A Linear Time Algorithm for Optimal Quay Crane Scheduling*

Mathias Offerlin Herup, Gustav Christian Wichmann Thiesgaard, Jaike van Twiller, and Rune Møller Jensen

IT University of Copenhagen, Rued Langgaards Vej 7, 2300, Copenhagen S, Denmark
{ofhe,gthi,jaiv,rmj}@itu.dk

Abstract. This paper studies the Quay Crane Scheduling Problem (QCSP). The QCSP determines how a number of quay cranes should be scheduled in order to service a vessel with minimum makespan. Previous work considers the QCSP to be a combinatorially hard problem. For that reason, the focus has been on developing efficient heuristics. Our study shows, however, that the QCSP is tractable in the realistic setting, where quay cranes can share the workload of bays. We introduce a novel linear time algorithm that solves the QCSP and prove its correctness.

Keywords: Quay crane scheduling · Container terminals · Computational complexity

1 Introduction

In 2020 the global export of merchandise amounted to US$ 17.6 trillion [12] of which maritime transportation and container shipping accounted for roughly 80% [9]. Due to the significant size of the industry, there is a constant need for improvement, as even minor optimizations may yield massive economic results. That being said, container shipping is a complex industry consisting of several individual parties monitoring a number of sub-processes of their own.

Two central players are the container terminals and the carriers. The terminals act as links between land and sea, while the carriers transport containers between terminals. The key interface between them are quay cranes that load and discharge containers to vessels. Both terminals and carriers benefit from a minimum makespan of quay cranes operating a vessel [5].

The Quay Crane Scheduling Problem (QCSP) explores how a number of quay cranes should be assigned and organized in order to service a vessel such that a minimum makespan of the quay cranes is achieved. Quay cranes must adhere to several ordering and separation constraints. Due to these constraints, the QCSP has been considered a hard combinatorial optimization problem since its introduction by Daganzo [2]. In 2006, Lee, Wang, and Miao [7] proved a specific version of the problem called the QCSNIP to be *NP*-complete. Due

* This research is sponsored in part by the Danish Maritime Fund under grant no. 2021-069.

to this complexity result, the use of heuristic algorithms in subsequent work has been widely justified as computing an optimal schedule would be too time consuming in practice, e.g., [1,10].

In contrast to the QCSP, however, the QCSNIP assumes that a bay at most can be serviced by a single quay crane throughout the whole schedule. This is an unrealistic assumption. It is common practice of terminals to have several quay cranes operating a bay during a schedule. They are merely restricted from operating a bay or a pair of adjacent bays simultaneously [5].

In this paper, we show that the QCSP is tractable in the realistic setting, where quay cranes can share the workload of bays. We introduce an algorithm that solves the QCSP to optimality in linear time and prove its correctness. Our results imply that it may be possible to formulate tractable and optimal quay crane scheduling algorithms that model all aspects of the real problem including variable separation distances, quay crane performance dependency on lift type, and time to move quay cranes between bays.

The remainder of the paper is organized as follows. Section 2 provides a brief introduction to the terminal domain. Section 3 reviews the existing literature relevant to the scope of this paper. Section 4 defines the QCSP. Section 5 discusses the validity of the QCSNIP model and corresponding proof as proposed by Lee et al. Furthermore, a linear time algorithm guaranteeing optimal solutions is proposed and its correctness is mathematically proven.

2 Container Terminals

Container terminals interact with carriers through service contracts. The contracts may include a berthing time window, a guaranteed number of crane moves to be processed during the port stay, and various handling fees. However, service contracts do not specify the number of quay cranes assigned to work on a given vessel, i.e., it is up to the terminal to decide the quay crane schedule as long as it ensures the guaranteed number of containers moved in the given berthing time window. As port fees are based on the time a vessel spends in a port, it is in a carrier's best interest to minimize this stay. In addition to a lower port fee, a shorter stay would enable a vessel to catch up on delays or sail at lower speeds, ultimately resulting in higher profits. Terminals also benefit from minimizing port stays since the resulting high utilization of equipment assets results in cost-efficient services [5].

A container terminal serves as a hub for shipment of containers. Containers are either transported to the terminal from the sea side on vessels or from the land side on trucks and trains. When containers are intermediately stored at the terminal, they are stored in the yard. The yard is an area designated to store containers in an organized manner. Containers are transported as needed from the yard to the quay when a vessel is being loaded and vice versa when containers are being discharged from the vessel. Figure 1 shows a typical terminal organization.

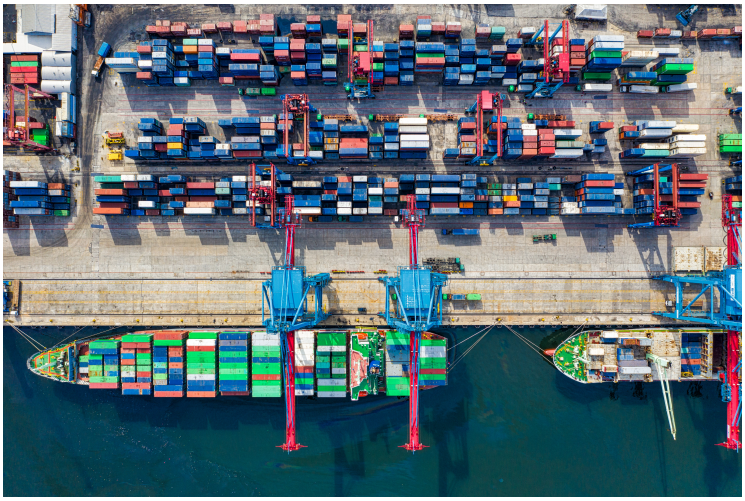

Fig. 1. A container terminal as seen from above (photo: Tom Fisk, 2019).

A container vessel is a large ship specifically made for transporting containers. The layout of a vessel is split into bays, which are spaces on the ship that can store containers. Containers are discharged and loaded onto a vessel using quay cranes as illustrated in Figure 2. All quay cranes of a berth section are mounted on a single track of rails which runs along the wharf. As quay cranes are all mounted on the same track, they can freely move to each side but are unable to pass each other. Moreover, quay cranes are wider than vessel bays meaning that two, and in rare cases three, adjacent bays cannot be operated by two quay cranes simultaneously.

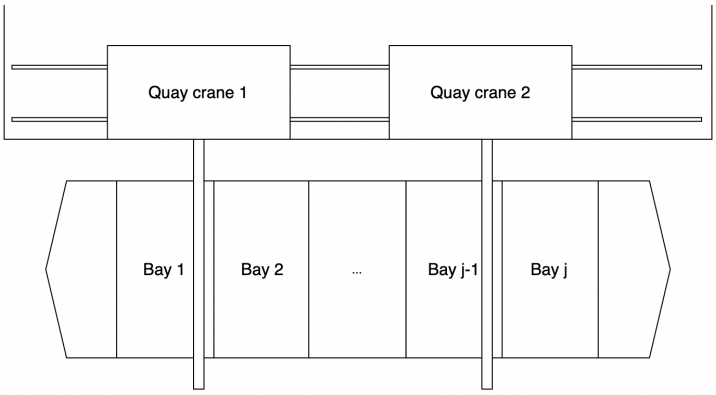

Fig. 2. A vessel being serviced by quay cranes.

There exists a plethora of techniques and equipment that enable quay cranes to load and discharge several containers at a time, e.g., twin-lifting, quad-lifting, and dual cycling. The move productivity of the quay cranes depends on the lift type and the skills of the human crane operator as well as the stowage condition of the vessel. If quay cranes must move over high stacks, it slows them down. In addition to moving containers, the quay cranes also spend time changing position between bays, known as *crane sets*. Normally this takes a few minutes, but if a quay crane passes the accommodation section, it must lift its arm possibly adding half an hour to the operation [5].

Quay crane scheduling is the problem of managing the movement of quay cranes, such that they operate as efficiently as possible. Part of this problem is determining the number of quay cranes to assign to a vessel based on the minimum number of crane moves defined in the service contract. Once this number is determined, a schedule is produced. The schedule describes where quay cranes should be positioned, which containers to load and discharge and when to execute these operations. The objective of the schedule is to process the vessel as fast as possible while using as few resources as possible. Producing an optimal schedule is considered a complex problem as there, as previously mentioned, are several real life limitations that restrict the movement and operation of quay cranes [5].

3 Literature Review

The scope of this paper is limited to the quay crane scheduling problem (QCSP) and its various formulations in previous literature. The QCSP was first introduced by Daganzo [2]. The paper focuses on the general problem of processing all arriving ships while minimizing delays. Exact and approximate solutions are presented without considering operational constraints such as non-interference and non-neighbouring constraints; constraints which prevents quay cranes from passing each other on the same quay track and working on two adjacent bays at the same time respectively.

Hereafter, Kim and Park [6] introduce non-neighbouring constraints, non-interference constraints, precedence constraints, i.e., some operations precede others and task-separation constraints, i.e., some tasks cannot be done simultaneously. The study formulates a mixed-integer programming model and proposes a branch-and-bound ($B\&B$) method in order to find an optimal solution. Additionally, a GRASP metaheuristic is proposed to overcome the computational difficulty of the $B\&B$ method.

Subsequently, a paper stimulated by the work of Kim and Park [6] was written by Lee et al. [7] in which a concise mathematical model for the quay crane scheduling with non-interference constraints problem (QCSNIP) is formulated. However, for unclear reasons, Lee et al. omit the non-neighbouring constraint introduced by Kim and Park and imposes that the workload of a bay cannot be split between quay cranes. Moreover, they analyse the computational complexity of the problem and provide a proof of NP-completeness. In addition,

the paper proposes a genetic algorithm that provides efficient performance and near-optimal solutions.

Hereafter, Moccia, Cordeau, Gaudioso, and Laporte [8] observed that Kim and Park's solutions [6] do not guarantee non-interference for every instance. Following this, they introduce travel time of quay cranes and propose a revised mixed-integer programming model along with a branch-and-cut algorithm ($B\&C$) to handle inputs with large solution spaces. A compact mathematical formulation of this model was subsequently proposed by Sammarra et al. [10].

Eventually, Bierwirth and Meisel [1] found that the revised model proposed by Sammarra et al. [10] may also obtain solutions that violate non-interference constraints, which is resolved in their study by the introduction of temporal distance constraints between tasks. Consequently, optimality of the solution would in some cases not be preserved. A heuristic that applies a $B\&B$ algorithm for searching a subset of above average quality, unidirectional schedules is proposed. Using the benchmark suite from Kim and Park [6], the heuristic found the best known solution in every problem instance and computational effort was cut down to a fraction of other methods proposed in previous work.

Subsequently, Fan, Low, Ying, Jing, Min, and Aye [3] introduce a key performance indicator called *Crane Intensity* (CI). CI indicates how well the workload is distributed among quay cranes with regards to minimum idle and movement time, and is thus argued to be an indicator of the quality of a quay crane schedule.

4 Problem Definition

This section defines the QCSP that is the scope of this paper. The QCSP is restricted by the following constraints.

1. Quay cranes move on the same track and thus cannot pass each other (*non-interference*).
2. Only one quay crane can work on a given bay at a time.
3. Quay cranes cannot work on two adjacent bays simultaneously (*non-neighbouring constraint*).

In addition to these constraints, we assume that time is discretized into equally sized time steps. In each time step a crane can make exactly one move, i.e., either loading a container to a bay or discharging a container from a bay. We further assume that it takes no time to move cranes between bays. Under these assumptions, an instance of the QCSP is defined by the following parameters.

B	Total number of bays
C	Total number of available quay cranes
m_j	Total number of containers to be moved in bay j $(1 \leq j \leq B)$

A solution to a QCSP is referred to as a *schedule*. A schedule defines for each time step which bay each crane is assigned to. A schedule is feasible if the constraints of the QCSP are satisfied and the crane assignment allows all

moves of each bay to be processed. The total number of time steps of a schedule is referred to as its *makespan MS*. A schedule for an instance of the QCSP is optimal if it is feasible and no other feasible schedule for the instance has a smaller *MS*. As an example, Figure 3 shows a feasible and optimal schedule for a QCSP instance with $B = 7$, $C = 3$, and the total number of containers to be moved for each bay as shown in the figure, i.e., $m_1 = 2$, $m_2 = 5$, etc..

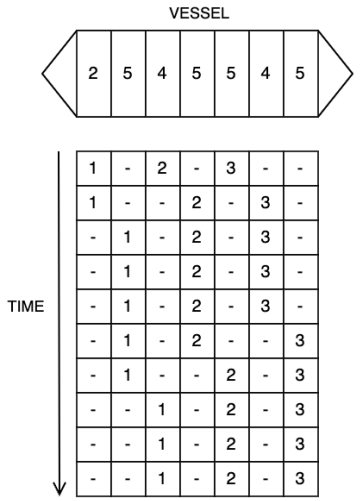

Fig. 3. An example of a feasible schedule of a QCSP instance.

5 Problem Complexity

Since its introduction by Daganzo [2], the QCSP has been considered a hard combinatorial problem. Lee et al. show that a special decision version of the QCSP, called the quay crane scheduling with non-interference constraints problem (QCSNIP), is NP-complete. The QCSNIP is restricted by the following constraints.

1. Quay cranes move on the same track and thus cannot pass each other (*non-interference*).
2. Only one quay crane can work on a hold[1] at a time until it completes the hold.
3. Compared with processing time of a hold by a quay crane, travel time of a quay crane between two holds is small and hence it is ignored.

[1] The hold is the part of a bay, which is under deck. In this setting, a hold can be considered to be the same as a bay.

Lee et al. prove that the QCSNIP is NP-complete by reducing the set partitioning problem to it. The decision version of the set partitioning problem is defined as follows [4]; given a finite set of H positive integers, $S = \{s_1, s_2, ..., s_H\}$, where the sum of all elements is equal to D; can S be partitioned into two disjoint subsets S_1 and S_2, such that $\sum_{s_i \in S_1} = \sum_{s_i \in S_2} = D/2$?

Lee et al. translate this problem into the QCSNIP by defining a hold for each element in S and some auxiliary holds. The set partitioning problem then becomes to find a schedule for two cranes, where the cranes work on subsets of the holds corresponding to S_1 and S_2. The proof is innovative and to our knowledge correct. Our main concern is that the QCSNIP substantially differs from the QCSP such that the complexity result of the QCSNIP cannot be transferred to the QCSP. In other words, the NP-completeness of the QCSNIP does not prove NP-hardness of the QCSP.

The most important limitation is that the QCSNIP assumes that *only one quay crane can work on a hold at a time until it completes the hold*, i.e., assumption 2. By constraining the workload of any bay to be processed to completion by one crane and one crane only, it is possible to let a bay of a vessel correspond to an element of a set in the set partitioning problem. However, this is not a realistic assumption. Several quay cranes often share the workload in a single bay, but cannot do so at the same point in time [5,11]. Without this restriction, Lee et al.'s reduction of the set partitioning problem to the QCSNIP is invalid.

Lee et al. also omit the non-neighbouring constraint, which ensures that a pair of cranes will not simultaneously work on any pair of adjacent bays. However, the removal of this constraint may not necessarily invalidate the reduction, as it may be possible to add "separation" bays to their reduction without invalidating it. Since the QCSP allows the workload of a bay to be split between quay cranes, Lee et al.'s NP-completeness proof does not apply to a decision version of the QCSP. In fact, below we show that the QCSP is tractable.

In the remainder of this section, we introduce the CreateSchedule algorithm that can solve the QCSP to optimality in time that is linear in $MS \times C$, where MS is the minimum makespan and C is the number of cranes assigned to the vessel.

5.1 Makespan Lower Bounds

We first define lower bounds of the makespan MS as a function of the number of cranes C assigned to the vessel.

The lower limit to MS is defined as the largest sum of moves of any two adjacent bays, as only one bay of these adjacent bays can be processed by a single crane at any given time. We refer to this pair of bays as the long crane, LC, and the number of moves in the pair, m_{LC}, is defined in equation (1).

$$m_{LC} = max(m_j + m_{j+1}) \quad 1 \leq j < B. \tag{1}$$

To achieve an optimal schedule with MS equal to m_{LC}, a sufficient amount of quay cranes is required such that (i) a single quay crane can always operate the

LC at all times and (ii) all other bays are completed in a number of moves less than or equal to m_{LC}. This means that the sum of all moves of the vessel M divided by the number of quay cranes needed, C_n, must be smaller than or equal to m_{LC} in order to optimally complete the vessel, as shown in equation (2).

$$\frac{M}{C_n} \leq m_{LC}, \tag{2}$$

Equation (2) can be rewritten into equation (3).

$$\frac{M}{m_{LC}} \leq C_n. \tag{3}$$

The number of quay cranes that can efficiently work in parallel on a given vessel is referred to as crane intensity, CI and is defined in equation (4).

$$CI = \frac{M}{m_{LC}}. \tag{4}$$

Thus, as shown in equation (5), if a sufficient number quay of cranes, C, are available for the processing of a given vessel ($CI \leq C$) the lowest possible makespan will be defined by m_{LC}.

$$\textit{If } CI \leq C, \textit{ then } MS = m_{LC}. \tag{5}$$

On the other hand, if an insufficient number of quay cranes are available for the processing of a given vessel (i.e., $CI > C$), the lowest possible makespan is defined by the ceiling of the total number of moves of the vessel divided by the number of available cranes as shown in equation (6).

$$\textit{If } CI > C, \textit{ then } MS = \left\lceil \frac{M}{C} \right\rceil. \tag{6}$$

5.2 The CreateSchedule Algorithm

Next, we define a linear time schduling algorithm for the QCSP called CreateSchedule. It returns schedules that achieve the lower bounds of the makespan defined above and for that reason is optimal. The pseudo code of CreateSchedule is shown below. It returns a table of size $B \times MS$ containing positions of all cranes to all points in time.

```
1  function CreateSchedule(C, B, [m_1, m_2...m_B])
2      returns: table of size B × MS containing positions of cranes
3               to all points of time.
4      inputs: C, the number of available cranes
5              B, the number of bays of the vessel
6              [m_1, m_2...m_B], list containing amount of moves of each bay
7
8      if CI ≤ C then
9          MS ← m_LC
10     else
11         MS ← ⌈M/C⌉
12
13     schedule ← empty table of size B × MS
14     k ← 1
15     for i = 1 to C
16         for j = 1 to MS
17             while m_k = 0
18                 k ← k + 1
19             schedule[k][j] ← i
20             m_k ← m_k − 1
21     return schedule
```

The algorithm assumes that cranes and bays are sorted in an increasing numerical order from left to right as illustrated in Figure 2. Initially, it calculates MS (line 8−11) in relation to CI as described in Section 5.1. Following this, every crane is assigned a workload equal to the calculated MS. Through *first iteration* of the inner and outer for-loop (line 15−20), the first crane with $i = 1$ is initially assigned a single move in the leftmost non-empty bay of the vessel. The remaining iterations of the inner for-loop (line 16−20) assigns crane 1 a workload equal to MS, such that it will be assigned as many moves as possible in the subsequent, non-empty bays. Note that the algorithm will confirm if a bay is empty in order to continue with non-empty bays (line 17−18). Any further iterations of the outer for-loop will assign any subsequent cranes, which will initially be assigned the bay where the previous crane finished. Hereafter, the process of assigning moves to quay crane is continued in an identical fashion, such that the crane can process at most MS moves.

In relation hereto, as M might not be divisible by MS, it may not be possible to assign every crane a workload equal to this. As a result, the last iteration of the outer-loop, corresponding to the the last crane, might not execute MS iterations of the inner loop. As there might not be enough moves left in the vessel, this results in this crane having a smaller workload than MS.

The time complexity of CreateSchedule is dependent on the calculated makespan MS, which can be calculated in constant time. When assigning cranes to bays, MS defines the amount of iterations of the inner loop (line 16−20). In addition, the amount of cranes C define the number of iterations of the outer loop (line 15−20). By extension, the time complexity of CreateSchedule is $O(C \times MS)$.

Example
Given a vessel with $M = 30$, $C = 3$ and $MS = 10$, all cranes are assigned a workload of 10 moves. Crane 1 is assigned the leftmost non-empty bay of the vessel, i.e., bay 1 as shown in Figure 4.

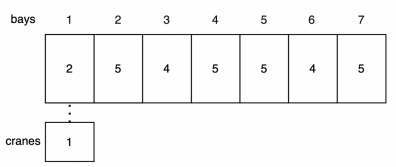

Fig. 4. Crane 1, assigned bay 1.

Hereafter, crane 1 is assigned workload from non-empty bays until it reaches its capacity at bay 3 as shown in Figure 5.

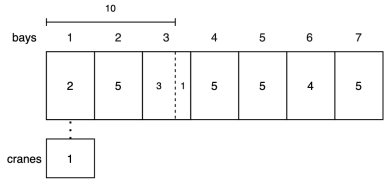

Fig. 5. Crane 1 starts in bay 1 and finishes in bay 3.

The subsequent crane 2 is initially assigned to bay 3 as crane 1 will not complete it entirely. Crane 2 has its workload assigned in the subsequent, non-empty bays until it reaches its capacity at bay 5. The following crane 3 is initially assigned bay 5, as crane 2 will not complete it entirely. Crane 3 has its workload assigned in the subsequent, non-empty bays until it reaches its capacity at bay 7 as shown in Figure 6.

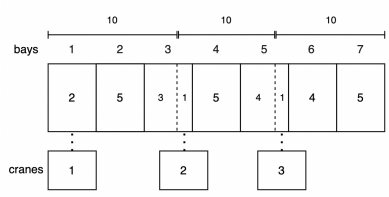

Fig. 6. All cranes are assigned.

Thus, all bays of the vessel are completed, resulting in a complete schedule as presented in Figure 7.

INPUT:
C = 3

VESSEL:	2	5	4	5	5	4	5

OUTPUT:

t_1	1	-	2	-	3	-	-
t_2	1	-	-	2	-	3	-
t_3	-	1	-	2	-	3	-
t_4	-	1	-	2	-	3	-
t_5	-	1	-	2	-	3	-
t_6	-	1	-	2	-	-	3
t_7	-	1	-	-	2	-	3
t_8	-	-	1	-	2	-	3
t_9	-	-	1	-	2	-	3
t_{10}	-	-	1	-	2	-	3

Fig. 7. An example of a schedule.

5.3 Correctness Proof

CreateSchedule does not explicitly prevent interference of cranes, which potentially leads to the constraints of the problem being broken. In other words, it remains to prove that the algorithm's assignment of cranes and workload does not violate non-interference and non-neighbouring constraints.

Proposition. *A schedule produced by* CreateSchedule *does not violate the non-interference and non-neighbouring constraints of the QCSP.*

Proof. For two arbitrary adjacent cranes $i-1$ and i, it holds that their workload will only overlap in a single bay j. If crane $i-1$ is attending bay $j-1$, the bay to the left of bay j, then crane i is not allowed to attend bay j. Let the moves in bay j processed by crane i be represented by m_j^i. We have that m_j^i must be less than or equal to the sum of moves processed by crane $i-1$ from its starting bay s_{i-1} to bay $j-2$ as shown in equation (7). As a consequence, crane $i-1$ and i will not violate non-interference and non-neighbouring constraints.

$$m_j^i \leq \sum_{k=s_{i-1}}^{j-2} m_k^{i-1} \tag{7}$$

The moves processed by crane i in bay j are equal to the sum of moves in the pair m_{j-1} and m_j subtracted by the number of moves processed by crane $i-1$ in the pair, m_{j-1}^{i-1} and m_j^{i-1}, as shown in equation (8).

$$m_j^i = (m_{j-1} + m_j) - (m_{j-1}^{i-1} + m_j^{i-1}) \tag{8}$$

Considering equation (8) above, let us define the two different cases of MS.

First, consider the case where MS is defined by m_{LC} (line 8–9). The work processed by a crane $i-1$ before arriving at the pair of bays, $j-1$ and j, is defined in equation (9), and is equal to the total capacity of a crane, m_{LC} subtracted by the moves to be processed in the pair, by crane $i-1$.

$$\sum_{k=s_{i-1}}^{j-2} m_k^{i-1} = m_{LC} - (m_{j-1}^{i-1} + m_j^{i-1}) \tag{9}$$

When substituting equation (8) and (9) into equation (7), we get equation (10) that can be rewritten into equation (11).

$$(m_{j-1} + m_j) - (m_{j-1}^{i-1} + m_j^{i-1}) \leq m_{LC} - (m_{j-1}^{i-1} + m_j^{i-1}) \tag{10}$$

$$(m_{j-1} + m_j) \leq m_{LC} \tag{11}$$

Equation (11) is true by the definition of LC as there cannot exist any pair of bays of which the sum of moves exceed m_{LC}.

Second, consider the case where MS is defined by $\lceil \frac{M}{C} \rceil$ (line 10–11). The work processed by a crane $i-1$ before arriving at the pair of bays, $j-1$ and j, is defined by equation (12), and is equal to $\lceil \frac{M}{C} \rceil$ subtracted by the moves to be processed in the pair, by crane $i-1$.

$$\sum_{k=s_{i-1}}^{j-2} m_k^{i-1} = \left\lceil \frac{M}{C} \right\rceil - (m_{j-1}^{i-1} + m_j^{i-1}) \tag{12}$$

When substituting equation (8) and (12) into equation (7), we get equation (13) that can be rewritten into equation (14).

$$(m_{j-1} + m_j) - (m_{j-1}^{i-1} + m_j^{i-1}) \leq \left\lceil \frac{M}{C} \right\rceil - (m_{j-1}^{i-1} + m_j^{i-1}) \tag{13}$$

$$(m_{j-1} + m_j) \leq \left\lceil \frac{M}{C} \right\rceil \tag{14}$$

Equation (14) is true by the definition of CI. If we have fewer cranes than the crane intensity suggests, then splitting M over all cranes would result in the average workload of all cranes, i.e., $\lceil \frac{M}{C} \rceil$, being greater than m_{LC}.

Thus, given any vessel and any number of cranes, the non-interference and non-neighbouring constraints of the QCSP are satisfied by CreateSchedule. ■

5.4 Extensions of the QCSP

The QCSP is more realistic than the QCSNIP, but as described in Section 2, real quay crane scheduling problems must take into account that time is spent moving between bays and that the productivity of cranes depends on the lift type. It is an open question whether these extensions of the QCSP makes it NP-hard. However, we do not see any obvious reasons that it should be the case.

Another issue is the operational scope of the QCSP. It does not consider in which order containers are picked from the yard of the terminal and brought to the quay cranes. From the terminal's point of view, this an important aspect of scheduling quay cranes as the schedule depends on where the containers are stored in the yard.

From the carriers point of view, on the other hand, the QCSP makes perfect sense. Recall that the relation between terminals and carriers is contractual. Typically, the terminal guarantees a total number of moves within a given time window [5]. In a situation, where the terminal fails to finish the vessel within the agreed time window, the carrier wants to be able to assess whether a makespan exists within this time window. The QCSP can answer this question, and it is perfectly fine for the carrier to ignore how the containers are handled on the yard as this is the terminal's problem. For that reason, modern stowage tools used by carriers include heuristics that are able to compute so-called *crane splits*. These crane splits corresponds to solutions of the QCSP [5].

6 Conclusion

In this paper, we show that the QCSP is tractable in the realistic setting, where quay cranes can share the workload of bays. This is done by proposing a linear time algorithm that finds optimal solutions to the QCSP and providing a proof of its correctness. Our findings imply that it may be possible to formulate tractable and optimal quay crane scheduling algorithms that model additional aspects of the real problem including variable separation distances, quay crane performance dependency on lift type, time to move quay cranes between bays, and varying processing time of containers.

Directions of future work include extending the QCSP with travel time of quay cranes, relaxing the single move per unit of time assumptions, processing time for various types of containers, and incorporating handling priority of cargo.

References

1. Bierwirth, C., Meisel, F.: A fast heuristic for quay crane scheduling with interference constraints. Journal of Scheduling **12**(4), 345–360 (2009)
2. Daganzo, C.F.: The crane scheduling problem. Transportation Research Part B: Methodological **23**(3), 159–175 (1989)
3. Fan, L., Low, M.Y.H., Ying, H.S., Jing, H.W., Min, Z., Aye, W.C.: Stowage planning of large containership with tradeoff between crane workload balance and ship stability. In: World Congress on Engineering 2012. July 4-6, 2012. London, UK. vol. 2182, pp. 1537–1543. International Association of Engineers (2010)

4. Garey, M.R., Johnson, D.S.: Computers and intractability, vol. 174. freeman San Francisco (1979)
5. Jensen, R.M., Pacino, D., Ajspur, M.L., Vesterdal, C.: Container Vessel Stowage Planning. Weilbach, Copenhagen, Denmark (2018)
6. Kim, K.H., Park, Y.M.: A crane scheduling method for port container terminals. European Journal of operational research **156**(3), 752–768 (2004)
7. Lee, D.H., Wang, H.Q., Miao, L.: Quay crane scheduling with non-interference constraints in port container terminals. Transportation Research Part E: Logistics and Transportation Review **44**(1), 124–135 (2008)
8. Moccia, L., Cordeau, J.F., Gaudioso, M., Laporte, G.: A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. Naval Research Logistics (NRL) **53**(1), 45–59 (2006)
9. Psaraftis, H.N.: The future of maritime transport. In: Encyclopedia of Transportation. Elsevier (2020)
10. Sammarra, M., Cordeau, J.F., Laporte, G., Monaco, M.F.: A tabu search heuristic for the quay crane scheduling problem. Journal of Scheduling **10**(4), 327–336 (2007)
11. Stahlbock, R., Voß, S.: Operations research at container terminals: a literature update. OR spectrum **30**(1), 1–52 (2008)
12. on Trade, U.N.C., Development: Review of maritime transport 2020. United Nations, 1 edn. (2021)