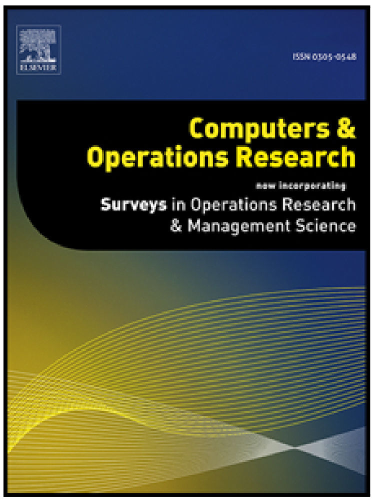
Accepted Manuscript



Exact methods for the quay crane scheduling problem when tasks are modeled at the single container level

Mohamed Kais Msakni, Ali Diabat, Ghaith Rabadi,
Mohamed Al-Salem, Mariam Kotachi

PII:	S0305-0548(18)30189-8
DOI:	10.1016/j.cor.2018.07.005
Reference:	CAOR 4512
To appear in:	*Computers and Operations Research*
Received date:	23 January 2018
Revised date:	29 June 2018
Accepted date:	5 July 2018

Please cite this article as: Mohamed Kais Msakni, Ali Diabat, Ghaith Rabadi, Mohamed Al-Salem, Mariam Kotachi, Exact methods for the quay crane scheduling problem when tasks are modeled at the single container level, *Computers and Operations Research* (2018), doi: 10.1016/j.cor.2018.07.005

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

Highlights

- The scheduling problem of quay cranes in container terminal is studied.
- Different technical constraints related to the quay cranes are considered.
- Two exact methods are proposed to provide optimal schedules.
- A construction heuristic is developed to provide near-optimal solutions.
- Computational experiments show the efficiency of the proposed methods.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Exact methods for the quay crane scheduling problem when tasks are modeled at the single container level

Mohamed Kais Msakni[a,*], Ali Diabat[b,c], Ghaith Rabadi[d], Mohamed Al-Salem[a], Mariam Kotachi[d]

[a]*Department of Mechanical & Industrial Engineering, Qatar University, Doha, Qatar.*
[b]*Division of Engineering, New York University Abu Dhabi, Saadiyat Island, 129188, Abu Dhabi, United Arab Emirates.*
[c]*Department of Civil & Urban Engineering, Tandon School of Engineering, New York University, Brooklyn, NY 11201, United States of America.*
[d]*Department of Engineering Management & Systems Engineering, Old Dominion University, Norfolk, VA 23529, United States of America.*

Abstract

The scheduling of quay cranes (QCs) to minimize the handling time of a berthed vessel is one of the most important operations in container terminals as it impacts the terminal's overall productivity. In this paper, we propose two exact methods to solve the quay crane scheduling problem (QCSP) where a task is defined as handling a single container and subject to different technical constraints including QCs' safety margin, non-crossing, initial position, and nonzero traveling time. The first method is based on two versions of a compact mixed-integer programming formulation that can solve large problem instances using a general purpose solver. The second is a combination of some constraints of the proposed mathematical model and the binary search algorithm to reduce the CPU time, and solve more efficiently large-sized problems. Unlike existing studies, the computational study demonstrates that both methods can reach optimal solutions for large-sized instances and validates their dominance compared to an exact model proposed in the literature which finds solutions only for small problems.

Keywords: Scheduling, Container terminals, Quay crane scheduling problem, Mixed-integer programming

1. Introduction

The recent rapid growth in the global economy in the last two decades would not have been possible without the increased capacity and efficiency of maritime systems including ports. Transporting goods by water remains by far the most economical mode of transportation. Governments and industries have been

*Corresponding author
Email address: `msakni.kais@qu.edu.qa` (Mohamed Kais Msakni)

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

investing heavily in expanding and modernizing ports' infrastructures, the standardization of equipment, and implementation of new technologies. Examples of deep-sea ports with high capacity include ports of Singapore, Shanghai, Jebel Ali in the United Aarab Emirates, and recently Hamad Port in Qatar. Managing operations at such mega ports can be very complex, and when most ports are part of a global economy, the speed at which products move at ports becomes very important as inefficiency is very costly in such systems. However, it is not feasible to continuously add resources to ports to increase their throughput because such resources are capital intensive, and in many cases, require space that can be scarce. Therefore, it is important to optimize the existing resources and increase efficiencies and throughput by managing operations intelligently.

In this paper, we focus on the scheduling of the quay cranes (QCs), which are dockside gantry cranes that load and unload containers onto and off of vessels and constitute one of the most expensive pieces of equipment in any port. Hence, the objective is to minimize the total time QCs take to process a vessel. In fact, their efficiency is considered a determining factor in measuring the overall terminal productivity according to Chung and Chan (2013).

There are two main decisions to make with regard to QCs: the first is assigning cranes to berthed vessels, which is known as the Quay Crane Assignment Problem (QCAP) and the second is partitioning the tasks of loading/unloading containers between all assigned QCs and determining their processing sequence. The latter is known as the quay crane scheduling problem (QCSP).

QC assignment to vessels is performed by a port operator after allocating vessels to berths at specific times. This process considers different factors including the type of containers, crane availability, and the possibility of moving a QC from one vessel to another during the service process. Effective allocation and scheduling of QCs can significantly increase terminals' throughput as they reduce vessels' berthing times.

Before the work on a vessel begins, the list of containers to be loaded/ unloaded is provided by a shipping company. This list is known as the *load profile* which specifies the containers to be operated as well as their location. The containers are moved from the vessel to trucks or the yard, and vice versa using QCs which are mounted on a single rail track to move alongside the quay without passing each other. This limitation is commonly enforced through non-crossing constraints. In addition, and for safety reasons, two adjacent QCs have to keep a minimum separation distance during the work, referred to as safety margin. A single task of a QC differs from an operation manager to another. In some cases, it is defined as a cluster of containers located in the same area of the vessel resulting in what is known as *container group* operations. In other cases, one *bay* (segment) of a vessel is considered as one block and a task refers to processing all containers of the same bay before the QC moves to another bay. This type is known as *complete bays.* Finally, there is a case which is referred to as *container* operation where a task is related to only one container. In this configuration, a QC can move to another bay without completing the processing of all containers of the current bay. As a result the tasks of the same bay can be split to more than one QC, which may produce in better

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

distribution of the workload that in turn may impact the total processing time of a vessel. The objective in the QCSP is to find the distribution of QCs that fulfills all transshipments of containers while satisfying the different technical requirements. In most cases, the performance of QC schedules is measured by the time required to handle a vessel. By minimizing this time, the vessel would have an earlier departure, which allows additional vessels to berth, and as a result, increases the terminal's throughput. This performance criterion of minimizing the completion time of the last task is referred to as the minimization of the makespan. Other measures such as minimizing the idle time of QCs or minimizing the moves of QCs between vessel segments can be considered. Nonetheless, the objective of such measures lies with the departure time of a vessel.

The aim of this research is to investigate the QCSP by considering different practical aspects with the objective of minimizing the makespan. The main contribution of this paper is to develop new methods based on a new representation of the problem that allows for obtaining exact solutions for large-sized problems in a reasonable amount of time. Our findings are confirmed by a computational study, in which our proposed methods are compared to current results in the literature.

This paper is organized as follows: Section 2 presents a literature review for research related to the QCSP. The problem characteristics are described in Section 3. The proposed methods to solve the problem under consideration are discussed in Section 4. Section 5 presents the results of our experiments on benchmarking instances. The conclusion and directions for future research are discussed in Section 6.

2. Literature review

The seaside operations optimization has been an active field of research that aims at improving operations efficiency especially with the impressive growth of maritime trade activities. For a comprehensive review of the different models and approaches used to address quay problems, the reader is referred to the surveys of Bierwirth and Meisel (2010, 2015). Although some research addressed the integrated operations for seaside problems (e.g. Diabat and Theodorou (2014); Fu et al. (2014); Msakni et al. (2016); Agra and Oliveira (2018)) or landside problems (e.g. Saini et al. (2017)), we focus in this review on the work exclusively related to the QCSP.

Kim and Park (2004) initiated a stream of works for the single ship QCSP where a task is defined as a *container group*. The set of containers that belong to the same group has to be processed simultaneously and without preemption. In addition, there is a precedence relation between the container groups as well as a non-simultaneity relation, in which it is not possible to process two container groups located at adjacent bays at the same time. A pre-assigned number of QCs is assumed to be available to perform the loading and unloading of tasks when they become ready. The problem with these specifications is referred to in the literature as the *group container* QCSP.

ACCEPTED MANUSCRIPT

Kim and Park (2004) formulated this problem using a mixed-integer programming (MIP) model that considers non-crossing, QC traveling time, and safety margin constraints. Later, this model was subject to many improvements in the literature, e.g. Moccia et al. (2005), Bierwirth and Meisel (2009). Nonetheless, all proposed models fail to solve medium- and large-sized instances and consequently heuristic solution methods have been developed. Sammarra et al. (2007) proposed a two-stage problem decomposition that was solved using a Tabu Search algorithm enhanced by a local search procedure. Chung and Choy (2012); Kaveshgar et al. (2012); Chung and Chan (2013) implemented different versions of a Genetic Algorithm for the problem. Nguyen et al. (2013) developed an evolutionary heuristic based on Genetic Algorithm and Genetic Programming. Izquierdo et al. (2011) proposed another variant of evolutionary heuristic, called Estimation of Distribution Algorithms, which generates new solutions using a probabilistic model that is constantly updated with the statistical information of the individual solutions. Due to the variety of the proposed models, Meisel and Bierwirth (2011) attempted to propose a unified platform for the evaluation of mathematical models and solution approaches.

The group container QCSP as defined by Kim and Park (2004) was subsequently extended in the literature to consider other characteristics. Chen et al. (2011) addressed the indented berth QCSP, in which QCs can simultaneously operate on both sides of the vessel in order to reduce the handling time. In other works, QCs were subject to time windows availability, initial bay position, and unidirectional movement. Many techniques have been proposed to solve this problem variant: Meisel (2011) proposed a tree-search solution method, Monaco and Sammarra (2011) implemented a Tabu Search algorithm, Guo et al. (2013) presented a modified generalized extremal optimization (MGEO), Chen et al. (2014) developed a compact mathematical model, and Legato et al. (2012) proposed a solution method based on Timed Petri Net model. In Legato and Trunfio (2013), QCs were only restricted to unidirectional schedules. The problem was solved using a branch-and-bound algorithm. Unsal and Oguz (2013) addressed the problem with time windows and bidirectional schedules and solve it with a constraint programming model. Wu and Ma (2017) extended the QCSP by including draft and trim constraints to ensure the safety of ship hull during the handling operations. The proposed problem was solved with a branch-and-bound algorithm and a hybrid genetic algorithm. In a recent study, Chen and Bierlaire (2017) examined the unidirectional QCSP problem and developed different versions of a makespan-constrained model to consider the vessel instability and solution robustness.

Some other articles addressed different variants of the QCSP. Lee et al. (2008a) were interested in the QCSP with handling priority of tasks that are defined as bays that have different priorities. The assignment of QCs has to ensure non-crossing and safety margin constraints. The authors formulated the problem using a mixed-inteer programming (MIP) and proposed a genetic algorithm to solve it. In Zhang et al. (2008), the on-line bay QCSP was studied where the objective was to minimize the makespan subject to non-crossing constraints. The authors proposed approximate algorithms for both the over-list

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

version, in which each task is assigned to QC in the one-by-one mode without considering the remaining tasks; and the over-time version, in which the tasks are assumed to arrive over time. Some other works addressed the bay QCSP with only non-crossing constraints (e.g. Wang and Kim (2009), Lee and Chen (2010) and Liu et al. (2014)). Recently, Zhang et al. (2017) developed approximate algorithms for multi QCs with a uniform work rate, and two QCs with different work rates. In Hakam et al. (2012), the safety margin was included and the problem was solved using a genetic algorithm. Conversely, Wang et al. (2012) included the traveling time of QCs and proposed a Particle Swarm Optimization. Lee et al. (2011) developed a solution method for the bay QCSP in which the QCs are placed in an indented berth and subject to non-crossing and safety constraints. Lu et al. (2012) included the concept of contiguous bay operations, known as area bay problem, and developed a polynomial time heuristic. Wang et al. (2013) took into consideration the vessel stability while building a solution for the bay QCSP.

A new stream of research in the literature that defines a task as a single container has recently emerged. Here, the granularity of one operation is smaller compared to the container group. This allows assigning containers in the same bay to more than one QC, which should result in more balanced workload and less idle time of QCs. Choo et al. (2010) tackled the single ship QCSP, where the QCs were subject to safety and non-crossing constraints, and the traveling time of QCs between bays was assumed negligible. The authors developed a compact mathematical formulation and a simple construction heuristic. In addition, they proposed a column generation algorithm in which the master problem is formulated as a QC-to-bay assignment and the pricing problem is modeled as a shortest path problem. The column generation was integrated into a branch-and-price algorithm that has provided exact solutions for large scale problems in a short CPU time. At the second stage, the authors addressed the multi-ship case that considers the yard storage congestion and solved it by a Lagrangian relaxation technique that decomposes the problem into single ship subproblems and relaxes the yard congestion constraints. Al-Dhaheri et al. (2016a) considered the vessel stability constraints during loading/unloading containers on/off the vessel, along with nonzero traveling time of QCs between bays. The authors proposed an MIP formulation that also integrates safety margin and non-crossing constraints. For medium and large scale problems, a genetic algorithm was proposed in which chromosomes were represented using QC-to-bay assignments. The fitness evaluation of the chromosome was performed by a graph transformation that builds unidirectional schedules only. Al-Dhaheri and Diabat (2015) addressed the problem of the QCSP with bidirectional movement and separate work rate for the QCs; however, their work neglected the safety margin constraints and the traveling time between bays. They proposed an MIP model to minimize the unbalanced workload between all bays over time. Al-Dhaheri and Diabat (2017) presented an MIP formulation for the QCSP subject to safety margin and non-crossing constraints with the objective of minimizing the makespan. The proposed formulation is based on an active bay per time segment representation, which allows for solving large problems using a standard

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

commercial solver. However, the performance of the model deteriorates when the ship stability constraints are considered, and thus a heuristic was developed. Moreover, the authors addressed the multi-ship problem and solved it using a Lagrangian relaxation-based method.

As discussed above, most works in the literature proposed near-optimal solutions. Few papers developed exact approaches to solve this problem, e.g. a branch-and-cut presented in Moccia et al. (2005), a branch-and price proposed by Choo et al. (2010) and a constraint programming model of Unsal and Oguz (2013). Generally, MIP models are proposed to provide a mathematical formulation of the addressed problem and solve small (or medium) problems. Typically, such models are based on assignment decision variables. For example, the models for complete bays QCSP are based on binary variables to assign QCs to bays, e.g Lee and Chen (2010). For the case of group container QCSP, the decision variables are defined to take one if a QC is assigned to a group of containers and, in addition, another index is added to establish the precedence relation between two groups of containers, e.g. Kim and Park (2004) and Bierwirth and Meisel (2009). Finally in the case of single container QCSP, the MIP models use decision variables based on QC-to-bay assignment over time to locate the position of each QC at each segment of the planning horizon, e.g. Choo et al. (2010), Al-Dhaheri and Diabat (2015), and Al-Dhaheri et al. (2016b).

The present study follows the latter stream, in which a task is defined for a single container considering non-crossing, safety margin, traveling time, and initial position for QC. The objective is to find the QCs' schedules that minimize the completion time of all vessel operations. Unlike similar works in the literature where the problem is formulated using QC-to-bay assignment, we propose a new mathematical model based on a graph-representation of the problem. Moreover, we use the developed model to derive another exact method that has the advantage of requiring shorter CPU time.

3. Problem characteristics

The start of the planning horizon for a vessel is when it is berthed. It is then divided into B contiguous segments (bays), each of which contains a stack of containers. Bays are indexed in ascending order from left (bow of the vessel) to right (stern of the vessel) according to their relative position on the vessel. The workload specifies the tasks for each bay defined by the number of containers to be loaded/unloaded. There are K QCs assigned to a vessel and are ready to operate at the beginning of the time horizon. The QCs are assumed to be identical, meaning that the required time to load or unload one container is the same for all QCs. Therefore, the planning horizon can be discretized into units where a unit corresponds to the time required to handle one container, including the time to load/unload the container onto/ from other resources, such as yard trucks. The QCs are indexed in the same order and direction as the bay index, and they move on a rail track from the left (bow) to the right (stern) of the vessel without allowing QCs to cross each other. In other words, the ordered

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

index of QCs has to be kept during the whole time horizon. In addition, for safety reasons, two adjacent QCs are not allowed to work on two adjacent bays at the same time, meaning that a safety margin (clearance) distance has always to be ensured during the planning horizon. This clearance can be measured by the number of bays, r, that must separate two adjacent QCs. Each QC has an initial known position and can work only on one bay at a given time. To move from one bay to an adjacent bay, a QC requires one unit of time. Therefore, the total traveling time between bays b_1 and b_2 is $|b_2 - b_1|$ and the objective is to minimize the total time necessary to complete processing all containers, or what is known as the makespan. Lee et al. (2008b) showed that the restricted version of the QCSP with non-crossing constraint is NP-complete, which means the more general problem addressed in this paper is also NP-complete.

It should be noticed that safety margin constraints along with non-crossing constraints restrict the assignment of QCs to bays. Indeed, QC k can only reach bay b if (1) $b \geq (k-1)(r+1)+1$ and (2) $b \leq B-(K-k)(r+1)$. Therefore, the set $\mathcal{B}_k$ of bays reachable by QC k can be defined as follows

$$\mathcal{B}_k = \{b : b \in \mathcal{B}, (k-1)(r+1)+1 \leq b \leq B-(K-k)(r+1)\}. \tag{1}$$

Consequently, a bay can be processed by at least one QC when $B \geq K(r+1)$; otherwise the problem becomes infeasible.

In our work, we consider both bidirectional and unidirectional schedules for the QCSP. Even though the bidirectional schedules of QCs are the most intuitive, allowing each QC to go from one bay to another regardless of the movement of the other QCs, some ports prefer to limit the movement of QCs to a unidirectional mode (Legato et al. (2012)). For the latter mode, QCs assigned to the same vessel and repositioned from their initial position follow the same direction from either bow to stern or stern to bow during the whole time horizon. Clearly, restricting the movement to unidirectional movement only reduces the search space and solutions can be obtained in less computational time than with the bidirectional movement, most probably at the expense of the solution quality. Bierwirth and Meisel (2009) highlighted that the optimal makespan of unidirectional schedules is greater or equal to that of bidirectional schedules.

The different input parameters and sets defined for the problem are summarized in Table 1.

Example 1: Table 2 gives the workload for a vessel with 10 bays. Each column indicates the bay and the number of containers to load or unload. We assume that 3 QCs are ready to operate at the beginning of the planning time horizon. The initial positions of QC1, QC2, and QC3 are 1, 7, and 10, respectively. The safety margin for two adjacent QCs is set to one bay, i.e. $r = 1$. All QCs are identical and have the same work rate of one container per one time unit. The traveling time of QC between two adjacent bays is equal to one time unit.

Figure 1 presents the optimal solution for Example 1. The work of each QC is represented with a different color, whereas the movement of QC between bays is represented by arrows. As illustrated, the movement of QCs is bidirectional

ACCEPTED MANUSCRIPT

Table 1: Parameters and sets of the QCSP.

K	total number of QCs assigned to vessel,
$\mathcal{K}$	set of QCs, indexed by k,
B	total number of bays,
$\mathcal{B}$	set of bays, indexed by b,
$\mathcal{B}_k$	set of bays reachable by QC k,
T	length of the planning horizon,
$\mathcal{T}$	set of the discretized planning horizon, indexed by t, $\mathcal{T} = \{1, \ldots, T\}$,
w_b	workload on bay b in terms of number of containers to be handled,
r	clearance distance between two adjacent QCs in terms of number of bays, $r \geq 1$,
I_k	initial bay position of QC k.

Table 2: Workload profile for Example 1.

Bay	1	2	3	4	5	6	7	8	9	10
Workload	10	6	0	4	2	1	10	4	1	1

and the corresponding optimal makespan is $C_{max} = 17$.

In Figure 2, we illustrate the optimal solution for the same example when the QCs are restricted to unidirectional schedules from right-to-left, for which the optimal solution is $C_{max} = 18$. When left-to-right movement is considered, the best C_{max} is equal to 19. This example shows that the bidirectional schedules dominate unidirectional ones. However, the latter configuration has the advantage of reducing the search space while generating high-solution quality.

4. Exact methods for the QCSP

In this section, we present two exact methods that are based on a graph representation of the problem. In Section 4.1, we give the different steps to construct the corresponding graph. In Sections 4.2 and 4.3, we present a mathematical model and an exact approach for the problem based on a binary search algorithm, respectively. In Section 4.4, we describe a construction heuristic that is used to reduce the graph size and improve the performance of the exact methods.

4.1. Graph representation

The QCSP can be represented by a directed graph $G = (N, A)$, where N and A are the set of nodes and arcs, respectively. The schedule of a QC is obtained through the flow of arcs from the source node s to the sink node u. Each QC k, $k \in \mathcal{K}$, is represented by a node s_k having only one incoming arc from the source

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

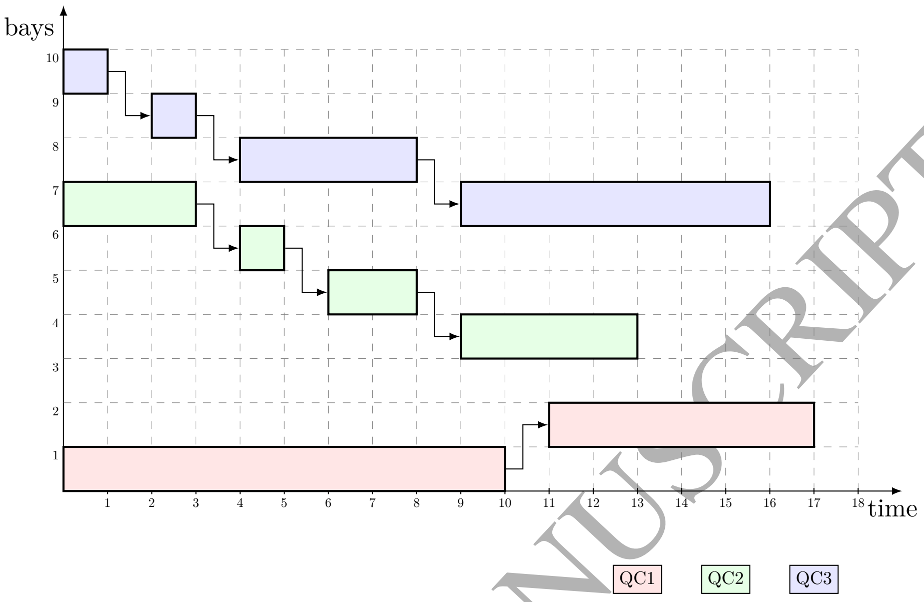


Figure 1: Optimal solution of Example 1 for bidirectional movement.

node s. The workload on a bay b at time t is represented by a node $j = \langle b, t\rangle$, where $b \in \mathcal{B}$ and $t \in \mathcal{T}$. Therefore, the total number of workload nodes is equal to the number of bays times the length of the planning horizon, $B \times T$. We denote by N_{QC} the subset of nodes that represents QCs and N_B the subset of nodes that represents the workload. Therefore, $N = \{s\} \cup N_{QC} \cup N_B \cup \{u\}$.

The set of arcs A represents the possible movement of QCs according to a bidirectional QC operation and the result of the selected arcs defines the schedule of all QCs. As mentioned above, a schedule of QC k is a path starting from the arc (s, s_k) and ending at the sink node u. The visited nodes represent the occupied bays by QC k throughout the time horizon. We now give a description of how the set, A, is built:

- An arc $(s, s_k) \in A$ represents a link between the source node s and a QC node $s_k \in N_{QC}$.
- An arc $(s_k, j) \in A$ links the QC node $s_k \in N_{QC}$ to the workload node $j = \langle b, t\rangle \in N_B$ to represent the first possible operation on b of QC k. Hence, $(s_k, j) \in A$ iff $t = |b - I_k| + 1$ and $b \in \mathcal{B}_k$.
- An arc $(j, j') \in A$ links two workload nodes $j = \langle b, t\rangle \in N_B$ and $j' = \langle b', t + 1\rangle \in N_B$, where $1 < b < B$ and $1 \leq t < T$. It represents the potentially occupied bay b' at time period $t + 1$. Thus, there are three scenarios for b': (i) $b' = b$, the same bay is occupied during t and $t + 1$,

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

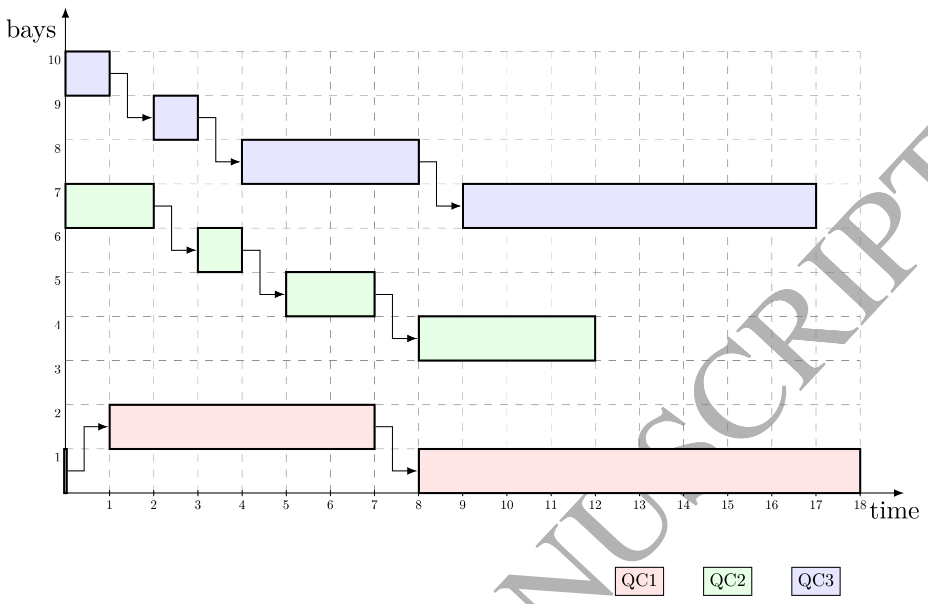


Figure 2: Optimal solution of Example 1 for unidirectional movement.

meaning that a container of b is being handled at $t+1$; (ii) $b' = b+1$, which models a QC movement from b to the adjacent bay on the right-hand side during t and $t+1$; and (iii) $b' = b-1$, which represents a QC movement from b to the adjacent bay on the left-hand side during t and $t+1$.

The first bay has one adjacent bay only. Therefore, for $j_1 = \langle 1, t\rangle \in N_B$, there are exactly two arcs $(j_1, j_1'), (j_1, j_1'') \in A$ that satisfy the aforementioned condition, where $j_1' = \langle 2, t+1\rangle$ and $j_1'' = \langle 1, t+1\rangle$.

Similarly, the last bay has only one adjacent bay. Thus, for $j_B = \langle B, t\rangle \in N_B$, the arcs (j_B, j_B') and (j_B, j_B'') belong to A, where $j_B' = \langle B, t+1\rangle$ and $j_B'' = \langle B-1, t+1\rangle$.

- An arc $(j,u) \in A$ represents a link between a workload node $j = \langle b, T\rangle \in N_B$ of the last time period T and the sink node u.

A unidirectional schedule can be derived from the graph G to obtain a reduced graph G'. The resulted graph contains only arcs that represent a unidirectional operating mode. For a left-to-right unidirectional mode, the arcs $(i,j) \in A$ between any two nodes $i = \langle b, t\rangle \in N_B$ and $j = \langle b-1, t+1\rangle \in N_B$ are removed from G. Similarly, the right-to-left unidirectional mode is obtained by removing the arcs $(i,j) \in A$ such that $i = \langle b, t\rangle \in N_B$ and $j = \langle b+1, t+1\rangle \in N_B$.

For each arc $(i,j) \in A$, we associate a binary cost $c_{i,j}$ that is equal to 1 if

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

(i,j) represents an effective operation of loading or unloading a container, and 0 otherwise. Formally, the weights of the arcs are computed as follows:

- $c_{i,j} = 1$, if
  - $i = s_k$ ($s_k \in N_{QC}$) which is the first operation to be performed by QC k,
  - $i = \langle b, t\rangle \in N_B$ and $j = \langle b, t+1\rangle \in N_B$ which establishes that the QC stays at the same bay b and one QC operation can be accomplished, $1 \le t < T$,
- $c_{i,j} = 0$, otherwise, which means that the corresponding QC is traveling from node i to node j.

Example 2: To illustrate how the graph G is built, we use a small example of QCSP with 2 QCs and 5 bays. The workload profile for each bay is as given in Table 3. The initial position of QC1 and QC2 are 1 and 5, respectively. At any time, a safety margin equivalent to one bay length has to be ensured between two adjacent QCs. The planning horizon can be set to 8 (or any valid upper bound). The resulting graph G of this example is shown in Figure 3 where only non-zero arc weights are labeled. In this example, $N_{QC} = \{s_1, s_2\}$ and N_B is formed with all the nodes of G except $\{s, s_1, s_2, u\}$. Here, QC1 can only reach Bays 1 to 3, i.e. $\mathcal{B}_1 = \{1, 2, 3\}$. The first operation of QC1 on a bay of $\mathcal{B}_1$ is only possible at $t = |b - I_1| + 1$. Therefore, s_1 has outgoing arcs to $\langle 1,1\rangle$, $\langle 2,2\rangle$, and $\langle 3,3\rangle$. The arcs between the nodes of N_B model the occupied bay at the next time period. For example, the arc between $\langle 4,2\rangle$ and $\langle 4,3\rangle$ means that the QC will stay at the same Bay 4 between periods 2 and 3 and one container can be handled during this period, $c_{\langle 4,2\rangle,\langle 4,3\rangle} = 1$; whereas arcs between $\langle 4,2\rangle$ and $\langle 3,3\rangle$, and $\langle 4,2\rangle$ and $\langle 5,3\rangle$ represent the movement of the QC to Bays 3 and 5 at $t = 3$, respectively.

Table 3: Workload profile for Example 2.

Bay	1	2	3	4	5
Workload	3	1	2	4	2

In addition, we present in Figure 4 the reduced graph G' when the QCs are restricted to left-to-right movement.

Figure 5 represents an optimal solution for Example 2 with an optimal makespan C_{max} of 8. The optimal solution is obtained with two different paths from nodes s to u. Each path corresponds to a schedule of a QC: the path starting with arc (s, s_1) represents the schedule of QC 1, whereas the path starting with arc (s, s_2) represents the schedule of QC 2. Below each visited node, the current cumulative workload is reported. For example, the workload of node $\langle 1,3\rangle$ related to Bay 1 at $t = 3$ is 3 unloaded containers. At the next time period $t = 4$, QC 1 is moved to the adjacent Bay 2, which is represented by

ACCEPTED MANUSCRIPT

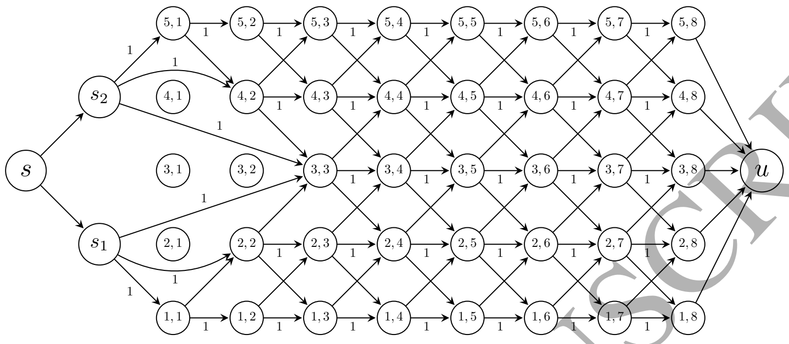


Figure 3: The obtained graph G of Example 2 for a bidirectional movement.

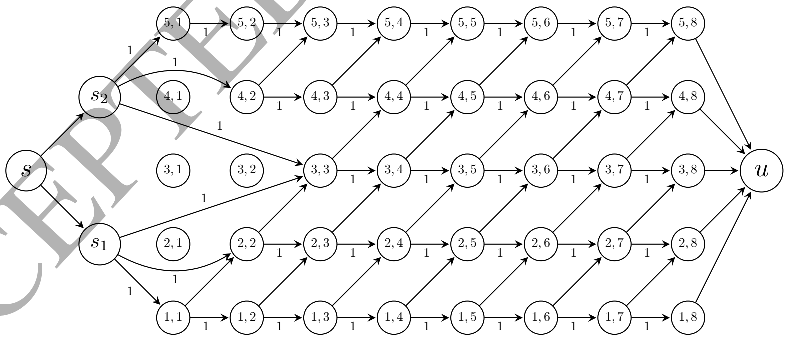


Figure 4: The obtained graph G' of Example 2 for a left-to-right movement.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

the arc going from node $\langle 1,3\rangle$ to $\langle 2,4\rangle$. The workload of the latter node is 0 because moving the QC from one bay to another bay requires one time unit and, therefore, no work can be performed on Bay 2 during the time period $t = 4$.

It is of note that the planning horizon, T, in Example 2 is set equal to the optimal makespan. However, T can be greater than the optimal makespan and, consequently, the graph G will include additional nodes $j = \langle b,t\rangle \in N_B$, such that $C_{max} < t \leq T$. This means that these nodes will be part of the optimal schedule as they will be used for each path to reach the sink node u. In such cases, the optimal makespan is obtained by finding the minimum value of t ($t \in \mathcal{T}$), for which the workload of all bays is satisfied.

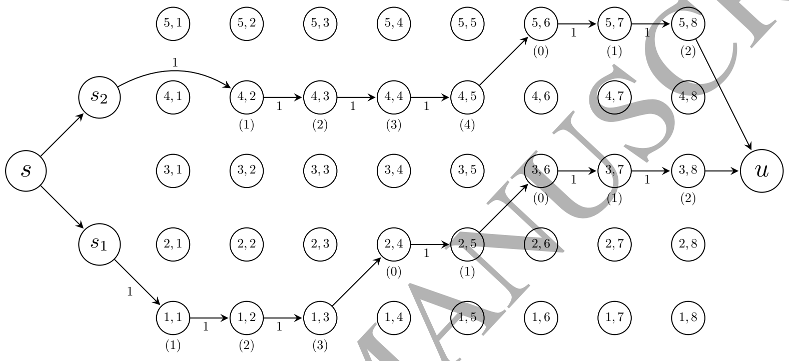


Figure 5: Optimal solution of Example 2

The graph representation can be generalized to consider other problem characteristics as detailed in the following remarks.

Remark 1. Suppose that a QC k can operate after its earliest available time (or ready time) e_k. Arcs A of G are updated so that a QC node $s_k \in N_{QC}$ has an arc with a workload node $j = \langle b,t\rangle \in N_B$ iff $t = e_k + |b - I_k| + 1$. It is noteworthy that in this case the set of reachable bays $\mathcal{B}_k$ should be redefined to consider the number of available QCs over the planning horizon.

Remark 2. In case the initial position of a QC is outside the boundaries of the vessel, i.e. $I_k < 1$ or $I_k > K$, the condition related to creating arcs linking QC nodes to workload nodes, as stated in the second bullet of building the set A in Section 4.1, remains valid.

Remark 3. We can generalize graph G to address the case where the QC traveling time is not equal to one time unit. Let $\bar{t}$ denote the traveling time of a QC to move between two adjacent bays. According to the value of $\bar{t}$, two cases can be identified. The first case is when $\bar{t}$ is an integer value such that $\bar{t} \geq 2$; i.e. the time required to move a QC between two bays is greater than handling one container, then, G is updated as follows. Any arc of A linking a QC node to a workload node $j = \langle b,t\rangle \in N_B$ has to ensure that $t = |b - I_k|\,\bar{t} + 1$. In addition, a QC movement between two adjacent bays is represented with an arc

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

that links two workload nodes $j = \langle b,t \rangle \in N_B$ and $j' = \langle b',t' \rangle \in N_B$ such that $|b - b'| = 1$ and $|t - t'| = \bar{t}$. The second case is related to QC traveling time less than handling one container, i.e. $\bar{t} < 1$. Here, we assume that $\bar{t}$ can be written as $\bar{t} = 1/\omega$, where ω is a positive integer. To solve this problem, we can use the same graph structure presented above and update only the non-null arc weights $c_{i,j}$ to be equal to ω. In addition, the load profile w_b of each bay $b \in \mathcal{B}$ is multiplied by ω. After solving this graph (as explained in Section 4.2), the obtained value is divided by ω to derive the makespan.

4.2. Mathematical formulation

The mixed integer programming (MIP) model requires the definition of the following parameters and decision variables:

Parameters:

δ_j^-: subset of A that contains incoming arcs to node $j \in N$,

δ_j^+: subset of A that contains outgoing arcs from node $j \in N$,

α_j: a function that returns the bay number of the node j, i.e. for $j = \langle b,t \rangle \in N_B$, $\alpha_j = b$.

$\beta(b)$: subset of workload nodes of N_B related to bay b. Hence, $\beta(b) = \{j = \langle b,t \rangle \in N_B\}$,

A_b: subset of arcs in A that involve all incoming arcs of nodes that represent the bay b. Hence, $A_b = \{(i,j) \in A : j \in \beta(b)\}$.

$\widehat{A}_b$: subset of arcs in A_b that represent possible effective operations on the bay b. This means that the arcs that represent movement from adjacent bays to b are excluded from this subset. That is, if the arc $(i,j) \in A_b$ and $c_{i,j} = 1$, then $(i,j) \in \widehat{A}_b$.

A_s^j: subset of arcs in A used to represent the safety margin constraint. Given node $j = \langle b,t \rangle \in N_B$, this subset includes all incoming arcs of nodes that *(i)* are positioned from 1 to r bays away from the right-hand side of bay b, and *(ii)* share the same time period t as j. Thus,

$$A_s^j = \{(i,k) \in A : j = \langle b,t \rangle, k = \langle b',t \rangle, b' \in [b+1, \min(b+r, B)]\}.$$

Defining this set will help write safety margin constraints; when bay b of node $j = \langle b,t \rangle$ is selected, there should be no incoming arcs for nodes (bays) that must be clear on the right-hand side of b; thus, no arcs of A_s^j will be selected. Hence, defining another set to guarantee the safety margin at the left-hand side of a node becomes redundant because if no r bays away are selected on the right-hand side of the current selected bay, it means that the adjacent selected bay has at least r free bays away on the left-hand side.

For the example in Figure 3, we have:

ACCEPTED MANUSCRIPT

- $\beta(3) = \{\langle 3,3\rangle, \langle 3,4\rangle, \langle 3,5\rangle, \langle 3,6\rangle, \langle 3,7\rangle, \langle 3,8\rangle\}$.
- $\widehat{A}_3 = \big\{\big(s_1, \langle 3,3\rangle\big), \big(s_2, \langle 3,3\rangle\big), \big(\langle 3,3\rangle, \langle 3,4\rangle\big), \big(\langle 3,4\rangle, \langle 3,5\rangle\big), \big(\langle 3,5\rangle, \langle 3,6\rangle\big), \big(\langle 3,6\rangle, \langle 3,7\rangle\big), \big(\langle 3,7\rangle, \langle 3,8\rangle\big)\big\}$,
- $A_s^{(3,5)} = \big\{\big(\langle 3,4\rangle, \langle 4,5\rangle\big), \big(\langle 4,4\rangle, \langle 4,5\rangle\big), \big(\langle 5,4\rangle, \langle 4,5\rangle\big)\big\}$.

Decision variables:

$x_{i,j}$: a binary variable that takes value 1 if arc $(i,j) \in A$ is selected, and 0 otherwise,

λ_t: a flag variable that takes value 0 if the workload has not been completed for all bays at time t, and 1 otherwise, $t \in \mathcal{T}$,

W_j: a positive integer variable defined on workload nodes that counts the number of operated containers for bay b at time period t, $j = \langle b,t\rangle \in N_B$. W-variables are also defined on QC nodes, where $W_{s_c} = 0$, $s_c \in N_{QC}$.

The mathematical model F1 will be

$$F1 : \text{maximize} \sum_{t \in \mathcal{T}} \lambda_t \tag{2}$$

subject to:

$$x_{s,j} = 1, \quad \forall j \in N_{QC}, \tag{3}$$

$$\sum_{(i,j)\in\delta_j^-} x_{i,j} = \sum_{(j,k)\in\delta_j^+} x_{j,k}, \quad \forall j \in N_{QC} \cup N_B, \tag{4}$$

$$\sum_{(i,j)\in\delta_j^-} x_{i,j} \leq 1 - \sum_{(a,b)\in A_s^j} x_{a,b}, \quad \forall j \in N_B, \tag{5}$$

$$\sum_{(s_k,j)\in\delta_{s_k}^+} \alpha_j x_{s_k,j} \leq \sum_{(s_{k+1},j)\in\delta_{s_{k+1}}^+} \alpha_j x_{s_{k+1},j}, \quad k \in \{1,\dots,K-1\}, \tag{6}$$

$$\sum_{(i,j)\in A_b} c_{i,j} x_{i,j} \geq w_b, \quad b \in \mathcal{B}, \tag{7}$$

$$W_j = 0, \quad \forall j \in N_{QC}, \tag{8}$$

$$W_j \leq \sum_{(i,j)\in\hat{A}_b} (W_i + c_{i,j} x_{i,j}), \quad \forall j = \langle b,t\rangle \in N_B, \tag{9}$$

$$\lambda_t \leq \frac{W_j}{w_b}, \quad \forall t \in T, j = \langle b,t\rangle \in N_B, \tag{10}$$

$$x, \lambda \quad \text{binary}, \tag{11}$$

$$W \quad \text{nonnegative integer}. \tag{12}$$

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Constraints (3) enforce all QCs to operate on the vessel. Constraints (4) guarantee the flow conservation: the QC nodes, N_{QC}, have an incoming arc from (3) that will be propagated to the bay nodes; this propagation is continued until it is stopped by the sink node u and a complete arc-flow that represents the schedule of a QC is obtained. The safety margin between two adjacent QCs is ensured by Constraints (5) where at a given time t within the planning horizon, a selected bay node (i.e. $x_{i,j} = 1$) restricts the r bay nodes on the right to be selected (i.e. all x-variables related to the incoming arcs of these adjacent r bay nodes must be equal to 0). Constraints (6) represent the non-crossing restriction for the outgoing arcs from the QC nodes. These constraints together with (4) and (5) will guarantee the non-crossing restriction for all selected nodes. Constraints (7) ensure that the obtained schedule satisfies the required workload for all bays. According to (8), the W-variables are set equal to 0. The computation of the achieved cumulative workload is performed by (9). Typically, the sum of the right-hand-side of (9) involves one arc only, which is the one representing an occupied bay during two consecutive periods of time, or an arc linking a QC node to a workload node. In some cases, it is possible to have more than one arc involved in this sum. Indeed, such arcs represent a movement of QCs from their initial positions and an arrival to a workload node at the same period of time; e.g. arcs $(s_1, \langle 3, 3\rangle)$ and $(s_2, \langle 3, 3\rangle)$ of Example 2. Constraints (10) guarantee that the workload flag variable λ will be equal to 0 when the right-hand side is strictly less than 1. The objective function (2) enforces λ to be 1 when the work on all bays is completed (the right-hand side of (10) is greater or equal to 1), and thus the makespan is evaluated. That is, if we let $z^\star$ be the optimal value of $F1$, the optimal makespan is $C_{max} = T - z^\star + 1$. Finally, the types of all variables are defined in (11) and (12).

It is worth mentioning that W-variables can be replaced with x-variables to derive a new model having fewer variables and constraints. However, we experimented with such a formulation and concluded that in general the current form of $F1$ performs better.

The computational time of the model $F1$ can be enhanced by adding valid cuts. Given a valid lower bound LB, (13) states that the last operation on the vessel could not be before $t = LB - 1$ and, therefore, λ is set to 0 during this period.

$$\lambda_t = 0, \quad t \in \{1, \dots, LB - 1\}. \tag{13}$$

For the unidirectional schedule mode, QCs are restricted to move in one direction from left-to-right or right-to-left. Thus, once a QC moves from one bay to an adjacent bay, it cannot be reassigned to the former bay. Suppose for example, we have 3 QCs assigned to Bays 2, 4, and 7 at time t, respectively. The total number of QCs from bays 1 to 7 is 3. When the movement of QCs is limited to left-to-right, at $t + 1$, it is possible to have (i) the same number of QCs on the same range of bays (i.e. from 1 to 7), which means that QC3 is still assigned to Bay 7, or (ii) the total number of QCs from bays 1 to 7 is reduced to

ACCEPTED MANUSCRIPT

2, which implies that QC3 is moved to Bay 8. This observation can be expressed as valid cuts that can be added in order to speed up the computational time required to find unidirectional schedules. Let $t_0 = \max_{k \in \mathcal{K}} I_k$ and $\tilde{A}_b^t$ be the set of incoming arcs of a workload node that represents a bay b at time t, $\tilde{A}_b^t = \{(i,j) \in A : j = \langle b, t \rangle \in N_B\}$. (14) is a valid cut for the left-to-right mode and (15) is a valid cut for the right-to-left mode.

$$\sum_{b'=1}^{b} x_{i,j \in \tilde{A}_{b'}^t} \geq \sum_{b'=1}^{b} x_{i,j \in \tilde{A}_{b'}^{t+1}}, \quad t \in \{t_0, \ldots, T-1\}, b \in \mathcal{B}, \tag{14}$$

$$\sum_{b'=1}^{b} x_{i,j \in \tilde{A}_{b'}^t} \leq \sum_{b'=1}^{b} x_{i,j \in \tilde{A}_{b'}^{t+1}}, \quad t \in \{t_0, \ldots, T-1\}, b \in \mathcal{B}. \tag{15}$$

4.3. Binary search-based optimal algorithm

The parameter T has a great impact on the size of the graph G and consequently on the performance of the model $F1$. When T is set to a large upper bound, solving $F1$ would take more time than with a tight value of T. Conversely, if T is assigned to a lower bound of the optimal solution, $F1$ becomes infeasible. Given this observation, finding the optimal makespan is equivalent to finding the smallest size of G constructed with $T^\star$ for which $F1$ is feasible. Indeed, for a graph G obtained with a value less or equal to $(T^\star - 1)$, the problem becomes infeasible. Therefore, we can adapt the well-known Binary Search Algorithm (BSA) to solve the problem under consideration. The objective is to reduce the number of checking (iterations) the feasibility of $F1$ for different values of T. Initially, the binary search starts by computing lower and upper bounds. Then, T is set to the middle value of the interval delimited by the computed lower and upper bounds. If the problem is feasible, the search continues in the left half of the interval; otherwise, the search continues in the right half of the interval. The binary search stops when the interval becomes empty and the last value for which the problem is feasible is returned as the optimal value.

The main step of the binary search is to check the feasibility of $F1$ to solve G with a size T. Therefore, only constraints related to flow conservations, non-crossing, and workload are mandatory, whereas the objective function and the remaining constraints of $F1$ are not necessary. The obtained mathematical model $F2$ is as follows.

$$F2 : find(x) \tag{16}$$

subject to:

$$(3), (4), (5), (6), (7)$$

$$x \quad \text{binary} \tag{17}$$

The advantage of the binary search is to reduce the number of iterations to $O(\log n)$ and the quality of bounds used as starting points can reduce the

ACCEPTED MANUSCRIPT

computational time. In this regard, we used the lower bound described in Al-Dhaheri et al. (2016a), which is adapted from the work of Guan et al. (2013). To obtain an upper bound, we developed a construction heuristic for the problem that will be discussed in Section 4.4.

The iterative procedure of BSA is described in Algorithm 1.

Algorithm 1 Binary search-based algorithm (BSA)

```
1: Compute a lower bound $LB$
2: Compute an upper bound $UB$
3: $UB^\star \leftarrow UB$
4: while $LB \leq UB$ do
5:     $T \leftarrow \lfloor \frac{LB+UB}{2} \rfloor$
6:     Construct the graph $G$ with $T$ and solve $F2$
7:     if the problem is feasible then
8:         $UB^\star \leftarrow T$
9:         $UB \leftarrow T - 1$
10:    else
11:        $LB \leftarrow T + 1$
12:    end if
13: end while
14: return $UB^\star$
```

4.4. Construction heuristic

In this section, we propose a polynomial time heuristic for the problem to compute a valid upper bound that can be set as the parameter T of the graph G. A good upper bound reduces the graph size, which means that the proposed formulation can be solved in a shorter time than a graph created with a higher value of T. Furthermore, the schedules obtained by this heuristic can be used as a starting solution to enhance the performance of the solver.

The construction heuristic is presented in Algorithm 2 and the rationale is to distribute the total workload between the QCs equitably (Steps 1 to 15). This distribution considers the assignment restriction since not all bays can be reached by all QCs, and also ensures that fractional workload is shared among QCs. For example, suppose we have a total workload $\mathcal{L} = 58$ to share among 4 QCs, Steps 4 and 5 ensure that the QC-workload for QC 1, 2, 3 and 4 are $l_1 = 15$, $l_2 = 14$, $l_3 = 15$ and $l_4 = 14$, respectively. In the next stage, the construction heuristic assigns bays to QCs using a unidirectional movement (by default it is set to left-to-right). The QCs are first moved from their initial positions to the initial working bay (computed in Step 17). Then, they start working on bays according to the partial assigned work until all jobs in all bays are performed (Steps 18 to 38) and the algorithm terminates with UB as an upper bound.

This algorithm can easily be adopted to the right-to-left movement.

ACCEPTED MANUSCRIPT

Algorithm 2 Pseudocode of the Construction Heuristic

```
                                        ▷ Compute the total workload $\mathcal{L}$
$\mathcal{L} \leftarrow \sum_{b\in\mathcal{B}} w(b)$
                                        ▷ Compute the QC-workload $l_k$ for each QC $k$
$l_1 \leftarrow round(\mathcal{L}/K)$
$l_k \leftarrow round(\mathcal{L} * k/K) - \sum_{i=1}^{k-1} l_i$, for all $k = 2, \dots, K$
                        ▷ Compute the partial QC-workload $w_{k,b}$ of the QC $k$ on bay $b$
for all $k \in \mathcal{K}, b \in \mathcal{B}_k$ do
    if there is still workload on $b$ ($w_b > 0$) then
        $w_{k,b} \leftarrow \min(w_b, l_k)$
        set $l_k \leftarrow l_k - w_{k,n}$ and $w_b \leftarrow w_b - w_{k,n}$
        if QC-workload of $k$ is achieved ($l_k = 0$) then
            the assignment of $k$ is terminated and go to next QC
        end if
    end if
end for
                                        ▷ Compute the initial working bay of each QC
$s_k \leftarrow \arg\min_b(w_{k,b} \neq 0)$, for all $k \in \mathcal{K}$
                                        ▷ Create the schedule for QCs
$UB \leftarrow 0$
while there is remaining work do
    $UB \leftarrow UB + 1$
    for all $k \in \mathcal{K}$ do
        if $k$ is moving from $I_k$ to $s_k$ then
            move $k$ to the adjacent bay
        else
            let $b$ the current bay-position of $k$
            if $w_{k,b} \geq 0$ then
                Subtract one work unit from $w_{k,b}$
            else
                if the safety margin allows for it then
                    move $k$ to the adjacent bay
                else
                    keep $k$ waiting in its current bay-position
                end if
            end if
        end if
    end for
end while
return $UB$
```

5. Computational study

The aim in this section is to assess the performance of the proposed methods for different problem sizes. The different algorithms were implemented in C++

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

language using Microsoft Visual Studio C++ 2012. The Concert Technology API for C++ is used to implement and solve the mathematical models with the commercial solver IBM ILOG CPLEX 12.6. All tests were carried out on a PC running Windows 7 with an Intel i7 3.1 GHz processor and 8 GB of RAM.

The experiments are based on the problem instances of Al-Dhaheri et al. (2016a) that include small-, medium-, and large-sized instances. The small-sized instance set has thirty instances in which the number of QCs ranges between 2 and 5, and the number of bays ranges between 8 and 12. The workload on each bay is between 0 and 20. The medium-sized instances have 15 bays that are shared by 2 or 4 QCs. For each QC size, there are 5 different instances with a workload between 0 and 100 work units on each bay. Finally, the large-sized instance set includes 20 instances with 3 or 6 QCs, and 20 or 25 bays, each of which has a workload between 0 and 100. For all instances, the traveling time of QCs between two adjacent bays and the working rate of QCs are set to one time unit. The clearance distance is equivalent to one bay ($r = 1$).

All the instances are used to test the performance of the developed methods (model $F1$ and the BSA algorithm). Since these methods depend on bound values, the heuristic presented in Section 4.4 as an upper bound and the lower bound of Guan et al. (2013) are used as follows. For $F1$, the upper bound UB provided by the heuristic is used to set T (the size of the graph G) and is also used as an initial solution for CPLEX. The lower bound LB is used for the valid cuts (13). When the solution is restricted to unidirectional schedules, either (14) or (15) is added to $F1$ depending on which direction is allowed, left-to-right or right-to-left, respectively. For BSA, LB and UB are used according to the description of Algorithm 1. Moreover, preliminary experiments showed that running $F1$ and BSA using the following parameters enhanced the speed of CPLEX in finding the optimal solution: `RootAlg` (*MIP starting algorithm*) is set to `Network`; and `HeurFreq` (*MIP heuristic frequency*) is set to `20`.

The proposed methods are compared to AJD – the mathematical model of Al-Dhaheri et al. (2016a). In order to make a consistent comparison, we removed the stability constraints of AJD and set the planning horizon parameter T of AJD to UB. For all tested methods, the CPU time limit is set to 10 hours (36,000 s). When a method is able to prove the optimality of a solution, its corresponding value is highlighted in bold in the tables of results.

The first experiment aims to test the ability of model $F1$ and the algorithm BSA to solve small- and medium-sized instances for a bidirectional mode. The results are reported in Table 4 where the characteristics and the obtained LB and UB values for each instance are detailed. Since the heuristic produces a unidirectional schedule, the column (DIR) specifies the direction for the obtained solution whether it is left-to-right (LTR) schedule or right-to-left(RTL) schedule. The CPU time to obtain LB and UB are not reported because the algorithms that generate these bounds run in a polynomial time and, for all instances, the results are obtained in a negligible CPU time (less than 0.01 s). The column (GAP) of Table 4 gives the gap (in percentage (%)) of the obtained UB to the optimal solution or the best-known solution. This measure reveals the high solution quality of the proposed heuristic. In fact, the average gap

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

for small - and medium-sized instances is only 2.2%. The heuristic was able to provide the optimal value for 13 out of 30 small-sized instances. In 7 out of the 13 instances, LB and UB were equal, which means that the optimal solution is obtained without the use of any advanced exact method.

It can be seen from Table 4 that $F1$ and BSA were able to solve most small instances in a few seconds and the best performance is obtained by BSA. In fact, the average CPU time for BSA to solve small instances is 12.6 s; whereas it is 35.9 s for $F1$. However, for medium-sized instances, $F1$ provides the best results in terms of both number of solved instances and average CPU time. $F1$ is able to solve 6 out of 10 instances compared to 4 out of 10 instances for BSA. For the remaining instances where the optimality is not proven, all developed methods generate high-quality solutions where the average gap between the provided solution and the lower bound for the unsolved instances is less than 0.6%. Furthermore, the experiment shows the superiority of the developed methods over the model of Al-Dhaheri et al. (2016a).

The aim in the second experiment is to test the developed exact methods for unidirectional schedules. Table 5 reports the results of this experiment for small- and medium-sized instances. The column OPT_DIR refers to the direction for which the best solution is obtained for each instance. Cases in which both directions yield the same optimal result are indicated by LTR-RTL. The results reveal the clear dominance of the proposed exact methods compared to AJD where all methods were able to solve all small-sized instances in less than 1 s; whereas AJD model always required more CPU time. Also, the AJD model could not provide any feasible solution for the medium-sized instances within the time limit, while $F1$ and BSA solved all these instances in few seconds. Overall, the best performance is shown by BSA, requiring less CPU time for medium-sized instances than $F1$. For example, the average CPU time of BSA to solve the instances with 15 bays and 4 QCs is 40% of the average CPU time of $F1$. Furthermore, the obtained results show the impact of the graph size G on the general performance of $F1$ and BSA, where in these methods, the CPU time required to solve unidirectional schedules is considerably lower than that required to solve the bidirectional schedules of the same instances (see Tables 4 and 5). Since, for a given instance, the graph G' (for the unidirectional movement) includes fewer arcs than the corresponding graph G (for the bidirectional movement), the performance of $F1$ and BSA have been drastically improved. Finally, from Tables 4 and 5, it can be observed that the optimal solution for the unidirectional schedule is also optimal for the bidirectional schedule for most instances. For 37 out of 40 instances where the optimal bidirectional solution is known, only two instances (20 and 30) have an optimal unidirectional value greater than the optimal bidirectional value. However, even for these two instances, the difference between the optimal unidirectional and bidirectional solutions is only 1 time unit, meaning that restricting the QCs to the unidirectional movement has the advantage of reducing the complexity of the problem, without compromising on quality.

In another experiment, we test the ability of the developed methods to solve large-sized instances. It is noteworthy that most of the literature on QCSP

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Table 4: The obtained results for bidirectional schedules for small-and medium-sized instances.

#.	K	B	$\sum w_b$	LB	UB			AJD model		$F1$		BSA	
					Sol.	DIR	GAP	Sol.	CPU	Sol.	CPU	Sol.	CPU
1	2	8	63	36	36	LTR	0.00	**36**	-	**36**	-	**36**	-
2			61	34	35	LTR	2.94	**34**	32.74	**34**	0.44	**34**	0.38
3			67	37	38	LTR	2.70	**37**	50.83	**37**	0.37	**37**	0.57
4			92	50	50	LTR	0.00	**50**	-	**50**	-	**50**	-
5			101	55	55	LTR	0.00	**55**	-	**55**	-	**55**	-
								avg.	16.71	**avg.**	0.16	**avg.**	0.19
6	3	8	63	23	26	RTL	0.00	**26**	0.59	**26**	0.29	**26**	0.65
7			61	23	31	RTL	0.00	**31**	1.11	**31**	0.26	**31**	0.75
8			67	33	33	LTR	0.00	**33**	-	**33**	-	**33**	-
9			92	33	34	LTR	0.00	**33**	7.80	**34**	1.00	**34**	1.25
10			101	37	38	LTR	2.70	**37**	153.61	**37**	0.60	**37**	2.70
								avg.	32.62	**avg.**	0.43	**avg.**	1.07
11	3	10	93	34	35	LTR	0.00	**35**	36,000	**35**	2.15	**35**	7.08
12			76	29	35	LTR	3.03	**32**	812.97	**32**	2.64	**32**	1.07
13			83	33	34	LTR	3.03	**33**	8.00	**33**	0.43	**33**	0.54
14			110	41	42	LTR	2.44	**41**	36,000	**41**	7.38	**41**	6.54
15			124	45	47	LTR	2.17	**47**	36,000	**46**	427.51	**46**	117.39
								avg.	21,764.19	**avg.**	88.02	**avg.**	26.53
16	4	10	93	26	31	RTL	0.00	**31**	3.10	**31**	0.37	**31**	0.66
17			76	22	32	RTL	6.67	**30**	6.33	**30**	0.36	**30**	0.78
18			83	33	33	LTR	0.00	**33**	-	**33**	-	**33**	-
19			110	32	37	RTL	5.71	**35**	30.64	**35**	1.51	**35**	2.03
20			124	34	37	LTR	5.71	**35**	450.84	**35**	10.56	**35**	3.10
								avg.	98.18	**avg.**	2.56	**avg.**	1.31
21	4	12	109	31	35	LTR	9.38	**32**	187.44	**32**	5.86	**32**	2.72
22			90	26	33	LTR	6.45	**31**	882.50	**31**	0.44	**31**	1.58
23			100	33	33	LTR	0.00	**33**	-	**33**	-	**33**	-
24			126	35	40	LTR	11.11	37	36,000	**36**	485.09	**36**	118.86
25			160	43	45	LTR	2.27	45	36,000	**44**	128.86	**44**	105.83
								avg.	14,613.99	**avg.**	124.05	**avg.**	45.80
26	5	12	109	25	31	LTR	0.00	**31**	5.40	**31**	0.24	**31**	0.81
27			90	21	32	LTR	6.67	**30**	45.08	**30**	0.85	**30**	1.07
28			100	33	33	LTR	0.00	**33**	-	**33**	-	**33**	-
29			126	32	37	LTR	8.82	**34**	171.43	**34**	0.87	**34**	0.80
30			160	35	38	LTR	2.70	**37**	46.97	**37**	0.70	**37**	2.34
								avg.	53.78	**avg.**	0.53	**avg.**	1.01
31	2	15	719	367	368	LTR	0.27		**	**367**	279.67	**368**	36,000
32			795	406	407	LTR	0.25		**	**406**	3,098.13	**407**	36,000
33			804	410	413	LTR	0.73		**	**410**	34,514.60	**410**	34,292.50
34			612	314	315	LTR	0.32		**	**314**	10,414.00	**314**	256.36
35			715	365	366	LTR	0.27		**	**365**	1,473.15	**365**	263.99
								avg.	**	**avg.**	9,955.91	**avg.**	21,362.57
36	4	15	719	184	186	RTL	0.54		**	185	36,000	185	36,000
37			795	203	204	LTR	0.49		**	**203**	18,453.40	204	36,000
38			804	205	207	RTL	0.49		**	206	36,000	**206**	4,754.02
39			612	157	158	LTR	0.00		**	158	36,000	158	36,000
40			715	183	184	LTR	0.00		**	184	36,000	184	36,000
								avg.	**	**avg.**	32,490.68	**avg.**	29,750.80

(-) Optimal solution obtained in a negligible time because $LB = UB$.
(**) No feasible solution within the CPU time limit (36,000 s).

ACCEPTED MANUSCRIPT

Table 5: The obtained results for unidirectional schedules for small-and medium-sized instances.

#.	LB	UB	OPT_DIR	AJD model		F1		BSA	
				Sol.	CPU	Sol.	CPU	Sol.	CPU
1	36	36	LTR	**36**	-	**36**	-	**36**	-
2	34	35	LTR	**34**	3.00	**34**	0.21	**34**	0.25
3	37	38	LTR	**37**	5.54	**37**	0.20	**37**	0.25
4	50	50	LTR	**50**	-	**50**	-	**50**	-
5	55	55	LTR	**55**	-	**55**	-	**55**	-
				avg.	1.71	**avg.**	0.08	**avg.**	0.10
6	23	26	LTR-RTL	**27**	0.41	**26**	0.10	**26**	0.27
7	23	31	RTL	**31**	0.81	**31**	0.13	**31**	0.38
8	33	33	LTR	**33**	-	**33**	-	**33**	-
9	33	34	RTL	**34**	1.40	**34**	0.07	**34**	0.21
10	37	38	LTR	**37**	3.93	**37**	0.28	**37**	0.16
				avg.	1.31	**avg.**	0.11	**avg.**	0.20
11	34	35	LTR	**35**	4.81	**35**	0.15	**35**	0.44
12	33	34	RTL	**33**	19.23	**33**	0.36	**33**	0.49
13	33	34	LTR	**33**	5.48	**33**	0.18	**33**	0.28
14	41	42	RTL	**41**	22.17	**41**	0.44	**41**	0.44
15	45	47	RTL	**46**	45.10	**46**	0.41	**46**	0.74
				avg.	19.36	**avg.**	0.31	**avg.**	0.48
16	26	31	LTR-RTL	**32**	0.84	**31**	0.11	**31**	0.40
17	22	32	RTL	**30**	4.13	**30**	0.34	**30**	0.62
18	33	33	LTR	**33**	1.45	**33**	-	**33**	-
19	32	37	RTL	**35**	4.13	**35**	0.36	**35**	0.49
20	34	37	RTL	**36**	6.12	**36**	0.20	**36**	0.58
				avg.	3.34	**avg.**	0.20	**avg.**	0.42
21	31	35	LTR-RTL	**34**	40.22	**32**	0.42	**32**	0.85
22	26	33	LTR	**31**	40.76	**31**	0.41	**31**	1.05
23	33	33	LTR	**33**	-	**33**	-	**33**	-
24	35	40	RTL	**36**	245.09	**36**	0.47	**36**	0.93
25	43	45	LTR	**44**	70.18	**44**	0.44	**44**	0.83
				avg.	79.25	**avg.**	0.35	**avg.**	0.73
26	25	31	LTR	**31**	2.93	**31**	0.52	**31**	0.67
27	21	32	RTL	**30**	5.48	**30**	0.53	**30**	0.97
28	33	33	LTR	**33**	-	**33**	-	**33**	-
29	32	37	RTL	**34**	15.88	**34**	0.43	**34**	0.71
30	35	38	RTL	**38**	19.64	**38**	0.24	**38**	0.76
				avg.	8.79	**avg.**	0.34	**avg.**	0.62
31	367	368	LTR	**		**367**	37.77	**367**	30.35
32	406	407	LTR	**		**406**	84.98	**406**	48.77
33	410	413	LTR	**		**410**	237.57	**410**	87.35
34	314	315	LTR	**		**314**	151.04	**314**	17.44
35	365	366	LTR	**		**365**	220.65	**365**	21.28
				avg.	**	**avg.**	146.40	**avg.**	41.04
36	184	186	LTR-RTL	**		**185**	39.66	**185**	15.02
37	203	204	LTR	**		**203**	71.24	**203**	11.38
38	205	207	LTR-RTL	**		**206**	30.87	**206**	23.45
39	157	158	LTR	**		**158**	18.47	**158**	10.31
40	183	184	LTR	**		**184**	27.84	**184**	15.12
				avg.	**	**avg.**	37.61	**avg.**	15.05

(-) Optimal solution obtained in a negligible time because $LB = UB$.
(**) No feasible solution within the CPU time limit (36,000 s).

ACCEPTED MANUSCRIPT

with nonzero QC traveling time proposed only near-optimal solutions to solve large instances due to the complexity of the problem. However, as it can be seen in Table 6, the proposed methods are able to solve all large-sized instances, in which the total number of containers range between 1028 and 1360. To compare between the developed methods, the results show that BSA performs better than $F1$ where the average CPU time on BSA is always the lowest (except for instances 56 and 58). In addition, all instances have been solved by BSA in an average CPU time of less than 30 minutes, whereas $F1$ was not able to prove the optimality for instances 51 and 57. Furthermore, the results of this experiment confirm the high solution quality of the heuristic method which produced solutions with an average gap of only 1.32% to the optimal values, while requiring a negligible CPU time (less than 0.01s).

Table 6: The obtained results for unidirectional schedules for large-sized instances.

#.	K	B	$\sum w_b$	LB	UB		OPT_DIR	F1		BSA	
					Sol.	Gap		Sol.	CPU	Sol.	CPU
41	3	20	959	327	328	0.31	LTR	**327**	388.47	**327**	84.76
42			1051	358	358	0.00	LTR	**358**	-	**358**	-
43			1051	358	359	0.28	LTR	**358**	660.66	**358**	121.81
44			892	305	306	0.33	LTR	**305**	366.52	**305**	85.07
45			967	330	330	0.00	LTR	**330**	-	**330**	-
								avg.	283.13	**avg.**	58.33
46	6	20	959	164	167	1.21	LTR-RTL	**165**	93.98	**165**	34.12
47			1051	179	182	1.11	RTL	**180**	171.45	**180**	112.91
48			1051	179	182	1.11	LTR	**180**	96.44	**180**	45.14
49			892	153	189	8.62	RTL	**174**	116.87	**174**	48.46
50			967	165	170	1.80	LTR	**167**	283.03	**167**	72.41
								avg.	152.35	**avg.**	62.61
51	3	25	1357	462	465	0.22	LTR	464	36,000	**464**	3,101.78
52			1360	462	470	1.08	LTR	**465**	6,523.88	**465**	1,415.00
53			1314	448	455	1.11	LTR	**450**	12,778.30	**450**	2,399.84
54			1028	351	360	1.69	RTL	**354**	16,246.50	**354**	579.81
55			1231	419	427	1.18	RTL	**422**	9,278.58	**422**	1,315.93
								avg.	16,165.45	**avg.**	1,762.47
56	6	25	1357	231	233	0.87	LTR-RTL	**231**	258.22	**231**	868.15
57			1360	232	234	0.43	RTL	233	36,000	**233**	2,995.73
58			1314	225	229	1.78	LTR	**225**	614.75	**225**	1,714.46
59			1028	177	183	2.23	RTL	**179**	5,576.05	**179**	455.02
60			1231	210	213	0.95	RTL	**211**	5,189.32	**211**	413.04
								avg.	9,527.67	**avg.**	1,289.28

(-) Optimal solution obtained in a negligible time because $LB = UB$.

In addition, we observe from Tables 4, 5, and 6 that the total number of periods T, and the solution value, are related to the total number of containers and the number of QCs. Since each bay size ($b = 8, \dots, 25$) is tested with two different allocated numbers of QCs, we can deduce that increasing the QC resources allows for a better repartition of the total workload and results in a reduced makespan. Furthermore, the number of QCs impacts the complexity of the problem. Indeed, $F1$ and BSA perform generally better with a higher number of QCs.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

To push our analysis further, we conducted a computational experiment on the instances provided by Meisel and Bierwirth (2011), which were proposed as benchmark problems for the group container QCSP. Because in our case the task is defined for a single container and neither precedence nor simultaneity relations exist among tasks, we adapted the instance sets slightly to be consistent with our problem characteristics. In particular, we considered the sum of the processing times of group tasks belonging to the same bay-location to derive the workload on each bay, w_b. In group container QCSP, the safety margin constraint is implicitly defined by preventing two tasks located in adjacent bays to be operated simultaneously, which is in our case equivalent to setting the safety margin to one bay, $r = 1$. For these benchmark instances, the QC traveling time is similar to our problem characteristics. Meisel and Bierwirth (2011) proposed different instance sets A–G that differ by problem parameters. In our experiment, we only retained sets A, B, and C as they are more relevant to our case. Indeed, sets A, B, and C have different number of bays (10, 15, and 20 bays, respectively) as well as the total number of containers to be handled (1000, 3000, and 6000 containers, respectively), whereas sets D–G have only 15 bays and differ by number of QCs, safety margin distance, and frequency of precedence or non-simultaneity relations between tasks. Each set of A, B, and C has different number of group containers defined by parameter n, and each combination within each set is composed of ten different instances. Due to instance sizes, this experiment is carried out for bidirectional schedules for set A and unidirectional schedules for sets B and C. For the latter sets, all obtained schedules are related to left-to-right movement because the initial positions of QCs, as defined in this benchmark instances, are placed on the left side of the vessel starting from bay 1 and with accordance to safety margin. Table 7 reports the obtained results of this experiment. Each row gives a summary of the ten instances of each combination of n within each set. The column ($UB - Gap$) indicates the average gap in percentage of the obtained UB to the optimal solution or the best-known solution. The columns (#. solved) and (CPU) refer to the number of solved instances within the CPU time limit and the average CPU time in seconds. Note that since we have a total of 190 different instances, we thus limited the time to 2 hours only (the obtained results of each instance can be found in the Appendix section). Table 7 confirms the high-quality solution generated by the construction heuristic where the average gap is less than 1% in most cases and the maximum average gap is only 2.55% obtained for set C and $n = 80$. Interestingly, the proposed exact methods are able to prove the optimality for most instances with a dominance for BSA, which provides optimal solutions for 174 out of 190 instances in an average of 1486 seconds, whereas $F1$ solves 161 instances and requires on average 2251 seconds. Finally, all experiments confirm that developing a graph-based approach significantly increases the efficiency of the proposed exact method in finding optimal solutions for the QCSP.

Table 7: A summary of the obtained results for Meisel and Bierwirth (2011) instances.

Class	n	UB	F1		BSA	
		(Gap)	#. slvd	CPU (avg)	#. solved	CPU (avg)
A	10	0.24	10	731.6	10	909.4
	15	0.14	10	303.3	10	367.5
	20	0.24	10	612.9	10	283.2
	25	0.22	10	1201.7	8	1820.5
	30	0.24	10	1124.4	10	595.9
	35	0.24	10	989.0	9	1333.0
	40	0.28	10	1410.4	9	1381.7
B	45	0.26	10	897.0	10	273.7
	50	0.24	10	1021.7	10	355.3
	55	0.23	10	1134.9	10	462.9
	60	0.28	10	1066.2	10	387.0
	65	0.20	10	1214.8	10	470.8
	70	0.26	10	704.0	10	432.3
C	75	1.80	4	5309.2	7	3202.1
	80	2.55	7	5159.3	9	2986.8
	85	2.19	6	5179.9	7	3752.1
	90	0.40	6	4238.3	7	3280.3
	95	0.19	3	5623.6	10	2876.9
	100	1.47	5	4854.6	8	3065.1

6. Conclusion

This paper addresses an important problem that arises at the quayside of port terminals of finding optimal schedules for QCs to load and unload a berthed vessel. The problem is subject to several practical constraints such as the initial position, non-crossing, and safety margin of QCs as well as nonzero traveling time. Following the most recent stream of research in the literature, the problem defines a task to be performed by a QC as a *single container*. Contrary to *group container* or *bay* QCSP, the granularity of the *single container* QCSP is finer and offers better schedules by reducing the QC idle time; however, the problem becomes more complex and hard to solve as the resulting schedules involve more operations.

A graph-based transformation approach was used in introducing two exact methods to optimally solve the problem. The first method is based on a mixed-integer programming model that is enhanced by additional cuts while the second is based on a binary search that exploits the graph structure to find optimal solutions efficiently. The computational study showed that the two proposed methods outperform the most recent model in the literature and were able to solve large-sized instances in a reasonable CPU time and smaller instances very efficiently. In addition, a heuristic algorithm was developed to provide an initial solution for both methods. This heuristic has the advantage of being simple to implement and offers near-optimal solutions within a negligible amount of time. Furthermore, the experiments provide a proof-of-concept that restricting the movement of QC to unidirectional schedules is a good option to find high quality solutions for the QCSP.

The current work can be extended to consider container-dependent operation time and/or different QC rates. Another extension could be integrating the

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

QCSP with the QCAP to address the Quay Crane Assignment and Scheduling Problem (QCASP).

Acknowledgments

This research was made possible by NPRP Grant No. NPRP 7-796-2-297 from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

References

Agra, A. and Oliveira, M. (2018). MIP approaches for the integrated berth allocation and quay crane assignment and scheduling problem. *European Journal of Operational Research*, 264(1):138–148.

Al-Dhaheri, N. and Diabat, A. (2015). The quay crane scheduling problem. *Journal of Manufacturing Systems*, 36:87–94.

Al-Dhaheri, N. and Diabat, A. (2017). A lagrangian relaxation-based heuristic for the multi-ship quay crane scheduling problem with ship stability constraints. *Annals of Operations Research*, 248(1):1–24.

Al-Dhaheri, N., Jebali, A., and Diabat, A. (2016a). The quay crane scheduling problem with nonzero crane repositioning time and vessel stability constraints. *Computers & Industrial Engineering*, 94:230–244.

Al-Dhaheri, N., Jebali, A., and Diabat, A. (2016b). A simulation-based genetic algorithm approach for the quay crane scheduling under uncertainty. *Simulation Modelling Practice and Theory*, 66:122–138.

Bierwirth, C. and Meisel, F. (2009). A fast heuristic for quay crane scheduling with interference constraints. *Journal of Scheduling*, 12(4):345–360.

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

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

Chen, J. H. and Bierlaire, M. (2017). The study of the unidirectional quay crane scheduling problem: complexity and risk-aversion. *European Journal of Operational Research*, 260(2):613 – 624.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Chen, J. H., Lee, D.-H., and Cao, J. X. (2011). Heuristics for quay crane scheduling at indented berth. *Transportation Research Part E: Logistics and Transportation Review*, 47(6):1005–1020.

Chen, J. H., Lee, D.-H., and Goh, M. (2014). An effective mathematical formulation for the unidirectional cluster-based quay crane scheduling problem. *European Journal of Operational Research*, 232(1):198–208.

Choo, S., Klabjan, D., and Simchi-Levi, D. (2010). Multiship crane sequencing with yard congestion constraints. *Transportation Science*, 44(1):98–115.

Chung, S. and Chan, F. T. (2013). A workload balancing genetic algorithm for the quay crane scheduling problem. *International Journal of Production Research*, 51(16):4820–4834.

Chung, S. and Choy, K. (2012). A modified genetic algorithm for quay crane scheduling operations. *Expert Systems with Applications*, 39(4):4213–4221.

Diabat, A. and Theodorou, E. (2014). An integrated quay crane assignment and scheduling problem. *Computers & Industrial Engineering*, 73:115–123.

Fu, Y.-M., Diabat, A., and Tsai, I.-T. (2014). A multi-vessel quay crane assignment and scheduling problem: Formulation and heuristic solution approach. *Expert Systems with Applications*, 41(15):6959–6965.

Guan, Y., Yang, K.-H., and Zhou, Z. (2013). The crane scheduling problem: models and solution approaches. *Annals of Operations Research*, 203(1):119–139.

Guo, P., Cheng, W., and Wang, Y. (2013). A modified generalized extremal optimization algorithm for the quay crane scheduling problem with interference constraints. *Engineering Optimization*, 46(10):1411–1429.

Hakam, M. H., Solvang, W. D., and Hammervoll, T. (2012). A genetic algorithm approach for quay crane scheduling with non-interference constraints at narvik container terminal. *International Journal of Logistics Research and Applications*, 15(4):269–281.

Izquierdo, C. E., Velarde, J. L. G., Batista, B. M., and Moreno-Vega, J. M. (2011). Estimation of distribution algorithm for the quay crane scheduling problem. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2011)*, pages 183–194. Springer Science + Business Media.

Kaveshgar, N., Huynh, N., and Rahimian, S. K. (2012). An efficient genetic algorithm for solving the quay crane scheduling problem. *Expert Systems with Applications*, 39(18):13108–13117.

Kim, K. H. and Park, Y.-M. (2004). A crane scheduling method for port container terminals. *European Journal of Operational Research*, 156(3):752–768.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Lee, D.-H. and Chen, J. H. (2010). An improved approach for quay crane scheduling with non-crossing constraints. *Engineering Optimization*, 42(1):1–15.

Lee, D.-H., Chen, J. H., and Cao, J. X. (2011). Quay crane scheduling for an indented berth. *Engineering Optimization*, 43(9):985–998.

Lee, D.-H., Wang, H. Q., and Miao, L. (2008a). Quay crane scheduling with handling priority in port container terminals. *Engineering Optimization*, 40(2):179–189.

Lee, D.-H., Wang, H. Q., and Miao, L. (2008b). Quay crane scheduling with non-interference constraints in port container terminals. *Transportation Research Part E: Logistics and Transportation Review*, 44(1):124 – 135.

Legato, P. and Trunfio, R. (2013). A local branching-based algorithm for the quay crane scheduling problem under unidirectional schedules. *4OR-Q J Oper Res*, 12(2):123–156.

Legato, P., Trunfio, R., and Meisel, F. (2012). Modeling and solving rich quay crane scheduling problems. *Computers & Operations Research*, 39(9):2063–2078.

Liu, M., Zheng, F., and Li, J. (2014). Scheduling small number of quay cranes with non-interference constraint. *Optimization Letters*, 9(2):403–412.

Lu, Z., Han, X., Xi, L., and Erera, A. L. (2012). A heuristic for the quay crane scheduling problem based on contiguous bay crane operations. *Computers & Operations Research*, 39(12):2915–2928.

Meisel, F. (2011). The quay crane scheduling problem with time windows. *Naval Research Logistics (NRL)*, 58(7):619–636.

Meisel, F. and Bierwirth, C. (2011). A unified approach for the evaluation of quay crane scheduling models and algorithms. *Computers & Operations Research*, 38(3):683–693.

Moccia, L., Cordeau, J.-F., Gaudioso, M., and Laporte, G. (2005). A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. *Naval Research Logistics*, 53(1):45–59.

Monaco, M. F. and Sammarra, M. (2011). Quay crane scheduling with time windows, one-way and spatial constraints. *International Journal of Shipping and Transport Logistics*, 3(4):454.

Msakni, M. K., Al-Salem, M., Diabat, A., Rabadi, G., and Kotachi, M. (2016). An integrated quay crane assignment and scheduling problem using branch-and-price. In *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Nguyen, S., Zhang, M., Johnston, M., and Tan, K. C. (2013). Hybrid evolutionary computation methods for quay crane scheduling problems. *Computers & Operations Research*, 40(8):2083–2093.

Saini, S., Roy, D., and de Koster, R. (2017). A stochastic model for the throughput analysis of passing dual yard cranes. *Computers & Operations Research*, 87:40–51.

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

Unsal, O. and Oguz, C. (2013). Constraint programming approach to quay crane scheduling problem. *Transportation Research Part E: Logistics and Transportation Review*, 59:108–122.

Wang, J., Hu, H., and Song, Y. (2013). Optimization of quay crane scheduling constrained by stability of vessels. *Transportation Research Record: Journal of the Transportation Research Board*, 2330:47–54.

Wang, S., Zheng, J., Zheng, K., Guo, J., and Liu, X. (2012). Multi resource scheduling problem based on an improved discrete particle swarm optimization. *Physics Procedia*, 25:576–582.

Wang, Y. and Kim, K. H. (2009). A quay crane scheduling algorithm considering the workload of yard cranes in a container yard. *Journal of Intelligent Manufacturing*, 22(3):459–470.

Wu, L. and Ma, W. (2017). Quay crane scheduling with draft and trim constraints. *Transportation Research Part E: Logistics and Transportation Review*, 97:38–68.

Zhang, A., Zhang, W., Chen, Y., Chen, G., and Chen, X. (2017). Approximate the scheduling of quay cranes with non-crossing constraints. *European Journal of Operational Research*, 258(3):820–828.

Zhang, L., Khammuang, K., and Wirth, A. (2008). On-line scheduling with non-crossing constraints. *Operations Research Letters*, 36(5):579–583.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Appendix A. Results for Meisel and Bierwirth (2011) benchmark instances

Table A.8: Results for set A and $n = 10$.

#	LB	UB	F1 Sol.	F1 CPU	BSA Sol.	BSA CPU
1	507	507	**507**	-	**507**	-
2	506	507	**506**	1,111.4	**506**	527.9
3	506	507	**506**	747.8	**506**	186.7
4	505	507	**505**	3,215.1	**505**	1,626.7
5	506	508	**507**	184.4	**507**	83.1
6	505	507	**505**	525.6	**506**	5,890.0
7	506	507	**507**	116.3	**507**	185.1
8	506	507	**506**	257.7	**506**	148.6
9	505	507	**505**	505.0	**505**	194.8
10	505	507	**505**	651.6	**505**	250.9
			avg.	731.6	**avg.**	909.4

(-) Negligible CPU time because $LB = UB$.

Table A.9: Results for set A and $n = 15$.

#	LB	UB	F1 Sol.	F1 CPU	BSA Sol.	BSA CPU
1	507	507	**507**	-	**507**	-
2	505	507	**505**	649.2	**505**	1,266.1
3	507	507	**507**	-	**507**	-
4	505	506	**506**	241.8	**506**	221.6
5	505	506	**505**	175.4	**505**	188.2
6	506	507	**506**	328.4	**506**	120.5
7	506	507	**506**	670.9	**506**	1,436.1
8	506	507	**506**	371.3	**506**	120.0
9	506	507	**506**	594.7	**506**	322.5
10	507	507	**507**	-	**507**	-
			avg.	303.3	**avg.**	367.5

(-) Negligible CPU time because $LB = UB$.

Table A.10: Results for set A and $n = 20$.

#	LB	UB	F1 Sol.	F1 CPU	BSA Sol.	BSA CPU
1	506	507	**506**	231.1	**506**	145.7
2	505	507	**505**	32.9	**505**	531.6
3	506	507	**506**	1,818.5	**506**	334.1
4	506	507	**506**	577.2	**506**	266.6
5	505	507	**505**	975.5	**505**	242.9
6	506	507	**506**	407.5	**506**	148.4
7	506	506	**506**	-	**506**	-
8	506	507	**506**	896.4	**506**	176.6
9	506	507	**506**	423.6	**506**	769.7
10	505	507	**505**	766.1	**505**	216.8
			avg.	612.9	**avg.**	283.2

(-) Negligible CPU time because $LB = UB$.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Table A.11: Results for set A and $n = 25$.

#	LB	UB	F1 Sol.	F1 CPU	BSA Sol.	BSA CPU
1	505	507	**505**	590.1	507	7,200.0
2	506	507	**506**	1,491.0	**506**	106.0
3	506	507	**506**	361.1	**506**	161.3
4	506	507	**506**	1,281.8	**506**	176.3
5	505	506	**505**	3,454.2	506	7,200.0
6	506	507	**506**	1,734.2	**506**	466.8
7	505	507	**506**	1,529.4	**506**	691.0
8	507	507	**507**	-	**507**	-
9	505	507	**505**	739.7	**505**	2,023.7
10	506	507	**506**	835.1	**506**	179.6
			avg.	1,201.7	**avg.**	1,820.5

(-) Negligible CPU time because $LB = UB$.

Table A.12: Results for set A and $n = 30$.

#	LB	UB	F1 Sol.	F1 CPU	BSA Sol.	BSA CPU
1	505	507	**505**	1,325.3	**505**	1,729.3
2	506	507	**506**	691.8	**506**	451.4
3	506	507	**506**	2,956.8	**506**	419.0
4	506	507	**506**	1,027.2	**506**	324.9
5	505	507	**505**	2,326.1	**505**	347.8
6	505	506	**505**	671.6	**505**	169.1
7	507	507	**507**	-	**507**	-
8	506	507	**506**	788.0	**506**	213.6
9	506	507	**506**	1,020.9	**506**	1,769.9
10	505	507	**505**	435.7	**505**	533.7
			avg.	1,124.4	**avg.**	595.9

(-) Negligible CPU time because $LB = UB$.

Table A.13: Results for set A and $n = 35$.

#	LB	UB	F1 Sol.	F1 CPU	BSA Sol.	BSA CPU
1	506	507	**506**	718.9	**506**	265.2
2	506	507	**506**	897.2	**506**	811.4
3	506	507	**506**	497.6	**506**	299.7
4	506	507	**506**	907.0	**506**	405.1
5	506	507	**506**	2,266.4	**506**	162.0
6	505	506	**505**	466.4	**505**	349.5
7	506	507	**506**	251.5	**506**	374.6
8	505	507	**505**	1,726.9	507	7,200.0
9	505	507	**505**	1,442.9	**505**	2,203.1
10	506	507	**506**	715.0	**506**	1,259.4
			avg.	989.0	**avg.**	1,333.0

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Table A.14: Results for set A and $n = 40$.

#	LB	UB	F1		BSA	
			Sol.	CPU	Sol.	CPU
1	505	507	**505**	903.7	507	7,200.0
2	505	507	**505**	1,702.6	**505**	579.9
3	505	507	**505**	551.1	**505**	1,245.4
4	506	507	**506**	229.4	**506**	116.1
5	505	507	**505**	2,048.9	**505**	2,516.5
6	505	507	**505**	3,911.2	**505**	1,240.6
7	507	507	**507**	-	**507**	-
8	506	507	**506**	1,011.9	**506**	204.6
9	506	507	**506**	797.4	**506**	348.6
10	506	507	**506**	2,947.1	**506**	365.2
			avg.	1,410.4	**avg.**	1,381.7

(-) Negligible CPU time because $LB = UB$.

Table A.15: Results for set B and $n = 45$.

#	LB	UB	F1		BSA	
			Sol.	CPU	Sol.	CPU
1	755	758	**755**	166.5	**755**	157.5
2	756	756	**756**	-	**756**	-
3	756	758	**756**	1797.3	**756**	381.1
4	758	790	**786**	1336.0	**786**	372.8
5	755	757	**755**	1044.2	**755**	333.0
6	755	768	**767**	679.0	767	172.3
7	756	799	**797**	1143.2	**797**	592.0
8	757	758	**757**	154.6	**757**	94.5
9	755	799	**797**	556.0	**797**	94.7
10	756	794	**791**	2090.9	**791**	539.0
			avg.	897.0	**avg.**	273.7

(-) Negligible CPU time because $LB = UB$.

Table A.16: Results for set B and $n = 50$.

#	LB	UB	F1		BSA	
			Sol.	CPU	Sol.	CPU
1	755	758	**755**	1032.5	**755**	297.6
2	755	770	**769**	861.9	**769**	144.4
3	756	773	**771**	885.8	**771**	616.7
4	757	758	**757**	452.3	**757**	225.9
5	755	757	**755**	301.5	**755**	225.2
6	755	758	**755**	4984.8	755	744.9
7	758	782	**782**	4.2	**782**	63.2
8	755	758	**755**	740.3	**755**	807.8
9	756	799	**798**	721.1	**798**	71.2
10	756	758	**756**	232.4	**756**	355.6
			avg.	1,021.7	**avg.**	355.3

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Table A.17: Results for set B and $n = 55$.

#	LB	UB	F1		BSA	
			Sol.	CPU	Sol.	CPU
1	755	758	**755**	3527.6	**755**	1387.5
2	757	780	**778**	809.2	**778**	291.0
3	755	780	**779**	1274.2	**779**	767.1
4	757	758	**757**	377.5	**757**	160.9
5	757	758	**757**	312.4	**757**	165.2
6	755	791	**788**	895.8	788	473.9
7	757	769	**766**	998.2	**766**	759.4
8	757	758	**757**	281.4	**757**	104.2
9	755	802	**801**	739.7	**801**	197.0
10	756	758	**756**	2132.8	**756**	322.6
			avg.	1,134.9	**avg.**	462.9

Table A.18: Results for set B and $n = 60$.

#	LB	UB	F1		BSA	
			Sol.	CPU	Sol.	CPU
1	755	782	**781**	1191.6	**781**	720.5
2	755	758	**755**	290.3	**755**	323.7
3	756	758	**756**	805.0	**756**	467.6
4	757	761	**759**	904.3	**759**	982.7
5	755	758	**755**	3916.9	**755**	438.3
6	755	758	**755**	233.0	755	208.6
7	757	787	**785**	101.0	**785**	110.8
8	755	758	**755**	1531.5	**755**	220.7
9	756	786	**785**	750.1	**785**	59.3
10	756	806	**805**	938.3	**805**	337.7
			avg.	1,066.2	**avg.**	387.0

Table A.19: Results for set B and $n = 65$.

#	LB	UB	F1		BSA	
			Sol.	CPU	Sol.	CPU
1	755	758	**755**	1904.2	**755**	721.4
2	799	799	**799**	-	**799**	-
3	755	804	**802**	965.7	**802**	475.1
4	757	758	**757**	1407.0	**757**	571.7
5	757	758	**757**	289.3	**757**	132.7
6	756	758	**756**	171.5	756	491.3
7	757	758	**757**	199.9	**757**	116.6
8	756	758	**756**	455.6	**756**	280.6
9	756	758	**756**	6405.7	**756**	1389.5
10	756	787	**786**	347.1	**786**	529.0
			avg.	1,214.8	**avg.**	470.8

(-) Negligible CPU time because $LB = UB$.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Table A.20: Results for set B and $n = 70$.

#	LB	UB	F1		BSA	
			Sol.	CPU	Sol.	CPU
1	755	758	**755**	771.6	**755**	525.5
2	755	758	**755**	226.6	**755**	75.7
3	755	758	**755**	914.9	**755**	410.9
4	757	758	**757**	1106.9	**757**	794.1
5	756	758	**756**	350.2	**756**	1252.6
6	755	762	**761**	383.3	761	124.1
7	757	758	**757**	105.0	**757**	158.2
8	756	758	**756**	267.4	**756**	256.0
9	755	758	**755**	1721.1	**755**	384.3
10	756	778	**777**	1192.9	**777**	341.4
			avg.	704.0	**avg.**	432.3

Table A.21: Results for set C and $n = 75$.

#	LB	UB	F1		BSA	
			Sol.	CPU	Sol.	CPU
1	1005	1181	1180	7200.0	**1178**	2446.1
2	1007	1009	**1007**	6215.9	**1007**	1927.9
3	1005	1184	1184	7200.0	**1182**	2906.5
4	1006	1108	1108	7200.0	1108	7200.0
5	1007	1356	1194	7200.0	**1192**	2363.0
6	1007	1159	1123	7200.0	1122	7200.0
7	1007	1200	**1200**	27.4	**1200**	163.9
8	1007	1174	**1174**	19.6	**1174**	129.3
9	1006	1075	1074	7200.0	1072	7200.0
10	1069	1189	**1188**	3629.5	**1188**	484.0
			avg.	5,309.2	**avg.**	3,202.1

Table A.22: Results for set C and $n = 80$.

#	LB	UB	F1		BSA	
			Sol.	CPU	Sol.	CPU
1	1005	1375	1175	7200.0	**1173**	2050.4
2	1005	1009	1007	7200.0	1006	7200.0
3	1006	1009	**1006**	6183.1	**1006**	6393.4
4	1008	1282	1203	7200.0	**1201**	3882.4
5	1007	1037	**1036**	2169.3	**1036**	188.6
6	1007	1118	**1116**	4819.6	**1116**	917.2
7	1008	1201	**1201**	20.6	**1201**	330.1
8	1007	1019	**1016**	4675.0	**1016**	6255.0
9	1020	1194	**1192**	6823.2	**1192**	588.0
10	1116	1208	**1205**	5302.6	**1205**	2062.8
			avg.	5,159.3	**avg.**	2,986.8

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Table A.23: Results for set C and $n = 85$.

#	LB	UB	F1 Sol.	F1 CPU	BSA Sol.	BSA CPU
1	1006	1094	1049	7200.0	1094	7200.0
2	1007	1009	1009	7200.0	1008	7200.0
3	1006	1028	**1026**	7010.9	**1026**	1546.3
4	1007	1187	1187	7200.0	**1185**	3396.3
5	1006	1083	**1081**	7003.5	**1081**	5235.5
6	1006	1009	**1006**	5009.6	1007	7200.0
7	1006	1389	1197	7200.0	**1194**	2710.2
8	1007	1106	**1105**	502.2	**1105**	783.1
9	1007	1009	**1007**	3472.4	**1007**	2250.1
10	1166	1166	**1166**	-	**1166**	-
			avg.	5,179.9	**avg.**	3,752.1

(-) Negligible CPU time because $LB = UB$.

Table A.24: Results for set C and $n = 90$.

#	LB	UB	F1 Sol.	F1 CPU	BSA Sol.	BSA CPU
1	1007	1009	**1007**	2799.5	1008	7200.0
2	1006	1009	**1006**	4237.1	**1006**	2298.4
3	1006	1009	1007	7200.0	**1006**	2186.1
4	1007	1087	1064	7200.0	1063	7200.0
5	1062	1062	**1062**	-	**1062**	-
6	1007	1196	1195	7200.0	**1193**	943.3
7	1007	1110	1110	7200.0	**1106**	5685.1
8	1007	1094	**1094**	15.8	**1094**	89.9
9	1007	1077	**1073**	6513.2	1074	7200.0
10	1049	1049	**1049**	-	**1049**	-
			avg.	4,237.3	**avg.**	3,280.3

(-) Negligible CPU time because $LB = UB$.

Table A.25: Results for set C and $n = 95$.

#	LB	UB	F1 Sol.	F1 CPU	BSA Sol.	BSA CPU
1	1006	1176	1176	7200.0	**1174**	2495.8
2	1007	1091	1090	7200.0	**1088**	3382.6
3	1007	1009	1008	7200.0	**1007**	2024.1
4	1007	1140	1140	7200.0	**1137**	4456.9
5	1007	1145	1144	7200.0	**1143**	1048.7
6	1006	1056	1054	7200.0	**1054**	2985.1
7	1007	1172	**1172**	20.0	**1172**	346.7
8	1006	1009	**1006**	5809.3	**1006**	5344.9
9	1019	1019	**1019**	-	**1019**	-
10	1006	1009	1007	7200.0	**1006**	6684.6
			avg.	5,622.9	**avg.**	2,876.9

(-) Negligible CPU time because $LB = UB$.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Table A.26: Results for set C and $n = 100$.

#	LB	UB	F1		BSA	
			Sol.	CPU	Sol.	CPU
1	1007	1009	**1007**	3161.6	**1007**	6031.5
2	1007	1105	1103	7200.0	**1103**	2915.4
3	1007	1108	**1107**	3624.2	**1107**	597.8
4	1007	1362	1204	7200.0	1202	7200.0
5	1007	1009	**1007**	3426.3	**1007**	1270.5
6	1007	1138	1137	7200.0	**1136**	1461.0
7	1006	1099	1098	7200.0	1099	7200.0
8	1151	1151	**1151**	-	**1151**	-
9	1006	1009	1007	7200.0	**1006**	1774.4
10	1007	1009	**1007**	2323.6	**1007**	2200.0
			avg.	4,853.6	**avg.**	3,065.1

(-) Negligible CPU time because $LB = UB$.

ACCEPTED MANUSCRIPT