A Benders decomposition-based framework for solving quay crane scheduling problems

A Benders Decomposition-based Framework for Solving Quay Crane Scheduling Problems

Defeng Sun[1], Lixin Tang[1], Roberto Baldacci[2]

[1] Institute of Industrial Engineering & Logistics Optimization, Northeastern University, Shenyang, 110819, P. R. China, sundefeng@ise.neu.edu.cn, qhjytlx@mail.neu.edu.cn

[2] Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi", University of Bologna, 47521 Cesena, Italy, r.baldacci@unibo.it

Abstract

In this paper, we study the Quay Crane Scheduling Problem (QCSP) in container terminals. We describe a new mathematical formulation for the QCSP and by addressing the structure of workload assignments we develop an easier way to handle non-crossing constraints. The proposed mathematical formulation is used in an exact solution framework based on logic-based Benders decomposition. The proposed approach decomposes the problem into a workload-assignment master problem and operation-sequence slave subproblems. Logic-based cuts are proposed to ensure the convergence of the approach. Computational results show the effectiveness of the proposed solution approach.

***Keywords*:** container terminal; quay crane scheduling; logic-based Benders decomposition.

1 Introduction

Along with the explosive growth of globalization, maritime transportation has rapidly increased during the last few decades, especially in terms of container flows. The increment of container flows raises challenges for the operational efficiency of container terminals, which play an essential role in global container logistics as an intermodal interface. Quay cranes (QCs), which perform the container loading and unloading operations between vessels and terminals, are one of the typical operational resources in container terminals. These operations are a frequent bottleneck due to resource limitations.

Container terminal operations usually comprise four main stages: assigning berths for vessels, loading and unloading containers between vessels and land-side trucks, transporting containers by land-side trucks between berths and the storage yard, and finally loading and unloading containers in the storage yard (for related literature reviews, see Vis and de Koster 2003, Stahlbock and Voß (2008) and Bierwirth and Meisel (2010, 2015). There are many different decisions to be made in these four operation stages, such as berth allocation (Imai et al., 2001), quay crane assignment and scheduling (Liu et al., 2006), yard crane scheduling (Ng and Mak, 2005), storage space allocation (Zhang et al., 2003), truck scheduling (Tang et al., 2014), and container reshuffling and stacking (Tang et al., 2015).

In this paper, we focus on the second stage of container terminal operations planning. In particular, we considered the Quay Crane Scheduling Problem (QCSP) involving one vessel and multiple QCs, while also considering the QCs' moving times.

The QCSP, a significant issue in container terminal operations, aims to minimize the completion time of operations in a vessel (the "makespan") by determining the optimal operation sequence of the workload (loading and unloading). These operations are usually implemented by QCs that are mounted on the same rail and work simultaneously. Therefore, spatial constraints need to be considered while making scheduling decisions in order to avoid crane interference. One of the most interesting spatial constraints is the non-crossing constraint—that is, the QCs cannot cross over each other—which we take into account when modeling the QCSP.

In this paper, we describe an exact method to solve the QCSP which considers QCs'

moving times. The major contribution of this paper is that we have designed a novel master-slave modeling technique as well as an efficient and flexible logic-based Benders decomposition framework which together optimally solve the QCSP. Our distinct contributions are as follows:

- We propose a new mathematical formulation which innovatively models the workload-assignment structure and further simplifies the consideration of QC interference constraints.
- We propose an efficient and flexible Benders decomposition framework based on the new formulation and on logic-based Benders decomposition — and we prove its convergence to an optimal solution.
- We perform computational results on medium- and large-sized instances. The results about large-sized instances involving up to 35 bays and 6 cranes show that the proposed Benders framework outperforms state-of-the-art methods.

The remainder of this paper is organized as follows. The next section reports a literature review about other works on closely related problems to the QCSP considered in this paper. Section 2 provides a problem description and a mathematical formulation of the QCSP. Section 3 presents the novel constructive Benders framework and the master-slave modeling technique for QCSP and provides a proof of optimality of the proposed method. Computational experiments are carried out in Section 4 to illustrate the performance of the proposed solution approach. Finally, we conclude the paper and indicate future research directions in Section 5.

1.1 Literature review

Daganzo (1989) was the first to study the crane scheduling problem. The existing literature provided different ways of modeling the QCSP with various problem characteristics. A classification scheme and a literature review for QCSP formulations can be found in the works of Bierwirth and Meisel (2010) and Bierwirth and Meisel (2015). Based on the classification scheme introduced by Bierwirth and Meisel, the QCSP considered in this paper belongs to the class of *Bay|move|cross|max(compl)* QCSPs, i.e., tasks refer to single bays,

travel time for crane movement is respected, non-crossing of QCs is respected and problem is to minimize the completion of the latest finished task – the reader is referred to Bierwirth and Meisel for details about the classification scheme.

The QCSP with tasks defined on the basis of container groups has been introduced by Kim and Park (2004). Their model considers QC operations in detail by taking crane attributes, crane interference, and precedence relations among tasks into account. The model of Kim and Park (2004) has been refined by Moccia et al. (2006) by incorporating travel times for QCs. Lim et al. (2004) also considered spatial constraints and assigned bay areas to QCs assuming individual throughput rates for the cranes – they aim at the maximization of the total throughput. Lim et al. (2007) studied the problem with respect to schedules where the QCs have identical directions of movement from lower to upper bays. The authors also showed that that there is always an optimal schedule among the unidirectional ones in the case of one task per bay. Unidirectionality of schedules is also supposed by Liu et al. (2006) who embedded the property in a MIP formulation for the QCSP. Zhu and Lim (2006) studied the scheduling decisions to minimize the latest departure time of a vessel under a non-crossing constraint for the QCs. Lee et al. (2008) provided a mixed-integer programming model for QCSP, which for the first time considered non-crossing constraints. Further mathematical formulations including non-crossing constraints can be found in Sammarra et al. (2007) and Bierwirth and Meisel (2009). Most of the models mentioned above contain both workload-assignment and operation-sequence binary variables, but deal differently with the non-crossing constraints. An interesting recent work by Guan et al. (2013) presented a time-space network flow model for QCSP which regarded the movements and operations of QCs as the flows in a time-space network. Unlike most previous QCSP models, theirs took QC moving time into consideration in their algorithms. Small-sized instances, involving up to 8 bays and three cranes, were solved to optimality by Guan et al. (2013) by running their model using a general integer programming solver. Al-Dhaheri et al. (2015) also took the moving time of QCs into account, although in an approximative manner.

Various solution approaches have been developed to study QCSP. Peterkofsky and Daganzo (1990) proposed a Branch-and-Bound algorithm to solve the QCSP to optimality. To

overcome the computational difficulty of the Branch-and-Bound method, Kim and Park (2004) developed a heuristic search algorithm. Moccia et al. (2006) developed a Branch-and-Cut algorithm incorporating several families of valid inequalities, and solved instances which cannot be solved by Kim and Park's algorithm. Bierwirth and Meisel (2009) proposed a fast heuristic solution procedure, of which the core is the Branch-and-Bound algorithm. The heuristic took advantage of more efficient criteria for branching and bounding the search for a subset of above-average quality schedules with respect to the impact of crane interference. Sampaio et al. (2016) chose a different approach, utilizing combinatorial Benders cuts to decompose the QCSP into a mixed-integer vehicle-routing problem and a linear scheduling-time problem.

Due to the computational complexity of QCSP, most exact algorithms perform poorly in large-scale instances. To address this challenge, a number of heuristic and meta-heuristic approaches have been proposed to obtain near-optimal solutions for large-scale instances. Sammarra et al. (2007) first decomposed QCSP into a routing problem and a scheduling problem, and then proposed a tabu-search approach to solve the routing problem, while a local search technique was used to solve the scheduling problem. Simple approximation heuristics, as well as a simulated annealing heuristic using random neighborhood generation, were provided by Lim et al. (2007) for large-scale problems. Tavakkoli-Moghaddam et al. (2009) proposed a genetic algorithm, with solution representation and operator design appropriate for QCSP, to efficiently solve real-sized problems. To handle medium-sized and large-sized instances (involving up to 50 bays and 20 cranes), Guan et al. (2013) developed a Lagrangian relaxation approach to obtain near-optimal solutions and two approximation algorithms.

The exact approach we propose is based on the logic-based Benders' decomposition, that was formally developed by Hooker (2000), and applied with success by Hooker and Ottosson (2003) to 0–1 programming and by Hooker (2007) to planning and scheduling problems. The approach was later specialized to mixed integer programming by Codato and Fischetti (2006) who introduced the so-called combinatorial Benders' cuts. Logic-based Benders decomposition is a generalization of classical Benders decomposition that can be applied to a

much wider variety of combinatorial optimization problems because the subproblem may be any combinatorial problem, not necessarily a linear or nonlinear programming problem (Hooker and Ottosson, 2003).

2 Problem description and mathematical formulation

This paper addresses the QCSP considering one vessel and multiple quay cranes. After a vessel berths at a wharf, crane scheduling determines the sequence of unloading and loading operations that the QCs will perform so that the makespan is minimized (Moccia et al. 2006).

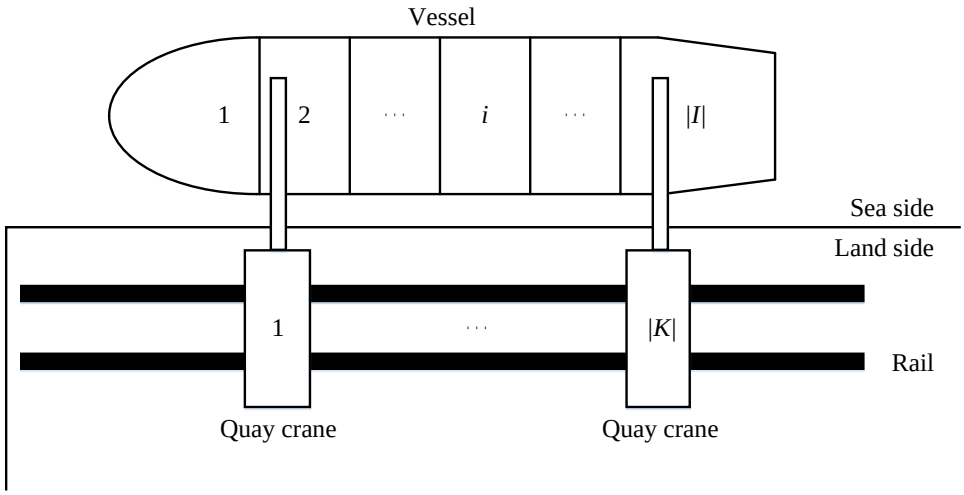

Fig. 1. Schematic diagram of QCSP (Lee et al. 2008).

Figure 1 schematically shows two QCs working on a vessel. As shown in the figure, all QCs are mounted on the same rail parallel to the vessel. This is the reason why the QCs cannot cross over each other. The QC is made of heavy steel, thus the movement of the crane is slow. In this paper, we account for the moving time needed by cranes to travel along the rail. The moving time for a crane to travel between two consecutive bays is regarded as a constant number, denoted as v. As is generally true in real life, this paper assumes that all QCs work at the same operating rate, denoted as u.

We consider the instance of one vessel loaded/unloaded by $|K|$ QCs. The total workload is divided into $|I|$ bays (see Fig. 1). I and K are the index sets of QCs and bays, respectively.

Parameter $w_{i,}\ i \in I_{,}$ indicates the operation time needed to complete the workload in bay *i*. Only one QC can work on a bay at a time. Once started, a QC will not stop its operation on a bay until it completes the bay, i.e. preemption is not allowed.

The decisions for QCSP can be categorized into two types: (i) workload assignment decisions and (ii) operation sequence decisions. Workload assignment decisions determine which QC is assigned to handle a given bay, and operation sequence decisions determine the operation sequence between two given bays. In the rest of this section, we first introduce a new approach to modeling the workload assignments, and then we give the corresponding mathematical formulation.

2.1 Description of workload assignments

To illustrate the modeling technique, we first introduce the concept of *work zone* for each QC.

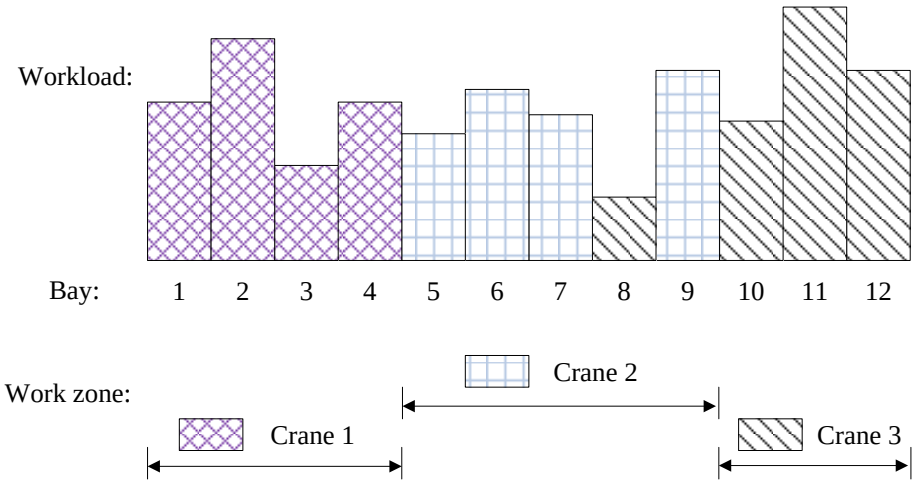

Fig. 2 Assignment structure of QCSP.

Figure 2 shows an optimal scheduling solution of a 12-bay and 3-QC instance. We index the QCs along the same directions as the indices for bays along the vessel. In this optimal solution, bays 1-4 are handled by QC 1, bays 5-7 and 9 are handled by QC 2, bay 8 and bays 10-12 are handled by QC 3. Since bay 9 is the last bay that QC 2 works on along the index axis of bays, we refer to it as the *border bay* of QC 2. Similarly, bays 4 and 12 are the border

bays of QCs 1 and 3, respectively. The *work zone* of a given QC consists of all the bays located between the border bay of the preceding QC (not included) and its own border bay. Therefore, in Figure 2, the work zone of QC 2 is from bays 5 to 9, and the work zone of QC 3 is from bays 10 to 12.

Each work zone is composed of consecutive bays. Given this concept, we could categorize the assignment decisions into two types: *group assignments* and *cross-group assignments*.

The aim of group-assignment decisions is to group the bays into several consecutive work zones, each assigned to a certain QC. For bay *i* located in the work zone of QC *k*, we say that bay *i* is group-assigned to QC *k*. Obviously, the group-assignment solution is a feasible solution of QCSP; each QC can move without interference in its corresponding work zone.

The aim of cross-group assignment decisions is to identify the *crossing bays* (i.e., bays outside the QC's work zone) in the optimal solution of QCSP. For example, as shown in Figure 2, bay 8 is located in the work zone of QC 2 but is assigned to QC 3 in the optimal solution. Therefore, bay 8 is a crossing bay. If bay *i* is a crossing bay located in the work zone of QC *k*, and is finally assigned to QC *l*, we say that bay *i* is group-assigned to QC *k* and cross-group-assigned to QC *l*.

Obviously, crossing bays can cause interference between QCs. In the rest of this section, we describe a new mathematical formulation based on the concepts of work zone and crossing bay illustrated above. The model is well suited to a solution using a decomposition-based approach.

2.2 Formulation of the QCSP

Let Z_{ki} be a binary variable taking value 1 if bay *i* is the border bay of the work zone of QC *k*, 0 otherwise. Let Y_{kli} be a binary variable taking value 1 if bay *i* is group-assigned to QC *k* and cross-group-assigned to QC *l* ($l>k$), and 0 otherwise. A nonnegative variable δ_k is used to indicate the minimum additional moving time needed by QC *k* to visit all its cross-group-assigned bays from the left side of its work zone. Further, nonnegative variable

σ_k indicates the total operating and moving time for QC *k* to handle all the bays in its work zone. The mathematical formulation is as follows:

(QCSP) Minimize C (1)

Subject to

$$C \geq \sigma_k + \frac{1}{u} \cdot \left(\sum_{i \in I} \sum_{l<k} w_i Y_{lki} - \sum_{i \in I} \sum_{l>k} w_i Y_{kli} \right) + \delta_k \qquad \forall k \in K \tag{2}$$

$$C \geq f(\mathbf{Z}, \mathbf{Y}) \tag{3}$$

$$\delta_k \geq v\left(j - i + 1\right) \cdot \left(Z_{k-1,j} + \sum_{l<k} Y_{lki} - 1 \right) \qquad \forall k \in K \setminus \{1\}, i, j \in I, i < j \tag{4}$$

$$\sigma_k \geq \sum_{i \in I} t_i Z_{ki} - \sum_{i \in I} t_i Z_{k-1,i} - v \qquad \forall k \in K \setminus \{1\} \tag{5a}$$

$$\sigma_k \geq \sum_{i \in I} t_i Z_{ki} \qquad k = 1 \tag{5b}$$

$$\sum_{l>k} Y_{kli} \leq \sum_{j<i} Z_{k-1,j} + \sum_{j>i} Z_{kj} - 1 \qquad \forall k \in K \setminus \{1\}, i \in I \tag{6a}$$

$$\sum_{l>k} Y_{kli} \leq \sum_{j>i} Z_{kj} \qquad k = 1, i \in I \setminus \{1\} \tag{6b}$$

$$\sum_{i \in I} Z_{ki} = 1 \qquad \forall k \in K \tag{7a}$$

$$Z_{|K||I|} = 1 \tag{7b}$$

$$\sum_{i \in I} i \cdot Z_{ki} \geq \sum_{i \in I} i \cdot Z_{k-1,i} + 1 \qquad \forall k \in K \setminus \{1\} \tag{8}$$

$$Y_{kli}, Z_{ki} \in \{0,1\}, \delta_k, \sigma_k, C \geq 0 \qquad \forall k, l \in K, l > k, i \in I, \tag{9}$$

where t_i is a parameter indicating the total operating and moving time for a QC to handle all the bays left to bay *i* (included), i.e., $t_i = \frac{1}{u} \cdot \sum_{j=1}^{i} w_i + (i-1) \cdot v, \ \forall i \in I$.

Constraints (2) are used to calculate a lower bound on the optimal makespan by disregarding crane interferences. For each QC, there are three terms on the right-hand-side of (2): (i) the total operating and moving time needed to complete all group-assigned bays, (i) the adjusted operating time according to the cross-group-assignments, and (iii) the additional moving time needed to access the cross-group-assigned bays. Constraint (3) calculates the

real makespan, incorporating the idle times that originate from interference between cranes - they can be expressed in different forms—based, for example, on the Guan et al. (2013), Lee et al. (2008), or Liu et al. (2006) models (see below). Constraints (4) calculate the additional moving time needed by each QC to access its cross-group-assigned bays without considering crane interference. Constraints (5a) and (5b) are to compute the total operating and moving time for each QC to handle all the bays located in its work zone. Constraints (6a) and (6b) ensure the logical correctness of cross-group-assignment decisions. Constraints (7a) and (7b) guarantee there is one and only one border bay for each work zone; constraints (8) ensure that the sequence of bays in each QC's work zone remains the same as the QC's index sequence.

Function $f(\mathbf{Z},\mathbf{Y}):\{0,1\}^{|K||I|}\times\{0,1\}^{|K|\frac{(|K|-1)}{2}|I|}\mapsto R_+$ calculates the real makespan by considering non-crossing constraints and, based on the mathematical formulation of Lee et al. (2008), can be defined as follows:

$$f(\mathbf{Z},\mathbf{Y}) = \text{Minimize} \quad \max_{i\in I} c_i \tag{10}$$

$$c_i - p_i \geq 0 \qquad \forall i \in I \tag{11}$$

$$c_i - \left(c_j - p_j\right) + M\cdot H_{ij} \geq 0 \qquad \forall i,j \in I \tag{12}$$

$$\left(c_i + m_{ij}H_{ij}\right) - \left(c_j - p_j\right) - M\cdot\left(1-H_{ij}\right) \leq 0 \qquad \forall i,j \in I \tag{13}$$

$$M\cdot\left(H_{ij}+H_{ji}\right) \geq \sum_{k\in K} k\cdot G_{ki} - \sum_{k\in K} k\cdot G_{kj} + 1 \qquad \forall i,j \in I, i<j \tag{14}$$

$$G_{ki} = \sum_{j<i} Z_{k-1,j} - \sum_{j<i} Z_{kj} - \sum_{l>k} Y_{kli} + \sum_{l<k} Y_{lki} \qquad \forall i \in I, k \in K \tag{15}$$

$$G_{ki}, H_{ij} \in \{0,1\}, c_i \geq 0 \qquad \forall i,j \in I\,. \tag{16}$$

In the above formulation, G_{ki} is a binary variable taking value 1 if bay i is handled by QC k, 0 otherwise, and H_{ij} is a binary variable equal to 1 if bay i finishes no later than bay j starts; the nonnegative variable c_i indicates the completion time for bay i. Parameter m_{ij} is the moving time needed by a QC to move from bay i to j, i.e. $m_{ij} = |i\text{-}j|\ v$. Parameter p_i indicates the processing time of bay i by a QC, i.e. $p_i = w_i / u$ and M is a nonnegative big number. In Section 3.2, we describe a simple algorithm to compute function $f(\mathbf{Z},\mathbf{Y})$.

As mentioned above, the advantage of this QCSP model is that it has a good

decomposition structure based on constraint (3). In the next section, we will explain the advantage in detail and propose a Benders decomposition-based framework to optimally and efficiently solve the QCSP.

3 Logic-based Benders decomposition

Benders decomposition was first proposed by Benders (1962) to efficiently solve mixed integer programming models. Basically, it decomposes the original problem into two simpler ones, i.e. an integer master problem (BMP) and a linear slave problem or subproblem (BSP), which are solved in an iterative fashion by utilizing the solution of one in the other. In every iteration, the master problem actually behaves as a relaxation of the original problem and provides fixed integer variable values for the slave problem to obtain a feasible solution. A Benders cut is constructed and added to the master problem in the next iteration to exclude the solution just obtained in the last master problem. Therefore, each solution of the master problem must satisfy all the Benders cuts generated so far to avoid repetition. The master problem and the slave problem are solved in this iterative fashion until an optimal solution to the original problem is obtained. The generation of Benders cuts is the core of Benders decomposition algorithm. Indeed, valid Benders cuts guarantee the convergence of the iterations to the optimal solution of the original problem, and also the cuts determine how fast the algorithm converges.

The classic Benders decomposition algorithm was proposed for linear programming problems (Benders, 1962), the cut generation of which is based on the strong duality property of linear programming. Geoffrion (1972) has extended it to a larger class of mathematical programming problems. For more general integer programming, logic-based Benders decomposition was proposed to generate valid integer Benders cuts (Hooker, 2000, Hooker and Ottosson, 2003). The key is to generalize the linear programming dual used in the classical method to an “inference dual” and the solution of the inference dual takes the form of a logical deduction that yields valid Benders cuts.

In this paper, we present an efficient, flexible decomposition solution framework based on logic-based Benders decomposition. In particular, we decompose the problem according to

its logical structure, categorizing the workload assignment decisions as the master problem and the operation sequence decisions as the subproblem. The resulting Benders master problem and subproblem involve both integer and linear variables. In every iteration, the master problem is first solved to obtain reasonable assignment decisions regardless of crane interference; then the corresponding subproblem is solved to obtain the optimal operation sequence solution (maintaining the assignment decisions of the master problem). The subproblem takes into account the idle time that originates from interference between cranes, which is ignored in the master problem. Solving the subproblem generates logic-based Benders cuts and adds them to the master problem in the next iteration as feedback information. Optimal stop criteria based on lower and upper bounds are provided, to guarantee that the algorithm will terminate at the optimal solution - details are given below.

3.1 Benders master problem

We first define the Benders master problem without the crane interference restrictions but with the minimum operation and moving times needed according to the assignment decisions. The Benders master problem provides a lower bound on the optimal QCSP makespan, which can be iteratively updated to approach the optimal objective value of QCSP.

The Benders master problem is as follows:

(BMP) Minimize C (17)

Subject to

Constraints (2), (4), (5)-(9).

A solution of BMP provides a valid lower bond on the optimal makespan since crane interference restrictions (i.e., constraints (3)) are ignored.

Given the current assignment solution provided by BMP, the subproblem consist of solving the operation-sequence problem to obtain a feasible solution of QCSP. The inclusion of work zones and cross-group-assignments in the general assignment structure enables us to construct more efficient Benders slave problems and Benders cuts. Details will be given in later sections.

To further improve the quality of assignment decisions, we add the following valid

inequalities to BMP:

$$\sum_{j<i}\left(\sum_{l\le k-2} Y_{l,k-1,j} + Z_{k-2,j}\right) \ge \sum_{l<k} Y_{l,k,i} \qquad \forall k \in K \setminus \{1,2\}, i \in I \setminus \{1\}. \tag{18}$$

$$\sum_{l<k} Y_{l,k,i} = 0 \qquad \forall k \ge i,\ k \in K, i \in I. \tag{19}$$

Inequalities (18) indicate that if $k_1 < k_2$, there is no bay, say i, both cross-group-assigned to QC k_2 and smaller in index than any bay assigned to QC k_1. The inequalities are based on the observation that there will be some idle time for QC k_1 when QC k_2 is working in the corresponding bay i, due to the non-crossing requirement. The idle time is enough for QC k_1 to handle the workload in bay i; therefore, rather than assigning bay i to k_2, we can assign it to k_1—without changing the optimal value of BMP. Inequalities (19) are natural considering the problem characteristic of the leftmost bay that each QC can reach.

In the following section, we describe the Benders slave problem, which makes the operation-sequence decisions on the basis of the current assignment solution and generates the corresponding logic-based Benders cuts.

3.2 Benders subproblem

The subproblem takes into account the idle times that originate from interference between cranes when calculating the real makespan. Note that if there are no crossing bays between the work zones of any two QCs, there will be no interference between them, and therefore the Benders subproblem can be further decomposed.

The Benders subproblem is defined by decomposing the original operation-sequence subproblem into several smaller independent problems, by putting all work zones that contain related crossing bays together into one problem. For example, in Figure 2, crossing bay 8 is related to the work zones of QCs 2 and 3 therefore QCs 2 and 3 and their work zones are considered together in one problem.

Let $(\hat{\mathbf{Z}}, \hat{\mathbf{Y}}, \hat{\boldsymbol{\delta}}, \hat{\boldsymbol{\sigma}}, \hat{C})$ be a feasible BMP solution. We propose the following procedure to generate Benders slave subproblems, taking full advantage of the work zone structure proposed in Section 2.1.

Step 1. Initialize k =1 and n =1, where k is the index of cranes, and n is the index of slave subproblems.

Step 2. If $k > |K|$, stop; otherwise, for the n^{th} slave subproblem, initialize the set of involved cranes $K_n = \emptyset$, and the set of involved bays $I_n = \emptyset$.

Step 3. Check if $\sum_{i \in Q_k} \sum_{l>k} \hat{Y}_{kli} = 0$ for QC k, where Q_k is the set of bays that belong to the work zone of QC k, and check if the index of k is greater than that of any QC in set K_n . If so, there is no crossing bay; then find the bay i that satisfies $\hat{Z}_{ki} = 1$ (i.e. the border bay of QC k); mark this bay as i_n, and go to Step 4; otherwise, let k=k+1, add k and all l that satisfy $\sum_{i \in Q_k} \hat{Y}_{kli} > 0$ to K_n, and go to Step 3.

Step 4. Add the bays from i_{n-1}+1 through i_n to set I_n (we assume i_0=0).

Step 5. Construct a slave subproblem for sets I_n and K_n, denoted as BSP^n. Set n=n+1, k=k+1, and go to Step 2.

As shown in Lim et al. (2007), once bays have been assigned to cranes, an optimal operation schedule can be obtained by a simple unidirectional algorithm as follows.

Let $C[k]$ be the completion time of QC k, let s_i be the time at which service starts at bay i, and let α_i be the index of the QC that bay i is assigned to. We apply the following algorithm to solve each slave subproblem BSP^n to optimality.

1: Initialization: $C[k]$=0 for $k \in K_n$, s_1=0, $pos[k]$=0 for $k \in K_n$ (position of crane k in I_n);
2: for $i \in I_n$ in ascending sequence, do
3: $s_i = \max\{C[k]: k \geq \alpha_i\}$;
4: if ($pos[\alpha_i]$=0) then
$C[\alpha_i] = s_i + w_i/u$;
5: else
$C[\alpha_i] = |pos[\alpha_i]-i| \nu + s_i + w_i/u$;
6: $pos[\alpha_i] = i$;
7: end for
8: return $\max\{C[k]: k \in K_n\}$.

Based on the above algorithm, to further strengthen the lower bound obtained from BMP, we add an additional term to the right hand side of constraints (2), called approximated idle

time θ_k , and defined as follows:

$$\theta_k \geq \sum_{j \leq i} \sum_{l \leq k} Y_{l,k+1,j} - M \cdot \left(1 - Y_{k,k+1,i}\right)$$
$$- \left(\delta_k + t_{i-1} - \sum_{j<i} t_j z_{k-1,j} - \sum_{j<i} \sum_{l \geq k+1} Y_{klj} + \sum_{j<i} \sum_{l<k} w_j Y_{lkj} \right) \quad \forall 1 \leq k \leq |K| - 1, i \in I \,. \qquad (20)$$

Hereafter, we refer to problem BMP as the problem defined by (17), (2), (4), (5)-(9), (18), (19) and (20).

After all the slave subproblems (BSP^n) are solved, the largest makespan obtained is a valid upper bound on the optimal makespan. By starting with an initial upper bound, iterative solutions of BMP and BSP update lower and upper bounds to gradually approach the optimal value. In the following, we first describe the procedure we used to compute a valid upper bound. Then a complete Benders decomposition algorithm is presented, and its finite convergency to optimality is proved.

3.3 Computing an initial upper bound

The computation of the upper bound is based on the observation that group assignments provide a feasible solution for the QCSP, since there will be no QC collisions (according to group assignments only). Thus, if BMP is modified by forbidding any cross-group assignments (i.e., by eliminating variables Y_{kli}), the optimal objective value obtained is a valid upper bound on the optimal makespan. The formulation of the resulting problem is as follows.

(BMP_R) Minimize C (21)

Subject to

$$C \geq \sum_{i \in I} t_i Z_{ki} - \sum_{i \in I} t_i Z_{k-1,i} - v \qquad \forall k \in K \setminus \{1\} \qquad (22a)$$

$$C \geq \sum_{i \in I} t_i Z_{ki} \qquad k = 1 \qquad (22b)$$

$$\sum_{i \in I} Z_{ki} = 1 \qquad \forall k \in K \qquad (23a)$$

$$Z_{|K||I|} = 1 \qquad (23b)$$

$$\sum_{i\in I} i\cdot Z_{ki} \geq \sum_{i\in I} i\cdot Z_{k-1,i} + 1 \qquad \forall k \in K\setminus\{1\} \tag{24}$$

$$Z_{ki} \in \{0,1\}, C \geq 0 \qquad \forall k \in K, i \in I\ . \tag{25}$$

We denote by UB0 the optimal solution cost of formulation BMP_R. The following proposition holds.

Proposition 1. Given a bay $i \in I$ and a QC $k \in K$, if $t_{|I|} - t_i >$ (|*K*|−*k*) UB0, where t_i denotes the total operating and moving time for a QC to handle all the bays left to bay *i* (included), then $\sum_{j\leq i} Z_{kj} = 0$ holds in any optimal solution.

Proof. (By contradiction) Assume that $\sum_{j\leq i} Z_{kj} = 1$ in an optimal solution, then the first *k* QCs will not operate on bays that are on the right of bay *i*. Therefore, due to the condition $t_{|I|} - t_i >$ (|*K*|−*k*) UB0, UB0 is a strict lower bound on the makespan of the (|*K*|−*k*) QCs operating on the right of bay *i*, a contradiction. □

The above proposition is used to reduce the number of variables **Z** of the master problem BMP.

3.4 Logic-based Benders algorithm

The logic-based Benders algorithm is an iterative process alternating between solving the master problem BMP, thus computing a lower bound LB, and the subproblem BSP that computes an upper bound UB on the optimal makespan. If the lower bound LB is equal to the upper bound UB then the algorithm terminates with an optimal solution. If LB<UB, then a Benders cut is computed and added BMP, and the process repeats.

Let $(\hat{\mathbf{Z}}, \hat{\mathbf{Y}}, \hat{\boldsymbol{\delta}}, \hat{\boldsymbol{\sigma}}, \hat{C})$ be an optimal BMP solution. Let BSP^n be the subproblem having the maximum makespan and let C^n be the corresponding value. Notice that given solution $(\hat{\mathbf{Z}}, \hat{\mathbf{Y}}, \hat{\boldsymbol{\delta}}, \hat{\boldsymbol{\sigma}}, \hat{C})$, BSP always admits a feasible solution. Let KI_n be the set of pairs of QCs and bays involved in slave subproblem BSP^n, i.e., $KI_n = \{(k,i): k \in K_n, i \in I_n\}$, and let KI'_n be defined as $KI_n \cup \{(k_1,i_1)\}$ where k_1 is the QC preceding the QC having minimum index in K_n, and i_1 is the border bay of k_1, that is $\hat{Z}_{k_1,i_1} = 1$. For each *k*, define N_k as the set of QCs *l* ,

$l \in K_n$, that satisfy $\sum_{i \in I_n} \hat{Y}_{kli} \geq 1$. Let LB0 be a valid lower bound on the optimal makespan. The Benders cut is defined as follows:

$$C \geq g_{(\hat{\mathbf{Z}}, \hat{\mathbf{Y}}, C^n)}(\mathbf{Z}, \mathbf{Y}) \tag{26}$$

where $g_{(\hat{\mathbf{Z}}, \hat{\mathbf{Y}}, C^n)}(\mathbf{Z}, \mathbf{Y})$ is computed as

$$C^n - (C^n - LB0)\left(\sum_{(k,i) \in KI'_n : \hat{Z}_{ki}=1} (1 - Z_{ki}) + \sum_{(k,i) \in KI_n} \sum_{l \in N_k : \hat{Y}_{kli}=1} (1 - Y_{kli}) + \sum_{(k,i) \in KI_n} \sum_{l \in N_k : \hat{Y}_{kli}=0} Y_{kli} \right) \tag{27}$$

The proposed Benders framework is given as follows.

Logic-based Benders algorithm

- Initialization. Compute lower bound LB0 and solve problem BMP_R (see Section 3.3) to compute upper bound UB0. Set Set LB=LB0 and UB=UB0.
- While LB<UB
  - (i) Solve problem BMP and let $(\hat{\mathbf{Z}}, \hat{\mathbf{Y}}, \hat{\boldsymbol{\delta}}, \hat{\boldsymbol{\sigma}}, \hat{C})$ be the corresponding optimal solution. Set LB= $\hat{C}$;
  - (ii) Solve problem BSP and let BSP^n be the subproblem having the maximum makespan and let C^n be the corresponding optimal value. Set $\mathrm{UB} = \max\{\mathrm{UB}, C^n\}$.
  - (iii) Add the Benders cut $C \geq g_{(\hat{\mathbf{Z}}, \hat{\mathbf{Y}}, C^n)}(\mathbf{Z}, \mathbf{Y})$.
- The optimal makespan is UB.

In the above procedure, the initial lower bound LB0 is computed by solving the initial master problem, i.e. problem BMP without any Benders cut.

The following lemma holds about Benders cut (26).

Lemma 1. Benders cut (25) satisfies the following properties:

(i) $g_{(\hat{\mathbf{Z}}, \hat{\mathbf{Y}}, C^n)}(\hat{\mathbf{Z}}, \hat{\mathbf{Y}}) = C^n$;

(ii) Any feasible solution $(\bar{\mathbf{Z}}, \bar{\mathbf{Y}}, \bar{\boldsymbol{\delta}}, \bar{\boldsymbol{\sigma}}, \bar{C})$ of formulation QCSP satisfies

$$\bar{C} \geq g_{(\hat{\mathbf{Z}},\hat{\mathbf{Y}},C^n)}(\bar{\mathbf{Z}},\bar{\mathbf{Y}}).$$

Proof. Property (i) holds since the second term at the right-hand-side of expression (27) is equal to 0 when the solution vectors $(\mathbf{Z},\mathbf{Y})$ define the same group assignments and cross-group assignments defined by solution vectors $(\hat{\mathbf{Z}},\hat{\mathbf{Y}})$ for QCs in set K_n and bays in set I_n. Notice that, due to constraint (7a), the term $\sum_{(k,i)\in KI'_n:\ \hat{Z}_{ki}=0} Z_{ki}$ can be eliminated from expression (27). Concerning property (ii), if solution vectors $(\bar{\mathbf{Z}},\bar{\mathbf{Y}})$ define a group assignments and a cross-group assignments for QCs in set K_n and bays in set I_n that differs from the ones defined by solution vectors $(\hat{\mathbf{Z}},\hat{\mathbf{Y}})$, then we have $\sum_{(k,i)\in KI'_n:\hat{Z}_{ki}=1}\left(1-\bar{Z}_{ki}\right)+\sum_{(k,i)\in KI_n}\sum_{l\in N_k:\hat{Y}_{kli}=1}\left(1-\bar{Y}_{kli}\right)+\sum_{(k,i)\in KI_n}\sum_{l\in N_k:\hat{Y}_{kli}=0}\bar{Y}_{kli}\geq 1$ and

$$C^n-(C^n-LB0)\left(\sum_{(k,i)\in KI'_n:\hat{Z}_{ki}=1}\left(1-\bar{Z}_{ki}\right)+\sum_{(k,i)\in KI_n}\sum_{l\in N_k:\hat{Y}_{kli}=1}\left(1-\bar{Y}_{kli}\right)+\sum_{(k,i)\in KI_n}\sum_{l\in N_k:\hat{Y}_{kli}=0}\bar{Y}_{kli}\right)\leq C^n-(C^n-LB0)=LB0$$

and the inequality $\bar{C}\geq g_{(\hat{\mathbf{Z}},\hat{\mathbf{Y}},C^n)}(\bar{\mathbf{Z}},\bar{\mathbf{Y}})$ holds since we have $\bar{C}\geq LB0$ from the definition of $\bar{C}$ and LB0, upper and lower bounds on the optimal makespan, respectively. □

The following theorem shows the correctness of the algorithm described above.

Theorem 1. The logic-based Benders algorithm terminates with an optimal solution after finitely many steps.

Proof. Suppose first that the algorithm terminates with a finite solution UB. Clearly, UB is an upper bound on the optimal makespan. Because the algorithm terminated, we have LB=UB and due to Lemma 1, property (ii), LB is a valid lower bound on the optimal makespan. Because the solution corresponding to UB is feasible, it is also optimal. Secondly, since the domain of the master variables $\mathbf{Z}$ and $\mathbf{Y}$ is finite, only finite many subproblems can be defined (and corresponding Benders cuts), and the optimal value is reached after finitely many steps. □

4 Computational results

In this section, we evaluate the computational efficiency of the logic-based Benders algorithm (hereafter called "EXM") described in Section 3.4. In the experiments, we have used the general purpose integer programming solver provided by the IBM ILOG CPLEX 12.6.1 callable library (IBM CPLEX, 2016). All the computational experiments were performed on a computer with Intel Core i7-6700 3.40 GHz CPU equipped with 16 GB of RAM.

To test the effectiveness of our approach, we considered the benchmark instances and the Java-implementation QCSPgen of generation schemes proposed by Meisel and Bierwirth (2010). Meisel and Bierwirth described a new platform for the comparison of models and algorithms with application to quay crane scheduling. They provided a scalable, reproducible, and unbiased test environment, which reflects many important features of crane scheduling that are met in practice as well as in research. In particular, we considered the following four sets of instances generated by Meisel and Bierwirth (2010):

- Set B composed of 60 instances based on medium sized vessels with 15 bays and a corresponding bay capacity of 400 containers that are served by four cranes;
- Set C composed of 60 instances based on large-sized vessels with 20 bays and a corresponding bay capacity of 600 containers that are served by six cranes;
- Set D composed of 60 instances based on medium-sized vessels with 15 bays that are served by four cranes. For these instances, the handling rate is equal to 0.2 or 0.8 and the location parameter follows the Gaussian distribution or the uniform distribution;
- Set F composed of 50 instances based on medium sized vessels with 15 bays that are alternatively served by two to six cranes from which we selected 30 instances that are used to analyses the quality of our lower bound LB0 and upper bound UB0 (see the initialization phase of EXM).

Meisel and Bierwirth generated instances that also consider precedence among tasks, which are not considered in our problem. For further details about the instances, the reader is referred to the work of Meisel and Bierwirth (2010).

The performance of EXM has been compared with the exact method obtained by solving

directly (using the CPLEX solver) the mathematical formulation proposed by Liu et al. (2006) (hereafter called "LIU"), that is, to the best of our knowledge, the best exact method proposed so far to solve the QCSP considered in this paper.

Table 1. Lower bound LB0 and upper bound UB0 on instances of Set F (|*I*|=15).

	|*K*| = 2						|*K*| = 4						|*K*| = 6					
Instance	LIU	Time	%LB0	Time	%UB0	Time	LIU	Time	%LB0	Time	%UB0	Time	LIU	Time	%LB0	Time	%UB0	Time
1	1510.0	0.3	100.0	0.1	108.0	0.1	767.0	0.4	99.3	0.3	109.9	0.1	535.0	1.3	99.4	0.4	111.6	0.2
2	1510.0	0.5	100.0	0.2	108.0	0.1	774.0	0.5	98.6	0.2	116.4	0.2	542.0	1.2	100.0	0.5	122.0	0.2
3	1512.0	0.3	100.0	0.1	101.4	0.1	774.0	0.5	100.0	0.4	105.4	0.2	521.0	0.9	100.0	0.4	118.2	0.1
4	1510.0	0.3	100.0	0.2	100.8	0.1	786.0	0.4	97.3	0.3	102.3	0.2	568.0	0.8	99.8	0.3	109.0	0.2
5	1514.0	0.3	100.0	0.2	108.2	0.1	776.0	0.6	99.7	0.4	107.3	0.1	541.0	1.1	100.0	0.3	118.3	0.2
6	1510.0	0.3	100.0	0.2	100.0	0.1	795.0	0.4	96.6	0.3	109.1	0.2	515.0	1.1	100.0	0.3	124.7	0.2
7	1510.0	0.3	100.0	0.2	102.3	0.1	770.0	0.4	99.4	0.4	104.0	0.2	585.0	0.8	96.2	0.3	110.1	0.2
8	1509.0	0.3	100.0	0.2	101.7	0.1	781.0	0.5	99.6	0.2	102.7	0.1	562.0	0.8	100.0	0.5	113.5	0.1
9	1512.0	0.3	100.0	0.1	102.9	0.1	770.0	0.3	100.0	0.2	112.1	0.1	576.0	0.9	100.0	0.2	103.0	0.1
10	1511.0	0.4	100.0	0.1	100.1	0.1	789.0	0.5	99.0	0.3	105.6	0.2	562.0	0.8	100.0	0.2	114.2	0.1
Average			100.0		103.34				98.94		108.13				99.54		115.49	

There are mainly two factors that affect the performance of the proposed Benders approach: (i) the solution of the Benders master problem, i.e., the quality of the corresponding lower bound as well as its solution time and (ii) the effectiveness of the logic Benders cut.

We first analyses the quality of lower bound LB0 and upper bound UB0 corresponding to the optimal solution costs of the initial master problem BMP and of problem BMP_R, respectively, obtained during the initialization phase of EXM. Table 1 compares LB0 and UB0 with the optimal solution cost obtained by method LIU on Set F of instances. For each instance and for each value of |*K*|, the table reports the optimal makespan *z* (column "LIU"), the percentage deviation of lower bound LB0 ("%LB0") computed as 100*LB0/*z*, the percentage deviation of upper bound UB0 ("%UB0") computed as 100*UB0/*z* and the corresponding computing times in seconds (columns "Time").

Table 1 shows that the instances of Set F can be easily solved to optimality by method LIU and that lower bound LB0 provided by the initial BMP is tight, as shown by the average percentage deviations equal to 100.00, 98.94 and 99.54 for number of QCs equal to 2, 4 and 6,

respectively. Upper bound UB0 is not so tight, especially for increasing values of |*K*|. Nonetheless, for large-sized instances, its computation let us quickly compute an initial upper bound for EXM.

Table 2. Detailed results of EXM on instances of set F with |*K*| = 4.

Instance	|*K*| = 2					|*K*| = 4					|*K*| = 6				
	EXM time	BMP time	BSP time	Iter	Cuts	EXM time	BMP time	BSP time	Iter	Cuts	EXM time	BMP time	BSP time	Iter	Cuts
1	0.21	0.10	0.02	1	0	2.55	2.31	0.08	6	5	1.53	1.33	0.07	3	2
2	0.29	0.18	0.02	1	0	1.41	1.26	0.06	3	2	0.63	0.50	0.02	1	0
3	0.23	0.12	0.01	1	0	1.03	0.73	0.04	2	1	0.57	0.44	0.01	1	0
4	0.30	0.23	0.02	1	0	1.88	1.69	0.10	6	5	1.02	0.81	0.03	2	1
5	0.26	0.16	0.01	1	0	0.67	0.44	0.05	2	1	0.46	0.34	0.02	1	0
6	0.32	0.24	0.02	1	0	1.33	1.16	0.07	4	3	0.39	0.29	0.02	1	0
7	0.21	0.19	0.01	1	0	1.76	1.33	0.09	3	2	3.38	3.32	0.06	8	7
8	0.32	0.23	0.01	1	0	1.38	1.07	0.08	4	3	0.67	0.52	0.02	1	0
9	0.19	0.10	0.01	1	0	0.25	0.21	0.01	1	0	0.28	0.16	0.02	1	0
10	0.19	0.10	0.01	1	0	1.88	1.40	0.06	3	2	0.65	0.53	0.03	4	3
Average	0.25	0.17	0.01	1.0	0.0	1.41	1.16	0.06	3.8	2.8	0.98	0.82	0.03	2.5	1.5

All the instances of class F can be easily solved to optimality by EXM with an average number of iterations equal to 1.0, 3.8 and 2.2 for |*K*| = 2, 4, and 6, respectively. In order to have some insights about EXM, Table 2 shows the details of the execution of EXM for instance set F for the case |*K*| = 4. The table reports the total computing time in seconds spent by EXM ("EXM time"), the total computing time spent during the solution of the master problem ("BMP time"), the total computing time spent during the solution of the subproblem ("BMP time"), the total number of iterations ("Iter") and the number of Benders cut added ("Cuts").

Table 3 gives the results about methods LIU and EXM on the set of instances B and C. The instances have been grouped into six groups of 10 instances each. Under column heading "Time" the table reports the average computing times obtained by the two methods. All the instances of sets B and C can be easily solved to optimality by both methods LIU and EXM. The table also displays the average percentage deviation of lower bound LB0 and the average number of iterations of EXM.

Table 4 gives the results about methods LIU and EXM on the instance set D. Two parameters are used to generate the instances of set D, (i) the handling rate "*f*", which denotes the percentage of containers handled in a service with respect to the total vessel capacity, and (ii) the location parameter "*loc*", which is responsible for the distribution of container groups over the set of bays. It can be seen from Table 4 that the handling rate *f* influences the computing time of method LIU, and that method EXM performs better if *loc* follows the Gaussian distribution (i.e., *loc* is set to "*cl* 1" or "*cl* 2") compared with uniform distribution (i.e., *loc* is set to "*uni*").

Table 3. Results about EXM and LIU on sets B (|*K*|=4, |*I*|=15) and C (|*K*|=6, |*I*|=20).

Set B	LIU	EXM			Set C	LIU	EXM		
Group	Time	Time	%LB0	Iter	Group	Time	Time	%LB0	Iter
1	0.47	1.13	99.39	3.7	1	5.34	2.63	98.02	2.6
2	0.41	1.41	98.94	3.8	2	5.91	1.89	98.18	2.2
3	0.57	1.27	98.69	3.5	3	6.16	2.53	97.28	2.0
4	0.53	1.58	99.60	4.1	4	8.65	4.24	97.81	3.1
5	0.51	1.30	98.75	4.0	5	9.26	3.08	98.26	3.5
6	0.46	1.15	99.31	3.0	6	9.60	2.83	97.59	3.6
Average	0.49	1.32	99.11	3.68	Average	8.98	2.85	97.86	2.83

Table 4. Results about EXM and LIU on set D (|*K*|=4, |*I*|=15)

Set D	LIU	EXM		
(*f*, *loc*)	Time	Time	%LB0	Iter
(0.2, *uni*)	0.47	1.27	96.98	4.1
(0.8, *uni*)	0.98	1.47	99.38	2.7
(0.2, *cl* 1)	0.46	1.06	96.11	3.0
(0.8, *cl* 1)	0.93	0.68	99.92	2.2
(0.2, *cl* 2)	0.49	0.50	99.37	1.5
(0.8, *cl* 2)	0.87	0.62	99.81	1.7

We also used instance generator QCSPgen developed by Meisel and Bierwirth (2010) to generate a new set of 90 large-sized instances. The workload of the bays in each instance is set to be equal to the workload of the |*K*| first tasks in the default instance group of QCSPgen (i.e., the number of tasks and bays are 50 and 15, respectively). The number of QCs of these

instances has been defined by following the schemes proposed by Lalla-Ruiz (2014) and Giallombardo et al (2010) as a function of the size of the vessel and of the workload. We generated instances with |*K*| ranging from 4 to 6 and |*I*| ranging from 25 to 35, 10 instances for a given pair of |*K*| and |*I*|. The average results obtained are reported in Table 5 where each rows is relative to the 10 instances generated for each the pair of values |*K*| and |*I*|.

Table 5. Results about EXM and LIU on large-sized instances based on QCSPgen.

	LIU	EXM		
|*I*|, |*K*|	Time	Time	%LB0	Iter
25, 4	6.77	2.59	99.93	1.6
25, 5	9.45	6.35	99.36	2.5
25, 6	10.81	8.05	97.78	1.8
30, 4	23.09	4.39	100.00	1.0
30, 5	37.19	17.98	99.37	1.4
30, 6	41.52	24.39	97.20	1.6
35, 4	88.71	11.96	100.00	1.2
35, 5	189.95	87.00	99.22	1.4
35, 6	273.18	230.29	98.12	1.3

Table 5 reports the results obtained using the same notation used for Table 4. All the instances 90 instances have been also solved to optimality by both methods. Tables 3 and 5 show that when the number of bays and QCs increases, method EXM outperforms method LIU. This efficiency is mainly due to the modeling technique we adopted which decomposes QCSP into a series of easier problems, and the logic Benders cuts which ensure quick convergence to optimality.

Since all the instances of sets B, C, D, F and the instances generated with QCSPgen can be solved to optimality by both LIU and EXM, to further test the effectiveness of EXM, we generated an additional set of randomly generated instances following the scheme proposed by Kim and Park (2004).

We generated 90 randomly instances divided into 9 groups, each of which contains 10 instances. The number of bays is 25, 30 or 35 while the number of cranes is 4, 6 or 8. The number of containers in each bay is randomly generated using a uniform distribution in the

interval [3, 180]. For these instances, we imposed a time limit of 1800 second to the execution of methods LIU and EXM.

Table 6 gives the results obtained on the set of randomly generated instances. For each method, the table reports the average computing time in seconds (column "Time") and the average optimality gap ("Gap"). The table also reports, for each group of instances, the number of instances solved to optimality ("Opt"). For each method, the average computing time is computed over the set of instances solved to optimality by the method.

Table 6. Results about EXM and LIU on large-sized instances based on Kim and Park (2004).

	LIU			EXM				
$\|I\|, \|K\|$	Time	Gap	Opt	Time	Gap	Opt	%LB0	Iter
25, 4	7.97	0	10	3.28	0	10	99.92	1.2
25, 5	35.60	0	10	11.59	0	10	99.03	1.5
25, 6	42.26	0	10	27.63	0	10	97.78	1.5
30, 4	25.21	0	10	7.60	0	10	100.00	1.0
30, 5	217.76	0	10	41.94	0	10	100.00	1.2
30, 6	312.89	0	10	129.10	0	10	99.56	2.1
35, 4	185.32	0	10	20.11	0	10	99.87	1.1
35, 5	834.40	0	10	215.51	0	10	98.65	1.3
35, 6	-	1.04	0	1186.03	2.65	6	99.25	1.5

Table 6 shows that the instances based on the Kim and Park (2004) scheme are more difficult than the instances of same dimension based on QCSPgen. A possible explanation for this behaviour is due to the fact that the range of values of the workload of the instances based on QCSPgen is larger than that of the instances based on Kim and Park. As a consequence, the mixed-integer programs being solved display many symmetries, thus slowing down the entire solution process of LIU and EXM. The table shows that EXM is superior to LIU, both in term of the average computing time and the number of instances solved to optimality. Instances with $|I|$=35 and $|K|$=6 are particularly difficult for both methods – indeed none of the 10 instances with $|I|$=35 and $|K|$=6 have been solved to optimality by LIU (within the imposed time limit) whereas 6 out of 10 instances were solved to optimality by EXM.

5 Conclusions and future research

This paper investigated the Quay Crane Scheduling Problem (QCSP) with QCs' moving times. We described a new modeling strategy to address the assignment structure of QCSP, and also developed an innovative and efficient Benders decomposition-based framework to solve QCSP based on based on logic-based Benders decomposition. Proofs have been provided to prove that the proposed framework convergences to an optimal solution.

We performed a computational study on both medium and large-sized QCSP instances involving up to 35 bays and 10 cranes, and generated using generation schemes proposed in the literature. The results show that the proposed Benders framework outperforms state-of-the-art method for QCSP on large-sized QCSP instances involving up to 35 bays and six cranes.

Future study will focus on the integrated problem of berth allocation and quay crane assignment and scheduling, because how berths are allocated affects how cranes are optimally utilized, and vice versa.

Acknowledgement

This research is supported by the Fund for Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 71321001), and the National Base of Intelligence Introduction for Innovation (‘111’ Project, Grant No. B16009).

The authors would like to thank the anonymous reviewers for their extremely helpful suggestions and very thorough review of the paper.

References

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

Benders, J.F. (1962) Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1), 238–252.

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

Codato G, Fischetti M. (2006) Combinatorial Benders'cuts for mixed-integer linear programming. *Oper Res*, 54(4):756–66.

Daganzo, C.F. (1989) The crane scheduling problem. *Transportation Research B*, 23(3), 159–175.

Geoffrion, A. M. (1972) Generalised Benders Decomposition. *Journal of Optimization Theory and Application*, 10, 237–260

IBM CPLEX (2016). IBM ILOG CPLEX 12.6.1 callable library.

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

Lim, A., Rodrigues, B., Xiao, F. and Zhu, Y. (2004) Crane scheduling with spatial constraints. *Naval Research Logistics*, 51(3), 386–406.

Lim, A., Rodrigues, B. and Xu, Z. (2007) A m-parallel crane scheduling problem with a non-crossing constraint. *Naval Research Logistics*, 54, 115–127.

Liu, J.Y., Wan Y.W. and Wang, L. (2006) Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. *Naval Research Logistics*, 53(1), 60–74.

Giallombardo, G., Moccia, L., Salani, M. and Vacca, I. (2010) Modeling and solving the tactical berth allocation problem. *Transportation Research Part B*, 44(2): 232-245.

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

Hooker J. (2000) Logic-based methods for optimization: combining optimization and constraint satisfaction. New York: John Wiley & Sons.

Hooker, J.N., G. Ottosson. (2003) Logic-based Benders decomposition. *Mathematical Programming*, 96: 33–60.

Hooker J. (2007). Planning and scheduling by logic-based Benders decomposition. *Oper Res* 55(3):588–602.

Imai, A., Nishimura, E., and Papadimitriou, S. (2001) The dynamic berth allocation problem for a container port. *Transportation Research B*, 35(4), 401–417.

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

Lalla-Ruiz, E. and Moreno-Vega, J.M. (2014) . Biased random key genetic algorithm for the tactical berth allocation problem. *Applied Soft Computing*, 2014, 22(6): 60-76

Meisel, F. and Bierwirth, C. (2010) A unified approach for the evaluation of quay crane scheduling models and algorithms. *Computers & Operations Research*, 38, 683–693.

Moccia, L., Cordeau, J., Gaudioso, M., and Laporte, G. (2006). A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. *Naval Research Logistics*, 53, 45–59.

Ng, W.C. and Mak, K.L. (2005) Yard crane scheduling in port container terminals. *Applied Mathematical Modelling*, 29, 263–276.

Peterkofsky, R. I. and Daganzo, C.F. (1990) A branch and bound solution method for the crane scheduling problem. *Transportation Research B*, 24(3), 159–172.

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.

Sampaio, A., Urrutia, S. and Oppen, J. (2016) A decomposition approach to solve the quay crane scheduling problem. Available online.

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

Tang, L.X., Jiang, W., Liu, J.Y. and Dong, Y. (2015) Research into container reshuffling and stacking problems in container terminal yards. *IIE Transactions*, 47(7), 751–766.

Tang, L.X., Zhao, J. and Liu, J.Y. (2014) Modeling and solution of the joint quay crane and truck scheduling problem. *European Journal of Operational Research*, 236, 978–990.

Tavakkoli-Moghaddam R., Makui A., Salahi S., Bazzazi M. and Taheri F. (2009) An efficient algorithm for solving a new mathematical model for a quay crane scheduling problem in container ports. *Computers & Industrial Engineering*, 56(1), 241-248.

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

Zhang, C.Q., Liu, J.Y., Wan, Y.-w., Murty, K.G. and Linn, R.J. (2003) Storage space allocation in container terminals. *Transportation Research B*, 37, 883–903.

Zhu, Y. and Lim, A. (2006) Crane scheduling with non-crossing constraint. *Journal of the Operational Research Society*, 57(12), 1464–1471.