows and both cranes being located in different handover areas at the beginning. Each single container move crosses the border between the left and the right part. Doing so we aim at a considerable number of potential conflicts. We generate different test sets by varying the start area of moves (handover areas/stacking areas) and the end area of move (handover areas/stacking areas). However, we do not consider moves from a handover area to the opposite handover area. Whenever we consider a start position or end position to be within one of the stacking areas we randomnly draw a row according to uniform distribution.

Furthermore, we vary the fraction of the overall number of containers going from left to right: all moves go from left to right, 75% of the moves go from left to right, or 50% of the moves go from left to right. Note that due to the symmetry of the setup this covers the cases where 25% of the containers or no containers go from left to right.

For each of these nine settings we consider test instances with 10, 20, 30, 40, or 50 containers and solve each of them for the crossover crane setting and the twin crane setting. In order to do so we make sure that whenever a handover area is either start point or end point of the move it gets assigned to the designated crane. For the settings where no handover area is involved we randomnly assign a container to one of the cranes. For each of 45 test sets we outline run times as well as relation of the optimum makespans found for both crane configurations.

		100%		75%		50%	
		ms	len	ms	len	ms	len
10	CO	0.00	1.00	0.00	0.85	0.01	0.74
	Twin	0.00	–	0.09	–	0.19	–
20	CO	0.00	1.00	0.19	0.83	0.78	0.72
	Twin	0.00	–	1.71	–	3.45	–
30	CO	0.00	1.00	1.90	0.80	8.37	0.75
	Twin	0.00	–	8.81	–	19.13	–
40	CO	0.00	1.00	9.73	0.82	42.10	0.76
	Twin	0.00	–	31.25	–	65.58	–
50	CO	0.00	1.00	32.80	0.83	129.96	0.77
	Twin	0.00	–	79.05	–	171.49	–

Table 1: Results for moves from handover area to stacking area

We present our results in Tables 1 and 2. Note that regarding the operations of cranes there is no difference between instances where we have only containers going from handover areas to stacking areas and instances where we have only containers going from stacking areas to handover areas. This is confirmed by our tests which is why we restrict ourselves to instances with containers moving from handover areas to stacking areas and 100%, 75% and 50%, respectively, of containers going from left to right.

We give average run times in ms and give the relation of schedules lengths (len) using the twin crane makespan as a basis. Note that the crossover crane makespan cannot be longer than the twin crane makespan since each feasible state for the crossover crane setting is feasible for the twin crane setting. Obviously, when we have a handover area involved and 100% of the moves going from left to right we obtain identical schedules since one of the cranes gets the full workload. In general, we can see that run times are well below a second for all test sets and below 0.1 sec when we do not consider more than 40 containers. The crossover makespan is between 70% and 87% of the twin crane makespan.

Table 1 gives the results for instances where containers are moved from handover areas to stacking areas. We observe that the relative gap between both makespans is rather independent from the instance sizes. However, it depends on the fraction of moves from left to right. If the difference in workloads is large, then the crane with the higher workload will get priority most of the time when a conflict occurs. The higher the difference in workloads is, the more likely it is that the other crane can afford to wait significant time periods. The higher flexibility implied by the crossover crane setting tends to get less beneficial then with respect to makespan minimization. However, even with a fairly unbalanced workload of one crane having 75% of the containers assigned the makespan is lower by 15% to 20% if the more flexible crossover cranes are employed. If the workload is rather balanced (in the 50% case) we have a makespan for crossover cranes which is shorter by 23% to 28% than the one of twin cranes. Having in mind the ever increasing pressure on terminal operators these numbers are quite substantial.

Recall that when no handover areas are involved we randomnly assign each container to one crane which leads to rather balanced workloads in all the cases in Table 2. Run times and relation of makespans behave similar for all three fractions of containers going from left to right. As we can see the gap between makespans seems to narrow down with increasing instance size. This may due to the following. There is a chance of both cranes going synchronously

		100%		75%		50%	
		ms	len	ms	len	ms	len
10	CO	0.03	0.74	0.03	0.70	0.02	0.74
	Twin	0.36	–	0.32	–	0.32	–
20	CO	0.94	0.76	0.68	0.74	0.53	0.71
	Twin	4.42	–	3.84	–	3.82	–
30	CO	8.23	0.78	5.55	0.81	4.33	0.75
	Twin	21.06	–	17.44	–	18.02	–
40	CO	30.75	0.82	24.00	0.81	18.26	0.81
	Twin	63.77	–	55.00	–	59.67	–
50	CO	92.30	0.84	65.01	0.84	56.37	0.87
	Twin	176.37	–	142.87	–	147.22	–

Table 2: Results for moves from stacking area to stacking area

back and forth (at least for some time). Of course it depends on the assignment of containers to cranes and their sequencing but chances are higher for it to happen for large instances. For the instances in Table 1 chances for that to happen are pretty low since whenever two cranes go synchronously in one direction they will split up immediately since of of the cranes goes all the way to the handover area. Even though the gap is narrowing down it is quite substantial (at least 13%) for instances with up to 50 containers (which we consider a huge instance regarding real world planning due to the very short term planning situation we encounter in our setting).

6 Summary and future work

Automated multi-gantry cranes are currently enjoying increasing adoption rates in modern container terminals, and the current outlook is that more terminal operators will implement such technologies in the years to come. While in practice each of the gantries will be focusing on different sides of the same stack interactions in the middle are common that are difficult to acknowledge in job allocation algorithms. This paper has presented a novel gantry scheduling algorithm that can determine optimal, precise and collision-free movement sets in real-time. The algorithm can accommodate any crane operation that can be represented in a movement graph, including container transfers to/from terminal vehicles and container reshuffling. Given its flexibility and sort execution times the approach is receptive for implementation in practice. We have investigated how the two systems under consideration, that is crossover cranes and twin cranes, relate to each other regarding the optimum makespan and we have seen that the more flexible crossover crane system can reduce the makespan of up to 30%. This obviously implies a benefit from crossover cranes in peak hours and emphasizes the need for sophisticated container assignments and sequencing methods.

There are two main directions for future research. First, future work may seek to embed this technique in job allocation algorithms that sequences container moves and co-ordinates crane activities with other terminal operations. Given the high-levels of accuracy and the fact that movement schedules can include any type of crane activity (not only container moves), the technique could also be used to develop scheduling methods that better synchronize AGV and

ASC operations. Second, future work may seek to generalize the approach to different crane setting, such as triple cranes or cranes with handover positions on the long side of the block, or to different objectives, such as minimization of lateness penalties for containers having delivery due dates.

Acknowledgement

The authors thank Amelie Eilken, University of Hamburg, for pointing out a flaw in an earlier version of the paper.

References

[1] S. Akers. A graphical approach to production scheduling problems. *Operations Research*, 4:244–245, 1956.

[2] P. Brucker. An efficient algorithm for the job-shop problem with two jobs. *European Journal of Operational Research*, 40(4):353–359, 1988.

[3] C. F. Daganzo. The crane scheduling problem. *Transportation Research B*, 23:159–175, 1989.

[4] U. Dorndorf and F. Schneider. Scheduling automated triple cross-over stacking cranes in a container yard. *OR Spectrum*, 32:617–632, 2010.

[5] W. W. Hardgrave and G. L. Nemhauser. A geometric model and a graphical algorithm for a sequencing problem. *Operations Research*, 11(6):889–900, 1963.

[6] D.-H. Lee, Z. Cao, and Q. Meng. Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm. *International Journal of Production Economics*, 107:115–124, 2007.

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

[8] W. C. Ng. Crane scheduling in container yardswith inter-crane interference. *European Journal of Operational Research*, 164:64–78, 2005.

[9] W. C. Ng and K. L. Mak. An effective heuristic for scheduling a yard crane to handle jobs with different ready times. *Engineering Optimization*, 37:867–877, 2005.

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

[11] D. Steenken, S. Voß, and R. Stahlbock. Container terminal operations and operations research – a classification and literature review. *OR Spectrum*, 26:3–49, 2004.

[12] I. F. A. Vis. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170:677–709, 2006.

[13] I. F. A. Vis and H. J. Carlo. Sequencing two cooperating automated stacking cranes in a container terminal. *Transportation Science*, 44:169–182, 2010.

[14] I. F. A. Vis and K. J. Roodbergen. Scheduling of container storage and retrieval. *Operations Research*, 57:456–467, 2009.