Transportation Research Part E 81 (2015) 52–74

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

A polynomial-time heuristic for the quay crane double-cycling problem with internal-reshuffling operations

Ming Liu [a], Feng Chu [b], Zizhen Zhang [c,*], Chengbin Chu [a,d]

[a] School of Economics & Management, Tongji University, Shanghai, PR China
[b] Laboratory IBISC, University of Evry-Val dÉssonne, Evry 91020, France
[c] School of Mobile Information Engineering, Sun Yat-sen University, Gangzhou 510275, PR China
[d] Laboratoire Génie Industriel, Ecole Centrale Paris, Grande Voie des Vignes, 92295 Châtenay-Malabry Cedex, France

ARTICLE INFO

Article history:
Received 19 November 2014
Received in revised form 24 June 2015
Accepted 25 June 2015
Available online 10 July 2015

Keywords:
Container port
Terminal operations
Double cycling
Internal reshuffling

ABSTRACT

One of great challenges in seaport management is how to handle containers under reshuffling, called reshuffles. Repositioning reshuffles in a bay (internal reshuffling) can improve the efficiency of quay cranes and help ports to reduce ship turn-around time. This paper studies the quay crane double-cycling problem with internal-reshuffling operations, and presents a fast solution algorithm. To reduce the number of operations necessary to turn around a bay of a vessel, the problem is first formulated as a new integer program. A polynomial-time heuristic is then developed. The analysis is made on the worst-case error bound of the proposed algorithm. Results are presented for a suite of combinations of problem instances with different bay sizes and workload scenarios. Comparisons are made between our algorithm and the start-of-the-art heuristic. The computational results demonstrate that our model can be solved more efficiently with CPLEX than the model proposed by Meisel and Wichmann (2010), and the proposed algorithm can well solve real-world problem instances within several seconds.

1. Introduction

Efficient seaports could help to lower transport costs by enabling cargos to get to and from markets in a more timely and cost-effective fashion. Nowadays, more and more goods in global trade are containerized and transported by container vessels. High utilization of container terminal resources are key in accelerating maritime logistics and lower operating costs. Quay cranes (QCs) are the most expensive single unit of handling equipment in container ports. One of the key operational bottlenecks at ports is QC availability (Crainic and Kim, 2005). To reduce ship turn-around time, ports make continuous improvement on the efficiency of QCs. As QC efficiency is the key bottleneck to port productivity, the work presented in this paper addresses such an operational-research problem. In contrast to terminal expansion or information technology deployments, the internal-reshuffling method, considered in this paper, is low-cost. The internal-reshuffling technique can be quickly implemented, and can be used to complement the classic double-cycling technique (i.e., loading ships as they are unloaded).

The layout of containers on a ship can be viewed as a box. Containers are stacked on top of one another and arranged in rows, also called bays. Fig. 1 illustrates the plan, side and front views of a container vessel: in the plan view, there are seven

* Corresponding author.
E-mail address: zhangzizhen@gmail.com (Z. Zhang).

http://dx.doi.org/10.1016/j.tre.2015.06.009

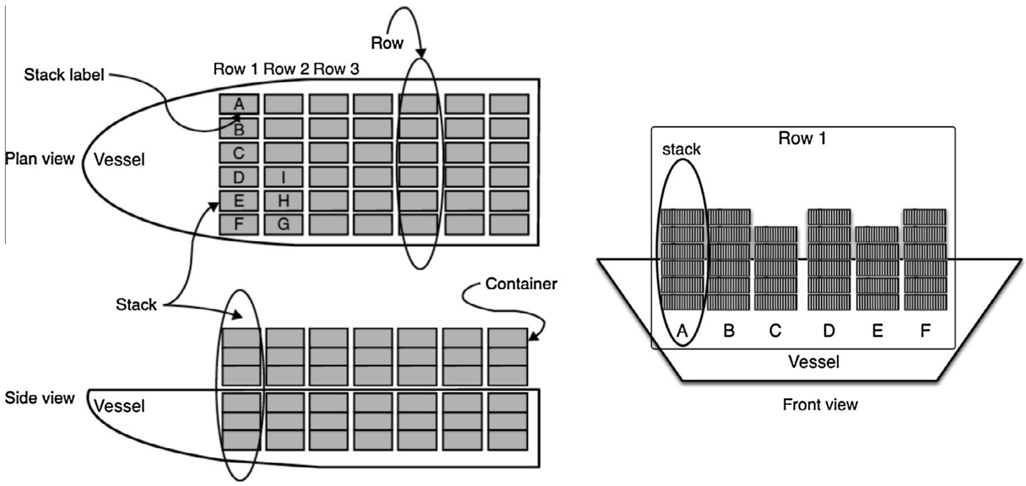

Fig. 1. Three views of a vessel (Goodchild and Daganzo, 2006).

rows where the first row consists of six container stacks; in the side view, a stack holds six containers; in the front view, six container stacks are listed from left to right on the vessel. (In Fig. 1, the number of containers are not representative of typical ship size.) Note that containers can be accessed only from above by QCs, and such a stacking manner induces the precedence constraints among containers of each stack, also called the *stack dependent accessibility*.

Considering that QCs move slowly, only after completing all operations in one bay a crane operator may drive it to the next one, as is current practice. Traditionally, the unloading and loading processes of a vessel are separated. In one cycle of quay crane's trolley, it unloads (or loads) one container and return without container, referred as single cycling (see Fig. 2a). To improve QC productivity, double-cycling technique is first proposed in a pioneering work (Goodchild, 2005). The technique consists of converting empty crane moves into productive ones. With double cycling (see Fig. 2b), containers are loaded and unloaded in the same crane cycle, i.e., a complete round-trip of the crane trolley from the ship to the dock and back. This allows a QC to double the number of containers transported in one cycle (Goodchild and Daganzo, 2007). Nowadays, the double-cycling technique has been widely adopted and implemented, e.g., at train terminals (Goodchild et al., 2011). Although double cycling has been used to some extent in practice, port operators ask for more QC efficiency to counter port congestion which has been hitting the headlines recently (Drewry Container Insight, 2014).

To improve QC efficiency, internal reshuffling, a complementing technique to the double-cycling technique, can be used to improve the efficiency of QCs by replacing couples of unloading and loading operations for reshuffles by repositioning operations (Meisel and Wichmann, 2010). Instead of using the current method (called *external reshuffling*), where often all reshuffles are unloaded from the vessel and then reloaded from the dock, some reshuffles can be repositioned directly in a bay on a ship. This allows the QC to replace two operations (i.e., one unloading and one loading operation) by one operation (i.e., a repositioning), to complete a reshuffling requirement. Thus, this technique reduces the number of operations, and ensures high utilization of QCs. In the remainder, for easy recognition where internal-reshuffling method is applied, we also refer to a repositioning operation induced by internal-reshuffling method as an *internal-reshuffling operation*. We assume the

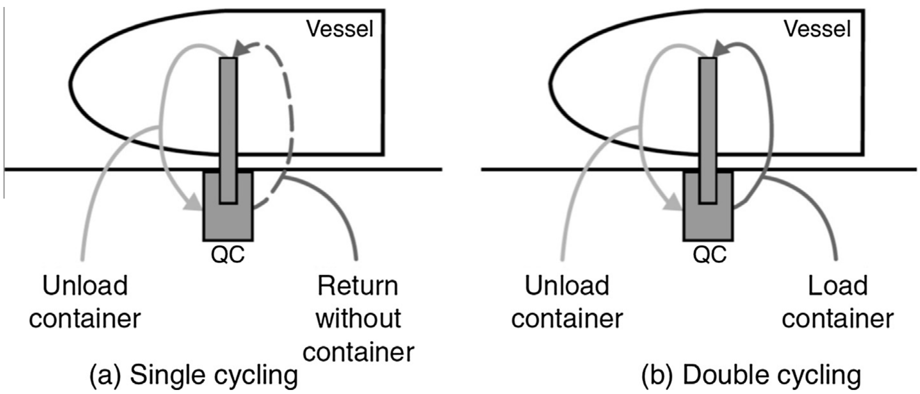

Fig. 2. Single cycling and double cycling (Goodchild and Daganzo, 2006).

ship's arrival stowage plan, also called *arrival plan*, and departure stowage plan, also called *departure plan*, are given, which indicate the positions, also called slots, of the containers to be unloaded, loaded and reshuffled. In current practice, shipping lines use software tools to create such plans that accommodate vessel stability requirements, priority of delivery, placement constraints on hazardous materials, etc. (Goodchild and Daganzo, 2006). We therefore consider improvements only on the crane's sequence of operations, to complete the conversion from the arrival plan to the departure plan. Note that not all reshuffles can be handled internally. Only if a slot to be hold by a reshuffle in the departure plan is empty, may an internal-reshuffling operation happen.

In the classic quay crane double-cycling problem (QCDCP), decisions are made on the basis of container stacks. That is, a solution consists of a sequence of container stacks. In our problem, the quay crane double-cycling problem with internal-reshuffling operations (QCDCP-IR), a solution must be made on a container basis (i.e., it consists of a sequence of container operations). The QCDCP-IR is more complicated and requires more sophisticated solution algorithms. Meisel and Wichmann (2010) initiate the study of the QCDCP-IR, and propose an integer programming formulation and a heuristic. In their work, the advantage of the internal-reshuffling method in improving QC productivity has been shown, compared with the sole application of the double-cycling technique. Their experimental results demonstrate that about 64% of the reshuffles in a bay can be reshuffled internally by their solution approach, and 32% operations related to reshuffles can be saved by the internal-reshuffling method. However, their mathematical formulation is relatively rough, and their heuristic approach has no guaranteed error bounds and requires a relatively large computational time, especially for practical-size problems. Thus, a fast solution approach with acceptable worst-case error bounds should further improve QC efficiency for the QCDCP-IR. We revisit the QCDCP-IR, aiming to provide some analytical results and an easy-to-implement solution approach. The contribution includes

- We formulate a new integer program based on new several observations. Comparisons are made between the existing model and our new formulation, both solved with CPLEX. Experimental results show that our model is so-called *CPLEX-effective*, which means (i) delivering the same quality solutions, our formulation saves computational time and (ii) consuming the same running time, our formulation delivers higher quality solutions.
- To address the general QCDCP-IR, especially for large-size problems, we devise a fast heuristic, and analyze its worst-case error bound. Computational experiments based on 1000 instances have been conducted. Compared with the state-of-the-art heuristic, results show that our solution approach is very efficient in terms of running time and it is also relatively robust.

The paper is organized as follows. A brief literature review is given in Section 2. In Section 3, we restate the QCDCP-IR. In Section 4, we provide a new integer programming formulation based on several observations. In Section 5, for the general case, we devise a polynomial-time heuristic and analyze its worst-case error bound. In Section 6, computational experiments are conducted. The performance of our model and the existing one are first compared. Then we evaluate the solution quality of the proposed algorithm by comparison with the state-of-the-art heuristic, to demonstrate the efficiency of our approach. Section 7 concludes the paper and give directions of further work. In Appendix, we correct errors in Meisel and Wichmann (2010)'s formulation, and identify a polynomially solvable case for the QCDCP-IR.

2. Literature review

In the literature, a comprehensive overview on applications and optimization models for container port optimization can be found in Stahlbock and Voß (2008), Bierwirth and Meisel (2010), and Carlo et al. (2013). Other study streams of maritime logistics may include liner shipping network management (Meng and Wang, 2011; Wang, 2014 and Wang et al., 2014), container assignment and routing (Bell et al., 2011; Bell et al., 2013 and Wang et al., 2015), and vessel bunker and speed management (Wang and Meng, 2012 and Wang et al., 2013). Recent advances on liner shipping and port operations planning can be found in Meng et al. (2014) and Bierwirth and Meisel (2015), respectively. As our concern falls in the scope of the quay crane scheduling at seaports, we review the most pertinent literature.

Daganzo (1989) analyzes the performance of different quay crane scheduling algorithms, with the objective to maximize the throughput. Kim and Park (2004) define the tasks on the basis of container groups, which are subsets of containers on a vessel with identical destinations or other attributes such as weight class. An exact algorithm and a greedy randomized adaptive search procedure are proposed. Lim et al. (2007) analyze a relatively simple model with a major feature of the non-crossing restriction. They prove the existence of an optimal schedule which is unidirectional if the bay-to-crane assignment is given. This rule has been widely adopted especially for the development of heuristics. Bierwirth and Meisel (2009) revise Kim and Park (2004)'s model and propose a heuristic based on a branch-and-bound framework. Choo et al. (2010) investigate a multi-ship quay crane scheduling problem with yard congestion constraint. They develop a Lagrangian relaxation-based heuristic. Recently, Liu et al. (2014) design a 4/3-approximation and 5/3-approximation algorithms for the quay crane scheduling problems with two and three QCs, respectively.

Goodchild and Daganzo (2006) initiate the study of the QCDCP. They formulate the unloading and loading operations of each container stack as a two-machine flow shop job with two sequential processing times. They show that when there are no hatch covers, the problem can be mapped into a two machine flow shop scheduling problem, and be solved optimally with Johnson's rule. They further study a general QCDCP in which hatch covers are involved, and propose a decomposition heuristic. Containers under reshuffling are handled externally in this work. For the general QCDCP, Zhang and Kim (2009) develop a local-search based heuristic. Numerical experiments show their solution approach is very efficient. Recently, Lee et al. (2014) have developed an optimal algorithm for the general QCDCP, which runs in polynomial time.

Meisel and Wichmann (2010) initiate the study of the QCDCP-IR, where internal-reshuffling operations are enabled. The problem is investigated on a container basis. An integer programming formulation and a heuristic approach are provided. They also demonstrate that the consideration of the internal-reshuffling operations leads to shortening of the vessel handling time compared to the sole application of the double-cycling technique.

3. Problem description

In this section, we restate the QCDCP-IR, which involves a single bay and a single quay crane. In one crane cycle, the QC is enabled to perform at most one unloading and one loading operations. Given the arrival and departure plans, the aim is to determine a feasible sequence of container operations that converts the arrival plan to the departure plan with minimum service time. A feasible solution must respect the stacking dependent accessibility. In other words, the precedence constraints induced by the container stacking manner cannot be violated.

Containers are categorized into four classes: import containers, export containers, fixed containers, and reshuffles (i.e., containers to be reshuffled). *Import containers* are those to be unloaded from the vessel to the dock, whereas *export containers* are those to be loaded from the dock to the vessel. Clearly, import (export) containers appear only in the arrival (departure) plan. There are also some containers on the ship not destined to the current terminal, which can be classified as (i) *reshuffles* which stay on top of the import containers, must be temporarily removed and then restored, as they block the unloading operations of the import containers and (ii) *fixed containers* which are to stay on the vessel involving no operations, and thus are not considered in the following analysis. Notice that the same reshuffles can be found in both plans.

From the perspective of terminal managers, reshuffles belong to the same class and thus they can exchange positions in the departure plan (Meisel and Wichmann, 2010). That is, each reshuffle in the arrival plan can be placed at any slot to be hold by a reshuffle in the departure plan. Reshuffles can be handled externally or internally. Apparently, an internal-reshuffling operation is preferable to an unloading operation plus a loading operation, as the former requires only one container operation, whereas the latter needs two. However, only when a slot to be hold by a reshuffle in the departure plan (thus it is ready to receive a reshuffle) is available, may an internal-reshuffling operation occur.

Fig. 3 illustrates an example. As there is a doubt how a change of the sequence of operations brings benefits to reduce ship turn-around time, we compare a solution without internal-reshuffling operations and a solution where internal-reshuffling technique is implemented. In the former solution, the sequence must consist of 15 operations, i.e., nine unloading and six loading operations, where reshuffles are handled externally. Fig. 4 illustrates the latter solution with internal-reshuffling operations. This solution comprises six unloading, five loading and two internal-reshuffling operations, thus 13 operations in total. For the latter solution, a step-by-step illustration of service time will be shown in Fig. 10. Note that at steps 5 and 10, internal-reshuffling operations happen. Compared with the former solution of 15 operations, the internal-reshuffling method helps to eliminate two container operations, completing the operations necessary to turn around a bay of a ship.

3.1. Assumptions

For the QCDCP-IR, the following assumptions are made (Meisel and Wichmann, 2010).

A1. Each of import and export containers is processed by one operation. For example, this prevents the repositioning of an import container to another slot in the bay before it is finally unloaded. A reshuffle handled internally needs one operation, whereas a reshuffle handled externally requires two operations.

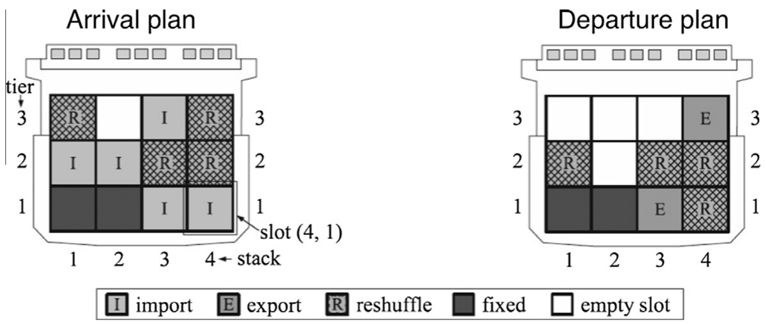

Fig. 3. The arrival and departure plans (Meisel and Wichmann, 2010).

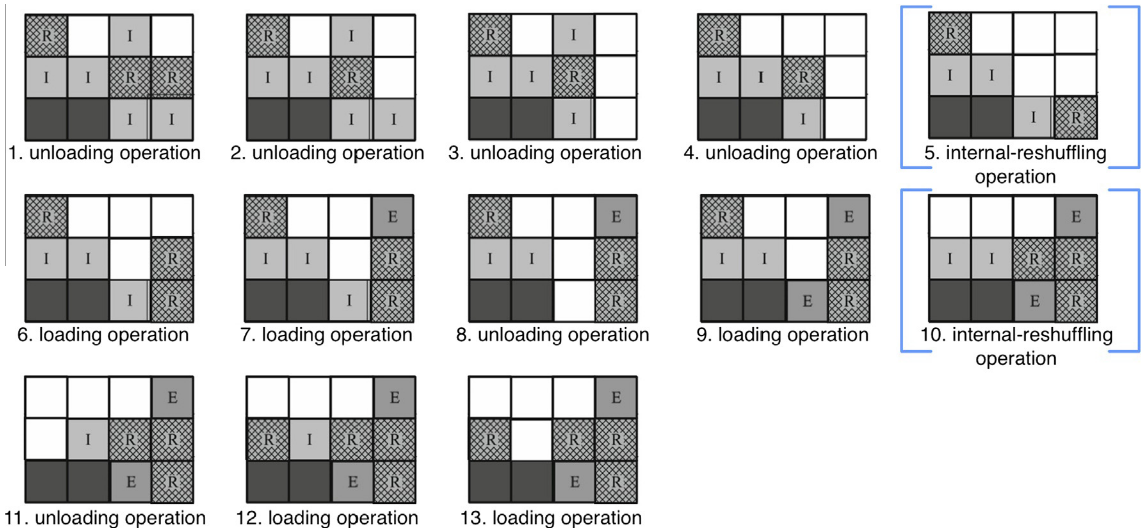

Fig. 4. A sequence of container operations, including six unloading, five loading and two internal-reshuffling operations.

A2. Reshuffles are exchangeable, i.e., every reshuffle in the arrival plan can be positioned at any slot to be hold by a reshuffle in the departure plan. The number of reshuffles in one stack is the same in both plans. This ensures that after the unloading operations of all the containers in a stack, a sufficient number of reshuffles is available for the subsequently loading operations for the stack.
A3. The stability issue of the vessel is not considered. The arrival and departure plans are preset for the planning of quay crane operations.
A4. A hatch-coverless container vessel is considered, where containers below and above deck are not separated by hatch covers.
A5. Horizontal transport vehicles are always available at the dock, waiting to receive unloaded containers or provide containers to be loaded. Hence, the service time of the bay only depends on the sequence of container operations.
A6. The quay crane starts processing of containers from the vessel. In fact, no matter where the quay crane starts, at the vessel or dock, it completes the same work.

Remark 1. Assumption A2 is relatively simplistic, because in reality a reshuffle may be moved from one bay to another. This happens, for instance, to consolidate containers with the same destination, which were not consolidated in the same bay due to previous port operations. This assumption is to make the QCDCP-IR restricted in a single bay, as the classic QCDCP.

3.2. Notation

Let m denote the number of container stacks, and n the total number of containers in both plans.

Five types of jobs (corresponding to container operations) are identified: (i) VY: unload an import container from the vessel to the yard; (ii) YV: load an export container from the yard to the vessel; (iii) VB: unload a reshuffle from the vessel to the buffer on the yard; (iv) BV: reload a reshuffle from the buffer to the vessel; and (v) VV: reposition a reshuffle from one slot on the vessel to another slot on the vessel. With these notation, unloading operations consist of VY and VB jobs, whereas loading operations consist of BV and YV jobs. Internal-reshuffling operations correspond to VV jobs. Let $\mathcal{T}$ denote the set of all the job types, i.e., $\mathcal{T}$ = {VY,YV,VB,BV,VV}. Let p^t denote the processing time of the t-type job, where $t \in \mathcal{T}$, and s^{tu} the setup time (corresponding to the required time of empty trolley move) between the t-type job and the u-type job $(t, u \in \mathcal{T})$, such that the latter immediately follows the former in a sequence. (In the scheduling literature, setup time means a time period required by the machine to be ready to process a job.)

VV job has a less processing time than those of other types, i.e., $p^{VV} = \min_{t \in \mathcal{T}} p^t$, because it starts and ends on the vessel, whereas other jobs travel between the vessel and the dock. The following Tables 1 and 2 are presented in Meisel and Wichmann (2010), where the time unit is second. In Table 1, a VV job has a processing time of 90 s whereas each of the others has a processing time of 100 s. In Table 2, if the two jobs are connected on the vessel or the dock, then the setup time in between requires 10 s, otherwise 20 s.

Table 1
Processing time p^t.

t	p^t
VY	100
YV	100
VB	100
BV	100
VV	90

Table 2
Setup time s^{tu}.

$t \downarrow \backslash u \rightarrow$	VY	YV	VB	BV	VV
VY	20	10	20	10	20
YV	10	20	10	20	10
VB	20	10	20	10	20
BV	10	20	10	20	10
VV	10	20	10	20	10

4. New mathematical formulation

In this section, we first transform the QCDCP-IR to an equivalent scheduling problem, and then formulate a new integer program. We follow Goodchild and Daganzo (2006)'s work by extending their scheduling approach to the QCDCP-IR and thus anticipate that the QCDCP-IR can be viewed as a single machine makespan minimization scheduling problem (with precedence constraints and sequence-dependent setup time) and the service time of a sequence of jobs in the QCDCP-IR is equal to the makespan of a schedule in the scheduling problem.

For simplicity, we use a concise representation of the problem hereafter. For ease of illustration, we index all the containers in both plans. Fig. 5 illustrates a new representation of the example given in Fig. 3 with indices, where n, the number of all containers, is equal to 15, which are stowed in $m = 4$ stacks. In Fig. 5, all fixed containers are dropped, as they are not in our consideration.

4.1. Problem transformation

A feasible solution to the QCDCP-IR must respect the stacking dependent accessibility. Fig. 6 gives the precedence digraph of the example problem. Each container in both plans is represented by a node. Because the precedence constraints only exist among containers in each stack (e.g., job 1 must be processed before job 2, and job 11 must be processed before job 12), and the containers in the arrival plan must be processed before those in the departure plan (e.g., job 2 must be processed before job 10), and thus the precedence relation of containers in each stack can be mapped as a chain. Therefore, in the precedence digraph, there are m parallel chains (here $m = 4$) for all n jobs (here $n = 15$). Each node is also associated with a two-tuple (i,j), which indicates that the container is located at the j-th position (from the chain head) in the i-th chain in the digraph. Such a set of notation is useful in our new formulation, which we will explain later. In Fig. 6, redundant arcs are reduced by the transitive property of precedence relations.

To make internal-reshuffling operations (corresponding VV jobs) easy to handle, we reduce the solution search space by the following observations.

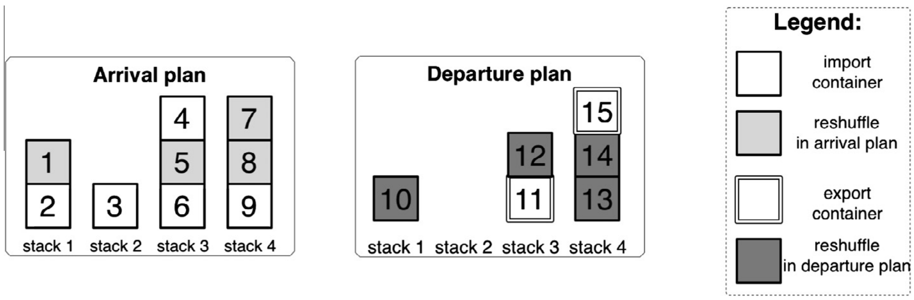

Fig. 5. A concise representation of the example.

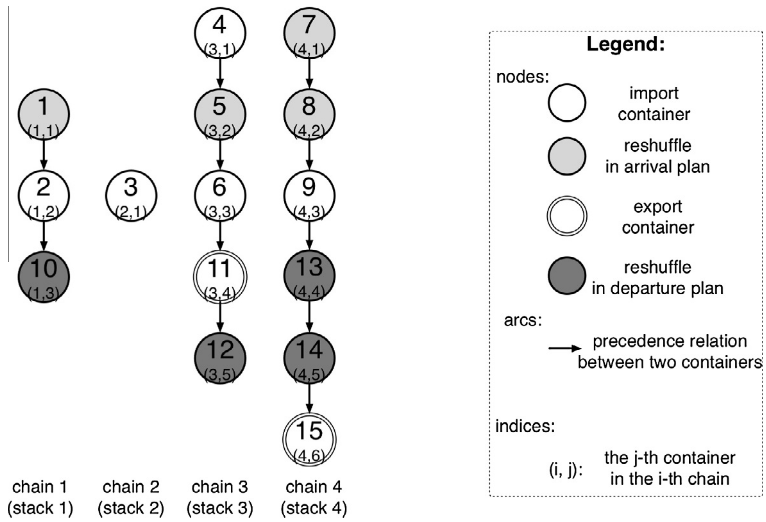

Fig. 6. Precedence digraph of the example problem.

Observation 1. *A VV job is equivalent to a VB job immediately followed a BV job, with a virtual setup time of* $\bar{s} = p^{VV} - (p^{VB} + p^{BV})$ *in between (e.g., see Fig. 7).*

For any sequence which contains VVs, we can replace these VV jobs according to Observation 1, and thus no sequences under our consideration contain VVs hereafter. As compensation, for such sequences, we need to further record where each VV is replaced by a couple of VB and BV and add a setup time $\bar{s}$ in between. This observation serves as a preprocessing step, and after this step each sequence in the search space consists of exact n jobs (i.e., the number of containers in both plans).

Now there exists a phenomenon that in some sequences, the setup time between consecutive VB and BV is $s^{VB,BV}$ (e.g., 10 s as in Table 2), and in some sequences that value is $\bar{s}$ (e.g., -110 s according to Table 2). We have $s^{VB,BV} > \bar{s}$, because the former corresponds to an unloading operation immediately followed by a loading operation for a reshuffle and the latter corresponds to an internal-reshuffling operation. We have the following observation.

Observation 2. *In a job sequence, if a VB job is immediately followed by a BV job with setup time* $s^{VB,BV}$ *in between, then this sequence is dominated by a corresponding sequence where each setup time* $s^{VB,BV}$ *is replaced by a setup time* $\bar{s}$.

Applying the above two observations, we efficiently reduce the search space. Now Tables 1 and 2 can be updated to Tables 3 and 4. Note that in these tables, VVs are dropped (by Observation 1), and the setup time between VB and BV is replaced by "-110" (by Observation 2).

Moreover, we have the following one-to-one relation, which maps each job (representing a container operation) to a container in both plans.

Observation 3. *In a job sequence, (i) each VY job corresponds to an import container, (ii) each VB job corresponds to a reshuffle in the arrival plan, (iii) each YV job corresponds to an export container, and (iv) each BV job corresponds to a reshuffle in the departure plan.*

By this observation, we can use VY, VB, YV and BV to denote the corresponding containers, respectively, and thus each container is also referred as a job with a certain type. (A sequence of containers can be used to portray a sequence of

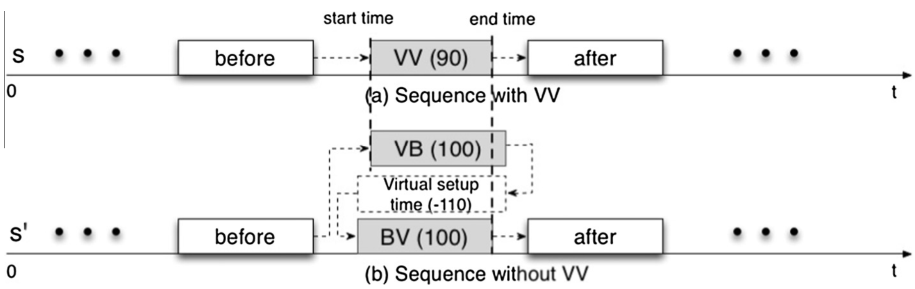

Fig. 7. A sequence with VV and a sequence without VV.

Table 3
Processing time p^t.

t	p^t
VY	100
YV	100
VB	100
BV	100

Table 4
Setup time s^{tu}.

s^{tu}	VY	YV	VB	BV
VY	20	10	20	10
YV	10	20	10	20
VB	20	10	20	−110
BV	10	20	10	20

container operations). The QC service time is the total time necessary to complete all the n jobs. The QCDCP-IR can be viewed as a kind of single machine makespan minimization scheduling problem, with precedence constraints and sequence-dependent setup time, which is NP-hard. Specifically, the quay crane is regarded as the single machine. Before the first job in a sequence, there is no setup time, or to say the setup time is zero. The scheduling problem aims to determine a sequence of jobs with minimum makespan (i.e., completion time).

4.2. Integer programming formulation

Based on the precedence digraph, we formulate a new integer program for the QCDCP-IR. Given is a precedence digraph with m chains and n jobs (e.g, see Fig. 6). Let $\mathcal{J}$ = {VY, VB, YV, BV} denote the set of job types. We also use the following notation.

Indices:

- (i,j): job index, indicating the job located at the j position in the i-th chain, where positions are indexed in a chain from head (top) to tail (bottom);
- t,u: job type indices, where $t,u \in \mathcal{J}$;
- k: position index in the sequence, $k \in \{1,2,\ldots,n\}$.

Parameters:

- $\mathcal{VY}$: set of VY-type jobs;
- $\mathcal{YV}$: set of YV-type jobs;
- $\mathcal{VB}$: set of VB-type jobs;
- $\mathcal{BV}$: set of BV-type jobs;
- $\mathcal{G}$: set of jobs of all types, i.e., $\mathcal{G} = \mathcal{VY} \bigcup \mathcal{YV} \bigcup \mathcal{VB} \bigcup \mathcal{BV}$;
- p^t: processing time of a t-type job, where $t \in \mathcal{J}$;
- s^{tu}: setup time of a u-type job, immediately preceded by a t-type job, where $\{t,u\} \in \mathcal{J}$.

For the digraph in Fig. 6, we know $\mathcal{VY} = \{(1,2),(2,1),(3,1),(3,3),(4,3)\}$, $\mathcal{YV} = \{(3,4),(4,6)\}$, $\mathcal{VB} = \{(1,1),(3,2),(4,1),(4,2)\}$ and $\mathcal{BV} = \{(1,3),(3,5),(4,4),(4,5)\}$.

Decision variables:

- x_{ijk}: equal to 1 if job (i,j) is arranged at the sequence's k-th position (i.e., the k-th processed); 0 otherwise.
- τ_k^t: equal to 1 if the sequence's k-th position is occupied by a t-type job, $t \in \mathcal{J}$; 0 otherwise.
- e_k^{tu}: equal to 1 if the sequence's k-th position is possessed by a t-type job, and the $(k+1)$-th position is occupied by a u-type job, where $k \in \{1,2,\ldots,n-1\}$; 0 otherwise. That is, $e_k^{tu} = \tau_k^t \cdot \tau_{k+1}^u$.

Let $M1$ denote a fixed integer, larger than all s^{tu}. In our formulation, we use a modified objective function Z' instead of the original objective function value Z, such that $Z = Z' - M1(n-1)$. For the minimization problem, the usage of Z' is to make variable e_k^{tu} preferable to be zero, guaranteed by positive multipliers $\{s^{tu} + M1\}$. The graph-based integer program (GBIP) is given below.

$$\text{(GBIP model)}\quad \min Z' = \sum_{k=1}^{n}\sum_{t\in\mathcal{J}} p^t \cdot \tau_k^t + \sum_{k=1}^{n-1}\sum_{\{t,u\}\in\mathcal{J}} (s^{tu} + M1)\cdot e_k^{tu} \tag{1}$$

$$\text{s.t.}\quad \sum_{k=1}^{n} x_{ijk} = 1,\quad (i,j)\in\mathcal{G}. \tag{2}$$

$$\sum_{k=1}^{n} x_{ijk}\cdot k + 1 \leqslant \sum_{k=1}^{n} x_{i,j+1,k}\cdot k,\quad \{(i,j),(i,j+1)\}\in\mathcal{G}. \tag{3}$$

$$\sum_{t\in\mathcal{J}} \tau_k^t = 1,\quad k\in\{1,2,\ldots,n\}. \tag{4}$$

$$\tau_k^{VY} = \sum_{(i,j)\in\mathcal{VY}} x_{ijk},\quad k\in\{1,2,\ldots,n\}. \tag{5}$$

$$\tau_k^{YV} = \sum_{(i,j)\in\mathcal{YV}} x_{ijk},\quad k\in\{1,2,\ldots,n\}. \tag{6}$$

$$\tau_k^{VB} = \sum_{(i,j)\in\mathcal{VB}} x_{ijk},\quad k\in\{1,2,\ldots,n\}. \tag{7}$$

$$\tau_k^{BV} = \sum_{(i,j)\in\mathcal{BV}} x_{ijk},\quad k\in\{1,2,\ldots,n\}. \tag{8}$$

$$e_k^{tu} \geqslant \tau_k^t + \tau_{k+1}^u - 1,\quad \{t,u\}\in\mathcal{J}, k\in\{1,2,\ldots,n-1\}. \tag{9}$$

$$x_{ijk}\in\{0,1\},\quad (i,j)\in\mathcal{G}, k\in\{1,2,\ldots,n\}. \tag{10}$$

$$\tau_k^t\in\{0,1\},\quad t\in\mathcal{J}, k\in\{1,2,\ldots,n\}. \tag{11}$$

$$e_k^{tu}\in\{0,1\},\quad \{t,u\}\in\mathcal{J}, k\in\{1,2,\ldots,n\}. \tag{12}$$

The objective function (1) minimizes the makespan (i.e., completion time), which is the sum of job processing times and setup times. Note that the original objective value is restored by adding $M1(n-1)$ to Z'. Constraints (2) ensure that each job (i,j) is processed exactly once, by summing up all positions k for each job. Constraints (3) guarantee that precedence relations are satisfied. In particular, item $\sum_{k=1}^{n} x_{ijk}\cdot k$ denotes the position in the sequence occupied by job (i,j) and item $\sum_{k=1}^{n} x_{i,j+1,k}\cdot k$ denotes the one possessed by job $(i,j+1)$. As job (i,j) must be processed before job $(i,j+1)$ due to precedence relations, the position of (i,j) in the sequence must be earlier than the position of job $(i,j+1)$ by at least one unit. Constraints (4) impose that only one job (of any type, by summing up all types) can be processed at each position k in the sequence. Constraints (5)–(8) guarantee the corresponding relation between τ_k^t and t-type jobs. Constraints (9) state the relation between the setup time and the related two jobs. Note that e_k^{tu} is minimized by the objective function. Domains of the decision variables are defined by (10)–(12).

Our GBIP model is suitable for different values of processing times and setup times (i.e., not restrictive to the data in Tables 3 and 4). To express the relation $e_k^{tu} = \tau_k^t \cdot \tau_{k+1}^u$, two sets of constraints $e_k^{tu} \leqslant \tau_k^t$ and $e_k^{tu} \leqslant \tau_{k+1}^u$ are theoretically required. However, as the objective function Z' forces e_k^{tu} to be zero, these two set of constraints are redundant in our formulation. (This may be one reason why our formulation is CPLEX-effective.) Moreover, the number of container operations in the sequence is a fixed number n, whereas in Meisel and Wichmann (2010)'s formulation (see Appendix A) the number of operations is upper bounded by n. (This may be another reason why our model is relatively competitive.)

5. Solution approach to large-scale problems

CPLEX is not suitable for large-scale NP-hard problems. In this section, we devise a very time-effective heuristic and analyze its worst-case error bound.

5.1. A fast algorithm

We design a constructive heuristic, called Internal-Reshuffle-Dense (IRD) algorithm. The intuitive idea is to maximize the number of internal-reshuffling operations. In particular, we update the digraph by reindexing all the chains in a decreasing

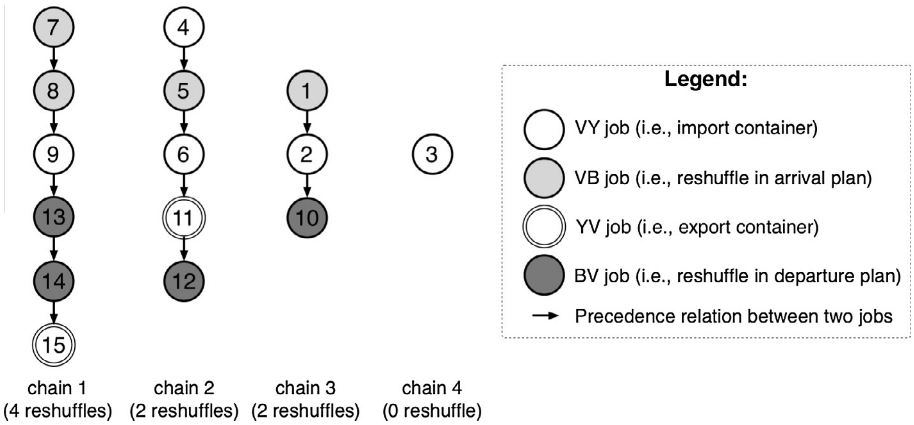

Fig. 8. Updated digraph of the example problem.

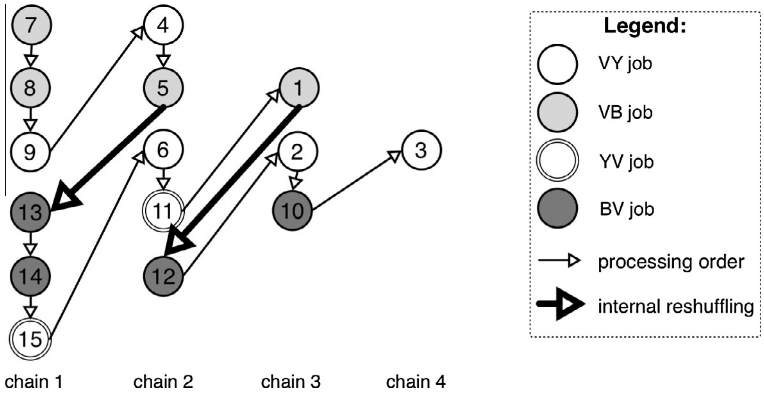

Fig. 9. IRD sequence of the example problem.

order of the number of reshuffles (see Fig. 8). Then we construct internal-reshuffling operations between two adjacent chains as many as possible (see Fig. 9).

In the following, a job is called *available* if it appears at the head (i.e., first position) of a chain. Note that we assume reshuffles exist in the problem input, since otherwise the problem can be solved to optimum by Johnson's rule (see Appendix B).

Internal-Reshuffle-Dense (IRD) algorithm

Step 1: Rearrange all chains in a decreasing order of the number of reshuffles. Let $k = 0$. (Suppose there are m chains indexed from 1.)
Step 2: $k := k + 1$. If $k = m$, go to Step 5.
Step 3: Process jobs in chain k until a BV job is available. If none, go to Step 2.
Step 4: Process jobs in chain $k + 1$ until a VB job is available. If none, go to Step 2. Process the VB job in chain $k + 1$ and the BV job in chain k. Go to Step 3.
Step 5: Process all the remaining jobs from top to bottom, chain by chain.

As each stack has an equal number of reshuffles in both plans, chain rearrangement in Step 1 ensures that all VB jobs in chain k and an equal number of BV jobs in chain $k - 1$ could construct internal-reshuffling operations, where $k = \{2, \ldots, m\}$. In other words, all VB jobs in chain k are internally reshuffled. For example, jobs 5 and 1 in Fig. 9 are internally reshuffled.

Remark 2. IRD algorithm takes $O(\max\{m \log m, n\})$ time, which is polynomial in input n, where $m \leqslant n$.

Let α denote the number of internal-reshuffling operations in a sequence (e.g., in Fig. 9, $\alpha = 2$). Use s_0 to denote the time an internal-reshuffling takes (e.g., according to Table 4, $s_0 = -110$). We use s_1 to indicate the setup time between an import container and an export container (no matter which goes first), and s_2 the setup time between two import (or export) containers (e.g., according to Table 4, $s_1 = 10$ and $s_2 = 20$). Clearly, $s_1 < s_2$. Each job has a processing time p (e.g., according to Table 3, $p = 100$). Recall that n denotes the number of containers in both plans.

Theorem 1. *The error bound of IRD algorithm is* $\frac{(n-1-\alpha)(s_2-s_1)}{np+\alpha s_0+(n-1-\alpha)s_1}$.

Proof. Let C_A denote the makespan of an IRD sequence, C_{OPT} an optimal objective value. Clearly, $C_A \leqslant np + \alpha s_0 + (n-1-\alpha)s_2$ and $C_{OPT} \geqslant np + \alpha s_0 + (n-1-\alpha)s_1$. Therefore the error bound is

$$\frac{C_A - C_{OPT}}{C_{OPT}} \leqslant \frac{(n-1-\alpha)(s_2-s_1)}{np+\alpha s_0+(n-1-\alpha)s_1}.$$

□

Corollary 1. *The error bound of IRD algorithm is 1/10 with regard to the data settings in* Meisel and Wichmann (2010).

Proof. Let C_A and C_{OPT} respectively denote the objective value achieved by IRD algorithm and the optimal solution. By Theorem 1 and the data settings in Meisel and Wichmann (2010) (also see Tables 3 and 4), we know

$$\frac{C_A - C_{OPT}}{C_{OPT}} \leqslant \frac{(n-1-\alpha)(s_2-s_1)}{np+\alpha s_0+(n-1-\alpha)s_1} = \frac{n-\alpha-1}{11n-12\alpha-1}.$$

Let $f(n,\alpha) = \frac{n-\alpha-1}{11n-12\alpha-1}$. By definitions of n (number of containers in both plans) and α (number of internal-reshuffling operations), we have $n \geqslant 1$ and $\alpha \leqslant \lfloor n/2 \rfloor$. Clearly, $n \geqslant 2\alpha$. For fixed α, the partial derivative of $f(n,\alpha)$ is

$$\frac{\partial f(n,\alpha)}{\partial n} = \frac{(s_2-s_1)(\alpha s_0+\alpha p+p)}{((p+s_1)n+(\alpha s_0-\alpha s_1-s_1))^2} = \frac{10-\alpha}{(11n+12\alpha+1)^2}.$$

We discuss three cases of α.

Case 1: $0 \leqslant \alpha < 10$. In this case, we know $\frac{\partial f(n,\alpha)}{\partial n} > 0$ and the function $f(n,\alpha)$ attains the maximum as n approaches infinity, i.e.,

$$\max f(n,\alpha) = \lim_{n\to\infty} \frac{n-\alpha-1}{11n-12\alpha-1} = \frac{1}{11} < \frac{1}{10}.$$

Case 2: $\alpha = 10$. In this case, $f(n,\alpha) = \frac{n-11}{11n-121} = \frac{1}{11} < \frac{1}{10}$.

Case 3: $\alpha > 10$. In this situation, we know $\frac{\partial f(n,\alpha)}{\partial n} < 0$ and the function $f(n,\alpha)$ attains the maximum when $n = 2\alpha$, i.e.,

$$\max f(n,\alpha) = f(2\alpha,\alpha) = \frac{\alpha-1}{10\alpha-1} = \frac{1}{10} - \frac{9}{100\alpha-10} < \frac{1}{10}.$$

This completes the proof. □

5.2. Comparison via example problem

We use the example provided in Meisel and Wichmann (2010) to demonstrate how IRD algorithm works. Then, based on this example, we compare the results generated with IRD algorithm, Johnson's rule (Goodchild and Daganzo, 2006), and GRASP heuristic (Meisel and Wichmann, 2010). In the example (see Fig. 3), there are five import containers, two export containers, four reshuffles, and two fixed containers. As fixed containers do not affect the decision process, we omit them in the concise representation (Fig. 5). The task is to convert the arrival plan to the departure plan. Below, we detail the solutions generated with different approaches.

- A sequence generated with IRD algorithm. We illustrate the details of how IRD algorithm works by Fig. 10, or to say how the IRD sequence, 7–8–9–4–5–13–14–15–6–11–1–12–2–10–3, depicted by Fig. 9 corresponds to operating time. In Fig. 10, arrows denote the moving direction of the quay crane, and dotted box indicates the starting slot of a container operation. In total, there are 13 container operations, and the total operating time is 1460 s. Moreover, we also depict where internal-reshuffling operations and double-cycling occur. This sequence has two internal-reshuffling operations (i.e., 5–13 and 1–12) and double-cycling happens twice (i.e., 6–11 and 2–10). In-bay empty crane move occurs four times (i.e., 15–6, 11–1, 12–2 and 10–3).

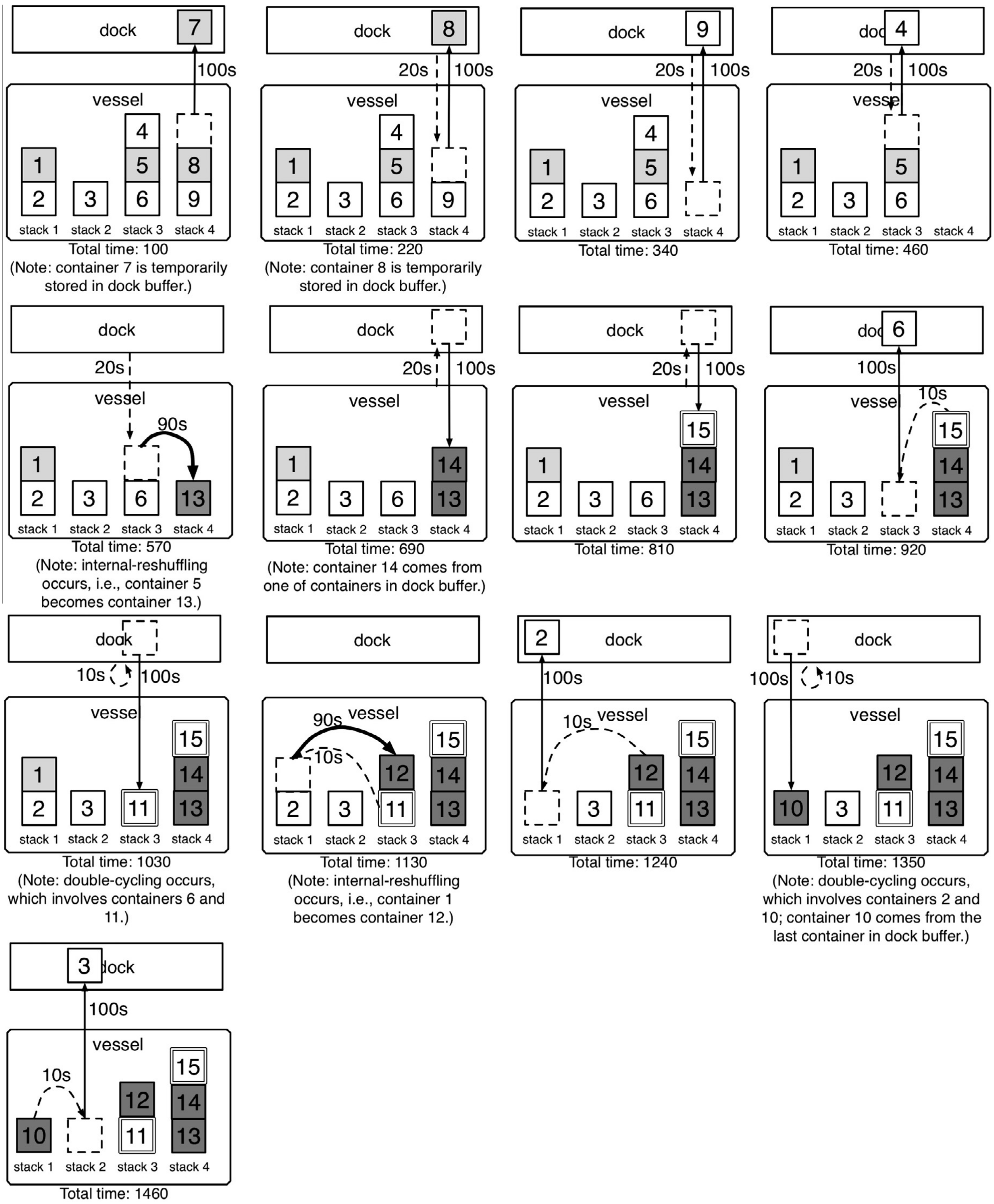

Fig. 10. IRD sequence: 7–8–9–4–5–13–14–15–6–11–1–12–2–10–3.

- A sequence generated with Johnson's rule. It is well known that Johnson's rule can be applied to solve the QCDCP (Goodchild and Daganzo, 2006). For the given example, the Johnson-rule sequence is 7–8–9–4–13–5–14–6–15–1–11–2–12–3–10. In total, there are 15 operations, and the time is 1670. In this sequence, double-cycling happens six times (i.e., 4–13, 5–14, 6–15, 1–11, 2–12, and 3–10), as expected, because the purpose of Johnson's rule is to maximize the number of double cycles. In-bay empty crane move happens five times (i.e., 13–5, 14–6, 15–1, 11–2 and 12–3).

- A sequence generated with GRASP heuristic. As Meisel and Wichmann (2010) devise a GRASP heuristic to solve the problem. The GRASP sequence given in their work is 4–7–8–9–1–13–5–14–6–15–11–2–12–3–10. There are 13 container operations, and the objective value is 1450. In this sequence, internal reshuffling operation occurs twice (i.e., 1–13, 5–14), double-cycling happens three times (i.e., 6–15, 2–12 and 3–10). Besides, in-bay empty move occurs four times (i.e., 13–5, 14–6, 11–2 and 12–3).
- A sequence generated with CPLEX. For the above example, an optimal solution can be obtained by CPLEX with either Meisel and Wichmann (2010)'s model or our GBIP model (see Section 4.2). An optimal solution is 7–8–9–13–4–5–14–6–11–1–12–2–10–3–15. To save space, we do not depict the details of operations. In total, there are 13 container operations, in which internal-reshuffling operation occurs twice (i.e., 5–14 and 1–12). Double-cycling happens four times (i.e., 9–13, 6–11, 2–10 and 3–15). The total time consumed is 1430. In the optimal sequence, in-bay empty move occurs five times (i.e., 13–4, 14–6 11–1, 12–2 and 10–3).

Table 5 sums up the results. The first column reports the objective function values. The second presents the optimality gap (compared with CPLEX solution). The following three columns illustrate the number of internal-reshuffling operations, the number of double-cycling and the number of in-bay empty moves, respectively. The last column illustrates time complexity of different methods. By comparison, we observe that (i) internal-reshuffling operations help to reduce service time by 16.8% with the sole application of the double cycling technique, (ii) IRD solution is better than Johnson-rule solution by 14.7%, (iii) IRD solution has an optimality gap 2.1%, and (iv) IRD algorithm's performance is slightly weaker than that of GRASP by 0.7%. As for time complexity, (i) IRD algorithm is approximately equal to Johnson's rule and (ii) IRD algorithm is significantly superior to GRASP heuristic. Besides, IRD algorithm generates the same number of internal-reshuffling operations as CPLEX does.

Remark 3. Notice that Johnson's rule generates an optimal solution to the problem without internal-reshuffling operations (i.e., QCDCP), whereas the CPLEX solution is an optimal solution to the problem with internal-reshuffling operations (i.e., QCDCP-IR). For this example, the internal-reshuffling method reduces the total service time by 16.8%, and it reduces the operations related to the four reshuffles by $(4\times 2-(2+2\times 2))/(4\times 2)=25\%$.

6. Computational experiments

In this section, we conduct computational experiments for the following purposes: (i) to evaluate the quality of GBIP model presented in Section 4.2, compared with the start-of-the-art model and (ii) to assess the solution quality of IRD algorithm, compared with GRASP heuristic. IRD algorithm is coded in C++ language. GRASP heuristic is reimplemented according to the description in Meisel and Wichmann (2010) in C++ language. CPLEX 12.5 is used to solve the two integer programming models. All the experiments are conducted on a Macbook with 2.4 GHz and 8 GB RAM (with software Xcode 4.6.3).

6.1. Instance generation

Experiments are conducted on a large set of test instances.

6.1.1. Bay sizes

All the bay sizes description in Meisel and Wichmann (2010) are included in our experiments. Besides, to evaluate the efficiency of our GBIP model, an "extra small" bay size is designed, as CPLEX cannot generate optimal solutions for relatively large instances. To accommodate mega-ships, such as Maersk Triple-E class of mega-ship with TEU capacity of 18,270 (see Port Technology International, 2015), we also add an "extra large" bay size (see Table 6).

6.1.2. Workload scenarios

Meisel and Wichmann (2010) generate three workload scenarios, i.e., *high load*, *low import*, and *low export*, which are distinguished by the ratio of import containers, export containers, reshuffles and fixed containers in the arrival and departure plans. In the example given in Fig. 3, there are five import containers, two export containers, four reshuffles (in either plan)

Table 5
Comparison between IRD, Johnson's rule, GRASP and CPLEX.

Methods	Objective value (sec.)	Optimality gap (%)	#Internal-reshuffling	# Double-cycling	#In-bay empty move	Time complexity
IRD	1460	2.1	2	2	4	$O(\max\{m\log m, n\})$
Johnson's rule	1670	16.8	0	6	5	$O(m\log m)$
GRASP	1450	1.4	2	3	4	Non-polynomial
CPLEX	1430*	0	2	4	5	Non-polynomial

* Optimal objective function value.

Table 6
Bay sizes for instance generation.

Bay sizes	No. of stacks	No. of tiers
extra small	5	5
small	10	10
medium	15	15
large	20	20
extra large	25	25

Table 7
Workload scenarios for instance generation.

Workload scenario	Import containers (%)	Export containers (%)	Reshuffles (%)	Fixed containers (%)
High load	70	70	2–20	10
Low import	40	70	2–20	10
Low export	70	40	2–20	10
Low load	40	40	5–50	10

Table 8
Computational results of extra-small bay size.

Workload scenario	RR (%)	GBIP model (CPLEX)			M&W model (CPLEX)			GRASP		IRD		Lower bound
		Value	Time (sec.)	#Opt.	Value	Time (sec.)	#Opt.	Value	Time (sec.)	Value	Time (sec.)	
High load	2	4450	292	5	4450	387	3	4620	23	4685	<0.01	4390
	4	4450	5	5	4450	175	5	4500	147	4690	<0.01	4390
	6	4450	5	3	4450	174	3	4525	10	4690	<0.01	4390
	8	4400	303	4	4400	388	3	4585	23	4690	<0.01	4330
	10	4450	5	2	4450	175	1	4510	148	4690	<0.01	4390
	12	4400	538	2	4400	600	0	4590	22	4615	<0.01	4330
	14	4490	328	3	4410	524	3	4420	10	4625	<0.01	4270
	16	4160	600	1	4270	600	0	4290	10	4275	<0.01	4090
	18	4130	600	1	4180	600	1	4200	26	4265	<0.01	4030
	20	4450	589	2	4470	600	2	4530	4	4640	<0.01	4390
Low import	2	3370	32	5	3370	47	5	3500	5	3520	<0.01	3290
	4	3370	22	5	3370	33	5	3470	84	3525	<0.01	3290
	6	3370	10	5	3370	38	5	3525	5	3530	<0.01	3290
	8	3370	15	5	3370	39	5	3460	86	3530	<0.01	3290
	10	3250	142	5	3250	312	5	3370	5	3455	<0.01	3170
	12	3320	138	5	3320	202	5	3345	30	3475	<0.01	3230
	14	3370	36	5	3370	70	5	3460	5	3525	<0.01	3290
	16	3370	71	5	3370	185	5	3430	5	3525	<0.01	3290
	18	3260	110	5	3260	333	4	3290	86	3480	<0.01	3170
	20	3150	226	5	3150	490	3	3200	5	3245	<0.01	3050
Low export	2	3330	45	5	3330	50	5	3400	12	3480	<0.01	3290
	4	3330	2	5	3330	7	5	3405	205	3485	<0.01	3290
	6	3330	84	5	3330	97	5	3425	12	3485	<0.01	3290
	8	3330	86	4	3330	172	4	3400	208	3490	<0.01	3290
	10	3330	80	5	3330	60	5	3410	12	3490	<0.01	3290
	12	3270	144	5	3270	257	4	3350	207	3490	<0.01	3230
	14	3220	301	4	3220	304	3	3310	12	3415	<0.01	3170
	16	3100	600	3	3100	600	1	3175	12	3225	<0.01	3050
	18	3330	82	4	3330	99	4	3465	207	3490	<0.01	3290
	20	3170	600	2	3170	600	2	3230	501	3280	<0.01	3110
Low load	5	2690	6	5	2690	4	5	2700	3	2810	<0.01	2630
	10	2590	52	5	2590	52	5	2720	53	2710	<0.01	2510
	15	2690	5	5	2690	4	5	3000	182	2760	<0.01	2630
	20	2630	16	5	2630	32	5	2970	3	2710	<0.01	2570
	25	2580	51	5	2580	115	5	2715	53	2610	<0.01	2510
	30	2580	119	5	2580	159	5	2800	1	2645	<0.01	2510
	35	2460	85	5	2460	163	5	2565	11	2610	<0.01	2390
	40	2400	65	5	2400	278	5	2575	62	2460	<0.01	2330
	45	2460	58	5	2460	133	5	2725	206	2610	<0.01	2390
	50	2470	71	5	2470	115	5	2605	1	2565	<0.01	2390
Average		3460	181	4.25	3460	251	3.90	3494	12	3625	<0.01	3390
GBIP's improvement on Time = (251–181)/251 = 28%, improvement on #Opt. = (4.25–3.90)/3.90 = 9%												
Average Gaps of GRASP and IRD (%)								4.13		7.02		

and two fixed containers. The capacity of a bay is 12 containers or slots (4 stacks $\times$ 3 tiers). The ratio of import containers in the arrival plan is $5/12 \approx 42\%$, the ratio of export containers in the departure plan is $2/12 \approx 17\%$, the ratio of reshuffles is $4/12 \approx 33\%$ in either plan. In their computational experiments, the percentage of fixed containers is set to 10% in all scenarios and the percentage of reshuffles varies in the range [0, 2%, . . ., 20%]. Note that by Theorem 2 in Appendix B, we show that an optimal solution to such a situation (with no reshuffles) can be obtained by Johnson's rule. Thus, it is not necessary to test the cases with 0% reshuffles. Besides, to study the cases where reshuffles pose a high percentage, we extensively design a fourth scenario, called *low load* (see Table 7). In the fourth scenario, the percentage of fixed containers is set to 10% as in other three scenarios, and the percentage of reshuffles varies in the range [5%, 10%, . . ., 50%]. Thus, in each workload scenario, there are 10 values of reshuffle ratios or percentages. Table 7 summarizes the workload scenarios used in our computational study.

In our experiments, there are in total 200 combinations (5 bay sizes $\times$ 4 workload scenarios $\times$ 10 reshuffle percentages). For each combination, five instances are generated by randomly assigning the different types of containers to slots in the plans. In total, 1000 benchmark instances are generated. In our tests, the data illustrated in Tables 1 and 2 are used, as in Meisel and Wichmann (2010).

6.2. Computational results

To demonstrate that our GBIP model (see Section 4.2) is CPLEX-effective, we compare it with Meisel and Wichmann (2010)'s formulation, called M&W model, both solved with CPLEX. For some cases of extra small bay size, CPLEX cannot

Table 9
Computational results of small bay size.

Workload scenario	RR (%)	GBIP model (CPLEX)			GRASP			IRD			Lower bound
		Value	Time (sec.)	#Int.	Value	Time (sec.)	Gap (%)	Value	Time (sec.)	Gap (%)	
High load	2	None	600	2	18,230	466	4.17	18,915	<0.01	8.27	17,470
	4	None	600	4	18,100	546	3.81	18,910	<0.01	8.62	17,410
	6	None	600	4	18,260	460	4.65	18,915	<0.01	8.64	17,410
	8	18,990	600	5	17,970	349	5.79	17,845	<0.01	5.40	16,930
	10	None	600	4	17,425	267	3.53	17,935	<0.01	6.69	16,810
	12	18,920	600	5	17,510	523	4.34	18,270	<0.01	9.07	16,750
	14	18,870	600	5	17,370	255	2.88	17,770	<0.01	5.33	16,870
	16	18,990	600	5	17,040	107	3.11	17,345	<0.01	5.06	16,510
	18	None	600	4	16,350	297	2.32	17,275	<0.01	8.17	15,970
	20	18,990	600	5	17,015	83	4.38	17,345	<0.01	6.61	16,270
Low import	2	14,250	600	5	13,645	139	5.09	14,050	<0.01	8.49	12,950
	4	14,190	600	5	13,425	320	3.54	13,765	<0.01	6.29	12,950
	6	14,270	600	5	13,035	128	2.03	13,645	<0.01	6.85	12,770
	8	14,250	600	5	13,745	135	5.57	14,200	<0.01	9.40	12,980
	10	14,170	600	5	12,740	362	1.65	13,200	<0.01	5.35	12,530
	12	14,270	600	5	13,125	439	4.53	13,410	<0.01	7.02	12,530
	14	14,200	600	5	13,140	560	3.27	13,580	<0.01	6.85	12,710
	16	14,130	600	5	13,345	409	4.31	13,625	<0.01	6.70	12,770
	18	14,270	600	5	12,890	140	4.19	13,270	<0.01	7.45	12,350
	20	14,270	600	5	12,285	382	3.87	12,625	<0.01	6.90	11,810
Low export	2	13,570	511	5	13,705	199	3.39	14,190	<0.01	7.18	13,240
	4	13,820	600	5	13,600	96	3.01	14,190	<0.01	7.58	13,190
	6	13,970	600	5	13,490	415	4.00	13,790	<0.01	6.49	12,950
	8	14,180	600	5	13,150	167	4.26	13,190	<0.01	4.77	12,590
	10	14,050	600	5	13,285	167	3.88	13,190	<0.01	3.29	12,770
	12	14,080	600	5	13,235	209	3.51	13,190	<0.01	3.29	12,770
	14	14,030	600	5	13,225	479	5.71	13,690	<0.01	9.78	12,470
	16	13,960	600	5	13,085	129	3.78	13,190	<0.01	4.77	12,590
	18	14,190	600	5	12,170	434	1.97	12,690	<0.01	6.37	11,930
	20	14,190	600	5	12,260	104	4.16	12,690	<0.01	8.00	11,750
Low load	5	10,890	600	5	11,530	245	8.41	11,390	<0.01	7.86	10,560
	10	11,240	600	5	10,170	268	2.16	10,600	<0.01	6.53	9950
	15	11,360	600	5	11,045	499	11.00	10,390	<0.01	5.70	9830
	20	11,350	600	5	10,490	269	4.58	10,890	<0.01	8.79	10,010
	25	11,160	600	5	10,965	439	8.71	10,390	<0.01	3.80	10,010
	30	11,390	600	5	10,610	265	12.44	9890	<0.01	6.46	9290
	35	11,380	600	5	9815	479	5.35	10,090	<0.01	8.61	9290
	40	11,390	600	5	9495	151	5.32	9790	<0.01	8.90	8990
	45	11,390	600	5	9775	417	8.64	9750	<0.01	9.18	8930
	50	11,390	600	5	9255	323	9.35	8950	<0.01	6.67	8390
Average					13742.5	395	5.91	13932.5	<0.01	7.75	12,930

Note. Value "None" means CPLEX does not deliver integer solutions for some instances in one combination.

optimally solve M&W model in 3600 s. Thus, we set a time limit of 600 s in CPLEX for both models, and count the number of optimal solutions obtained in the tests. Lower bounds can be obtained by the procedure provided by Meisel and Wichmann (2010), who demonstrate the bounds generated with their procedure are better than those generated with CPLEX in one hour in most of the cases. Therefore, we use this procedure to calculate lower bounds.

Table 8 shows the computational results of extra small bay size in all four scenarios. Every row in this table reports average results for a combination of five instances. Column "Workload scenario" indicates the four workload scenarios. Column "RR (%)" gives the reshuffle ratios or percentages. The third to fifth columns report the average objective function values obtained by solving GBIP model with CPLEX, and the average running times, and the number of optimal values among five instances, respectively. The sixth to eighth columns report those of M&W model. The ninth to eleventh (twelfth to fourteenth) columns represent the average objective function values obtained with GRASP heuristic (IRD algorithm) and its average running times, respectively. The last column reports the average lower bounds delivered by the lower-bounding procedure (Meisel and Wichmann, 2010). In the experiments, the units of objective function values and running time are seconds (sec. for short).

Results in Table 8 demonstrate that on average using GBIP model CPLEX can optimally solve 4.25 instances (of five instances in each combination) in the time limit, whereas this value of M&W model is 3.90. Thus, in terms of solution quality, compared with M&W model, GBIP model can obtain 9% more optimal solutions in the time limit. In terms of running time, results show that GBIP model has an improvement about 28%, compared with M&W model. In this table, the superiority of GBIP model is clearly shown. The "Average Gap (%)" in the last row reports the relative error gaps of IRD and GRASP for all the

Table 10
Computational results of medium bay size.

Workload scenario	RR (%)	GBIP model (CPLEX)			GRASP			IRD			Lower bound
		Value	Time (sec.)	#Int.	Value	Time (sec.)	Gap (%)	Value	Time (sec.)	Gap (%)	
High load	2	None	600	0	39,675	569	1.29	41,300	<1	5.44	39,170
	4	None	600	0	41,175	334	4.96	42,815	<1	9.14	39,230
	6	None	600	3	40,110	584	2.87	42,745	<1	9.63	38,990
	8	None	600	3	40,190	752	5.51	41,370	<1	8.61	38,090
	10	None	600	2	39,745	635	3.69	41,370	<1	7.93	38,330
	12	None	600	3	39,780	345	5.77	38,945	<1	3.55	37,610
	14	None	600	2	38,370	475	2.84	39,500	<1	5.87	37,310
	16	None	600	3	38,495	626	5.38	38,455	<1	5.27	36,530
	18	None	600	3	37,260	567	3.70	38,010	<1	5.79	35,930
	20	42,890	600	5	38,130	323	5.77	38,005	1	5.42	36,050
Low import	2	None	600	0	31,080	455	5.53	30,050	<1	2.04	29,450
	4	None	600	1	29,775	79	2.99	30,000	<1	3.77	28,910
	6	None	600	0	30,265	398	3.19	32,135	<1	9.56	29,330
	8	None	600	2	29,890	519	5.14	29,935	<1	5.29	28,430
	10	None	600	2	29,870	646	4.62	31,105	<1	8.95	28,550
	12	None	600	1	29,910	262	6.10	30,210	<1	7.17	28,190
	14	None	600	4	29,290	346	3.68	30,105	<1	6.57	28,250
	16	None	600	1	28,960	165	4.06	30,210	<1	8.55	27,830
	18	None	600	1	28,200	246	4.02	29,000	<1	6.97	27,110
	20	None	600	3	28,605	486	4.36	28,650	1	4.52	27,410
Low export	2	None	600	1	30,580	419	3.84	32,015	<1	8.71	29,450
	4	None	600	2	30,210	643	3.21	32,080	<1	9.60	29,270
	6	None	600	2	29,595	537	1.53	31,090	<1	6.66	29,150
	8	None	600	3	30,070	61	5.32	30,085	<1	5.38	28,550
	10	None	600	2	29,040	441	2.15	31,090	<1	9.36	28,430
	12	None	600	4	30,570	414	7.08	30,090	<1	5.39	28,550
	14	32,060	600	5	29,480	308	5.93	28,660	<1	2.98	27,830
	16	32,090	600	5	29,405	611	5.21	30,375	<1	8.68	27,950
	18	32,090	600	5	28,295	219	4.14	29,825	<1	9.77	27,170
	20	None	600	3	27,990	453	2.34	29,345	<1	7.29	27,350
Low load	5	None	600	1	26,830	299	14.71	25,585	<1	9.38	23,390
	10	None	600	2	23,825	108	4.82	24,730	<1	8.80	22,730
	15	None	600	2	24,530	263	7.92	23,590	<1	3.78	22,730
	20	None	600	2	23,850	319	9.86	23,230	<1	7.00	21,710
	25	None	600	4	24,435	667	13.49	23,230	<1	7.90	21,530
	30	None	600	4	22,380	167	9.76	22,230	<1	9.02	20,390
	35	25,730	600	5	21,150	609	5.28	21,910	<1	9.06	20,090
	40	None	600	4	22,105	695	12.04	20,575	<1	4.28	19,730
	45	25,730	600	5	19,575	311	2.33	20,445	<1	6.87	19,130
	50	25,730	600	5	19,885	313	6.97	19,980	<1	7.48	18,590
Average					29,780	441	4.13	30,640	<1	6.46	28,880

Note. Value "None" means CPLEX does not deliver integer solutions for some instances in one combination.

combinations, where a relative error gap of each combination is calculated by $(Value - Lower\ bound)/Lower\ bound \times 100\%$. In terms of solution quality, on average, IRD algorithm has a relative error bound of 7.02%, slightly weaker than that of GRASP by 2.89%. In terms of running time, IRD algorithm clearly outperforms GRASP, with a running time far less than one second, whereas GRASP consumes 181 s on average.

Table 9 presents the results of small bay size. We only report the CPLEX solutions solving with GBIP model, as M&W model is less CPLEX-effective, shown previously. The purpose is to report the best solutions of CPLEX in 600 s for the instances. Column "#Int." reports the number of integer (feasible) solutions obtained for each combination of five instances, as for some instances CPLEX cannot deliver integer solutions. In Column "Value", we present the average objective function values where integer solutions are obtained for all five instances, as CPLEX cannot generate integer solutions for some instances. Column "Gap (%)" gives the relative error gap for each combination. Results show that CPLEX can obtain feasible solutions for most instances of small bay size, however these objective values are not competitive, compared with the values generated with IRD algorithm. Generally, the GRASP heuristic outperforms the IRD algorithm in terms of solution quality, as IRD algorithm has a relative error gap 7.75% on average, slightly weaker than that of GRASP by 1.84%. Notice that for some combinations, the relative error gap of GRASP is larger than 10%, whereas the IRD algorithm guarantees a relative error bound less than 10%. Thus, the IRD algorithm is relatively robust. In terms of running time, IRD algorithm clearly outperforms GRASP, with running time less than one second, whereas GRASP consumes 395 s. On average, in terms of solution quality, IRD outperforms GRASP in the low load scenario.

Table 10 reports the computational results of the instances of medium bay size. As CPLEX has not totally lost its power for small bay sized instances, we also report the solutions generated with CPLEX (with GBIP model). Results show that CPLEX

Table 11
Computational results of large bay size.

Workload scenario	RR (%)	GRASP			IRD			Lower bound
		Value	Time (sec.)	Gap (%)	Value	Time (sec.)	Gap (%)	
High load	2	70,635	20	1.74	72,240	<1	4.05	69,430
	4	72,140	220	5.54	71,625	<1	4.79	68,350
	6	70,540	613	3.93	69,145	<1	1.88	67,870
	8	68,965	587	2.06	69,085	<1	2.24	67,570
	10	68,855	404	3.37	71,390	<1	7.18	66,610
	12	67,550	274	2.71	70,390	<1	7.02	65,770
	14	68,505	295	4.73	71,390	<1	9.14	65,410
	16	68,400	441	4.76	68,390	<1	4.75	65,290
	18	66,825	645	4.86	66,390	1	4.17	63,730
	20	65,965	360	4.49	66,890	1	5.96	63,130
Low import	2	53,900	334	3.75	54,200	<1	4.33	51,950
	4	54,810	234	5.51	54,200	<1	4.33	51,950
	6	52,945	273	2.51	54,200	<1	4.94	51,650
	8	51,790	112	1.21	53,275	<1	4.11	51,170
	10	51,235	402	1.92	54,830	<1	9.07	50,270
	12	50,675	150	1.41	54,350	<1	8.77	49,970
	14	51,530	216	4.63	52,350	<1	6.29	49,250
	16	51,300	686	4.04	53,350	<1	8.19	49,310
	18	50,285	284	4.91	50,440	<1	5.24	47,930
	20	50,625	481	4.32	50,350	<1	3.75	48,530
Low export	2	54,885	110	4.68	56,115	<1	7.03	52,430
	4	51,930	428	1.37	55,190	<1	7.73	51,230
	6	52,850	223	3.53	53,765	<1	5.32	51,050
	8	53,415	296	5.88	53,205	<1	5.46	50,450
	10	53,480	651	5.50	53,630	<1	5.80	50,690
	12	51,525	777	4.75	52,625	<1	6.98	49,190
	14	50,500	171	2.79	51,215	<1	4.24	49,130
	16	50,705	574	5.53	50,615	<1	5.34	48,050
	18	50,105	217	3.25	51,190	<1	5.48	48,530
	20	48,710	171	2.92	51,965	<1	9.79	47,330
Low load	5	44,390	515	8.51	42,520	<1	3.94	40,910
	10	43,000	311	7.31	43,970	<1	9.73	40,070
	15	43,720	337	11.45	41,970	<1	6.98	39,230
	20	42,350	639	8.95	41,540	<1	6.87	38,870
	25	38,715	203	2.61	41,040	<1	8.77	37,730
	30	38,580	454	6.13	38,430	<1	5.72	36,350
	35	39,405	268	11.16	38,415	<1	8.36	35,450
	40	38,045	332	10.50	36,470	<1	5.93	34,430
	45	35,435	450	3.46	35,400	<1	3.36	34,250
	50	34,445	441	4.60	35,400	<1	7.50	32,930
Average		52,540	231	3.17	53,820	<1	5.77	51,180

Table 12
Computational results of extra large bay size.

Workload scenario	RR (%)	GRASP			IRD			Lower bound
		Value	Time (sec.)	Gap (%)	Value	Time (sec.)	Gap (%)	
High load	2	112,040	96	3.10	115,200	<1	6.01	108,670
	4	111,140	504	3.18	114,700	<1	6.49	107,710
	6	109,840	620	2.66	114,700	<1	7.21	106,990
	8	110,010	540	4.23	114,340	<1	8.33	105,550
	10	108,680	642	3.61	109,340	<1	4.24	104,890
	12	106,970	448	4.13	109,270	1	6.37	102,730
	14	106,175	372	3.78	109,770	1	7.29	102,310
	16	104,705	309	3.19	110,200	1	8.60	101,470
	18	104,605	276	4.64	107,270	2	7.30	99,970
	20	98,260	197	1.21	106,050	2	9.23	97,090
Low import	2	85,395	98	4.28	89,615	<1	9.43	81,890
	4	84,010	345	3.50	86,400	<1	6.44	81,170
	6	85,515	480	6.06	87,470	<1	8.48	80,630
	8	83,885	369	6.57	81,690	<1	3.79	78,710
	10	81,405	317	3.82	82,615	<1	5.36	78,410
	12	78,620	514	1.98	83,540	<1	8.37	77,090
	14	79,585	244	3.48	79,995	<1	4.01	76,910
	16	77,165	496	2.16	80,190	<1	6.17	75,530
	18	77,625	317	3.27	80,615	1	7.24	75,170
	20	77,680	517	5.53	79,840	1	8.46	73,610
Low export	2	86,645	420	6.43	89,410	<1	9.83	81,410
	4	83,520	372	3.05	86,490	<1	6.71	81,050
	6	83,510	692	4.04	86,490	<1	7.75	80,270
	8	81,685	327	3.62	82,940	<1	5.21	78,830
	10	83,565	536	5.29	85,990	<1	8.34	79,370
	12	82,380	427	5.55	82,940	1	6.27	78,050
	14	79,990	413	3.60	82,715	1	7.13	77,210
	16	79,755	418	4.11	80,990	1	5.72	76,610
	18	79,655	426	4.71	79,415	1	4.40	76,070
	20	75,835	548	2.44	79,340	1	7.17	74,030
Low load	5	69,945	29	8.36	69,545	<1	7.74	64,550
	10	66,135	380	5.80	65,540	<1	4.85	62,510
	15	65,990	530	10.00	63,905	<1	6.53	59,990
	20	61,670	297	4.05	61,970	<1	4.56	59,270
	25	60,865	356	6.35	60,040	<1	4.91	57,230
	30	58,430	610	3.29	59,540	<1	5.25	56,570
	35	61,260	634	11.85	56,980	<1	4.04	54,770
	40	57,515	382	7.61	58,760	<1	9.93	53,450
	45	53,735	123	5.01	54,980	<1	7.45	51,170
	50	54,325	358	8.07	54,980	1	9.37	50,270
Average		83,182	227	5.58	85,090	<1	7.69	79,470

cannot deliver integer solutions for most of the instances. The average relative error gap of IRD is 6.46%, slightly weaker than that of GRASP by 2.33%. The maximum relative error gap of GRASP is 14.71%, whereas that value of IRD is 9.56%. Therefore, IRD algorithm is relatively robust. In terms of running time, IRD consumes less than one second, whereas GRASP takes 441 s on average. Thus, IRD saves a lot of computational time. On average, in terms of solution quality, IRD outperforms GRASP in the low load scenario.

The computational results of the instances of large bay size is given in Table 11. As shown previously that CPLEX cannot generate integer solutions for most of the instances of medium bay size in the time limit, we only reports solutions generated with GRASP and IRD. On average, the relative error gap IRD is 5.77%, slightly weaker than that of GRASP by 2.60%. In Table 12, the instances of extra large bay size are tested. On average, IRD has a relative error gap 7.69%, slightly weaker than that of GRASP by 2.11%. On average, in terms of solution quality, IRD outperforms GRASP in the low load scenario.

Summing up, (i) our GBIP model is CPLEX-effective compared with Meisel and Wichmann (2010)'s model, (ii) IRD algorithm, taking about one second to solve problems of practical size, is very efficient in terms of running time, (iii) IRD algorithm is relative robust with a relative error gap less than 10%, (iv) on average, in terms of solution quality, IRD outperforms GRASP in the scenario with high reshuffle percentages, and (v) with the increase of problem sizes, the difference between the two average relative error gaps decreases. Our experiment results also demonstrate that although the solution quality of IRD algorithm is slightly weaker than that of GRASP heuristic by about 2%, IRD algorithm can save a lot of computational time. Moreover, as IRD is a constructive heuristic, it is very easy to implement.

7. Conclusion

In this paper, we revisit the quay crane double cycling problem with internal-reshuffling operations. For the problem, we propose some observations to map container operations to containers in the arrival and departure plans. Then we present a new integer programming model. Compared with Meisel and Wichmann (2010)'s formulation, experimental results demonstrate our formulation is CPLEX-effective. To efficiently solve real-world problem instances, we devise a constructive heuristic approach with worst-case error guarantee. Experimental results demonstrate that our algorithm is very fast and relative robust than the existing method. As the internal-reshuffling method can be used to complement the classic double-cycling technique to further improve QC efficiency, our algorithm is very attractive because it is low-cost and very easy to implement.

Future research directions may include the general problems where the following assumptions are overcome: (i) every container to be reshuffled in the arrival plan can be positioned at any slot reserved for reshuffles in the departure plan, (ii) the number of containers under reshuffling in one stack is the same in both plans, and (iii) the hatch-covered problems are not considered. Moreover, (i) exact solution approaches, such as branch-and-price algorithm, are needed for the QCDCP-IR and its generalization, and (ii) aiming at looking for a double-cycling operation while pursuing internal reshuffling, the possibility of inserting a look-ahead step within IRD should be further explored.

Acknowledgement

The authors would like to acknowledge the constructive comments by the referees and the editor. The authors thank Prof. Chung-Yee Lee for valuable suggestions. The work described in this paper was supported by the National Natural Science Foundation of China (Grant Nos. 71101106, 71428002, 71272045, and 71072026).

Appendix A

We restate the integer programming formulation proposed by Meisel and Wichmann (2010). Moreover, we correct some mistakes, i.e., constraints (16)–(18) are corrected.

Meisel and Wichmann (2010)'s **model with corrections**

In Meisel and Wichmann (2010)'s formulation, (i,j) denotes the container occupying the position in i-th stack and j-th tier in a plan (see Fig. 3). Note that such a definition of (i,j) differs from ours. Clearly, the container in slot $(i,j+1)$ is laid on top of the container in slot (i,j).

Parameters:

m: number of stacks in the bay;
n: number of tiers in the bay;
A^I: set of slots that hold an import container in the arrival plan;
A^R: set of slots that hold a reshuffle in the arrival plan;
A: $A = A^I \bigcup A^R$, i.e., set of slots that hold an import container or a reshuffle in the arrival plan;
D^E: set of slots that hold an export container in the departure plan;
D^R: set of slots that hold a reshuffle in the departure plan;
D: $D = D^E \bigcup D^R$, i.e., set of slots that hold an export container or a reshuffle in the departure plan;
T: set of container operation types, $T = \{VY, YV, VB, BV, VV\}$;
d^t: processing time of a container operation of type $t \in T$;
d^{tu}: time needed for empty crane spreader movement in-between a container operation of type t and a container operation of type u, where $t, u \in T$;
l: $l = |A| + |D|$, i.e., an upper bound on the number of container operations;
$K, \overline{K}$: $K = \{1, 2, \ldots, l\}$ and $\overline{K} = \{1, 2, \ldots, l-1\}$, i.e., index sets of container operations.

Variables:

x^V_{ijk}: equal 1 if the container at bay slot (i,j) is picked in move $k \in K$; 0 otherwise;
x^Y_k: equal to 1 if a container is picked from the yard in move k; 0 otherwise;
x^B_k: equal to 1 if a container is picked from the buffer in move k; 0 otherwise;
y^V_{ijk}: equal to 1 if a container is dropped at bay slot (i,j) in move k; 0 otherwise;
y^Y_k: equal to 1 if a container is dropped at the yard in move k; 0 otherwise;
y^B_k: equal to 1 if a container is dropped at the buffer in move k; 0 otherwise;
τ^t_k: equal to 1 if container operation k is of type $t \in T$; 0 otherwise;

e_k^{tu}: equal to 1 if container operation k is of type $t \in T$ and operation $k+1$ is of type $u \in T$, i.e., $\tau_k^t = \tau_{k+1}^u = 1$; 0 otherwise;
b_k: number of externally buffered reshuffles at the end of operation k.

$$\text{(M\&W model)} \quad \min Z = \sum_{k \in K} \sum_{t \in T} d^t \cdot \tau_k^t + \sum_{k \in \overline{K}} \sum_{t,u \in T} d^{tu} \cdot e_k^{tu} \tag{13}$$

$$s.t. \quad \sum_{k \in K} x_{ijk}^V = 1, \quad \forall (i,j) \in A. \tag{14}$$

$$\sum_{k \in K} y_{ijk}^V = 1, \quad \forall (i,j) \in D. \tag{15}$$

$$\sum_{k \in K} x_{i,j+1,k}^V \cdot k + 1 \leqslant \sum_{k \in K} x_{ijk}^V \cdot k, \quad \forall (i,j), (i,j+1) \in A. \tag{16}$$

$$\sum_{k \in K} y_{ijk}^V \cdot k + 1 \leqslant \sum_{k \in K} y_{i,j+1,k}^V \cdot k, \quad \forall (i,j), (i,j+1) \in D. \tag{17}$$

$$\sum_{k \in K} x_{ijk}^V \cdot k + 1 \leqslant \sum_{k \in K} y_{ijk}^V \cdot k, \quad \forall (i,j) \in A \bigcap D. \tag{18}$$

$$\sum_{t \in T} \tau_k^t \leqslant 1, \quad \forall k \in K. \tag{19}$$

$$\tau_k^{VY} = \sum_{(i,j) \in A^I} x_{ijk}^V, \quad \forall k \in K. \tag{20}$$

$$\tau_k^{VY} = y_k^Y, \quad \forall k \in K. \tag{21}$$

$$\tau_k^{YV} = x_k^Y, \quad \forall k \in K. \tag{22}$$

$$\tau_k^{YV} = \sum_{(i,j) \in D^E} y_{ijk}^V, \quad \forall k \in K. \tag{23}$$

$$\tau_k^{VV} + \tau_k^{VB} = \sum_{(i,j) \in A^R} x_{ijk}^V, \quad \forall k \in K. \tag{24}$$

$$\tau_k^{VB} = y_k^B, \quad \forall k \in K. \tag{25}$$

$$\tau_k^{BV} = x_k^B, \quad \forall k \in K. \tag{26}$$

$$\tau_k^{VV} + \tau_k^{BV} = \sum_{(i,j) \in D^R} y_{ijk}^V, \quad \forall k \in K. \tag{27}$$

$$\sum_{t \in T} \tau_k^t \geqslant \sum_{t \in T} \tau_{k+1}^t, \quad \forall k \in \overline{K}. \tag{28}$$

$$e_k^{tu} \geqslant \tau_k^t + \tau_{k+1}^u - 1, \quad t,u \in T, \forall k \in \overline{K}. \tag{29}$$

$$b_0 = 0. \tag{30}$$

$$b_k = b_{k-1} - x_k^B + y_k^B, \quad \forall k \in K. \tag{31}$$

$$b_k \in Z^+, \quad \forall k \in K. \tag{32}$$

$$x_{ijk}^V, x_k^Y, x_k^B, \tau_k^t, e_k^{tu} \in \{0,1\}, \quad \forall (i,j) \in A, k \in K. \tag{33}$$

$$y^V_{ijk}, y^Y_k, y^B_k \in \{0,1\}, \quad \forall (i,j) \in D, k \in K. \tag{34}$$

The objective function (13) is to minimize the service time of the bay, which is the sum of total time needed for container operations and total time needed for empty spreader move. Constraints (14) ensure that each import container and each reshuffle in the arrival plan is picked up once by the QC. Constraints (15) make sure that exactly one container is placed at every export and every slot to be hold by a reshuffle in the departure plan. Constraints (16) guarantee that the pick-up of slot $(i,j+1)$ precedes the pick-up of slot (i,j) in the arrival plan. (We fixed this set of constraints in Meisel and Wichmann, 2010.) Constraints (17) ensure that the drop-off of slot (i,j) precedes the drop-off of slot (i,j) in the departure plan. Constraints (18) ensure that a container is removed from a slot before a new container is loaded to that place. (We fixed this set of constraints in Meisel and Wichmann, 2010) Constraints (19) make sure at most one container is moved in operation $k \in K$. Constraints (20) and (21) guarantee that one import container (or one *VY* move) corresponds to one pick-up and one drop-off operations. Constraints (22) and (23) provide the relation between an export container and its pick-up and drop-off operations. Similarly, constraints (24)–(27) ensure the relation between one reshuffle and its pick-up and drop-off operations. Constraints (28) enforce a consecutive sequence of non-empty container operations in case that less than l moves are contained in a solution. Constraints (29) establish the relation between one empty spreader move and the related two container operations. Constraints (30) and (32) guarantee the number of reshuffles in the dock buffer is not less than zero. Domains of the decision variables are defined in (33) and (34).

Remark 4. This formulation is a natural representation as the problem appears.

Appendix B

In this part, we identify a special case for the QCDCP-IR which can be solved in polynomial time. The potential benefit may be that if an instance is recognized as such a case, then it is can be easily solved.

B.1. The non-reshuffle QCDCP-IR

We consider the case where no reshuffles exist. That is, the problem contains only import containers and export containers, and thus no internal-reshuffling operations could ever happen. We denote this special case by the nr-QCDCP-IR. Note that the nr-QCDCP-IR is different from the QCDCP, because in the QCDCP container's operation time and crane's empty move time (from the dock to the ship, or in a reverse direction) are identical, whereas the nr-QCDCP-IR involves different kinds of empty move times. Thus, the nr-QCDCP-IR is not trivial.

For simplicity, let s_1 denote the setup time between an import container and an export container (no matter which goes first), and s_2 the setup time between two import (or export) containers. Clearly, $s_1 < s_2$.

Lemma 1. *The nr-QCDCP-IR is to maximize the number of* s_1.

Proof. This is because the number of setup times is a constant $n-1$, where n denotes the number of containers. The more time s_1 appears in the sequence, the smaller the makespan is. □

B.2. Two machine flow shop

Following Goodchild and Daganzo (2006)'s work by extending their methodology, we regard each stack as a job (with two sequential operations) for the two machine flow shop problem. The first (second) operation of a specific job is constructed by concatenating the import (export) containers from top to bottom (bottom to top) in the arrival (departure) plan. Clearly, the processing time of its first (second) operation is the number of import (export) containers. See Fig. 11a for illustration. The flow shop problem allows preemption at integral time points, since each container is represented by a processing request of length one. Moreover, only active schedules are considered, i.e., there is no unnecessary inserted idleness in the schedule.

Remark 5. All precedence constraints have been captured within flow shop job definition.

We next map a flow shop schedule to a single machine sequence. Given a flow shop schedule, we can obtain a job sequence for single machine by the following procedure (e.g., see Fig. 11b).

Mapping Procedure

Step 1: Set time $t = 0$.

Step 2: Choose and add to the targeted sequence, from the schedule of flow shop, the job segment which starts at time t on machine 1 (if any), and then the job segment which starts at time t on machine 2 (if any). If no further processing exists, stop; otherwise, set $t := t+1$ and go to Step 2.

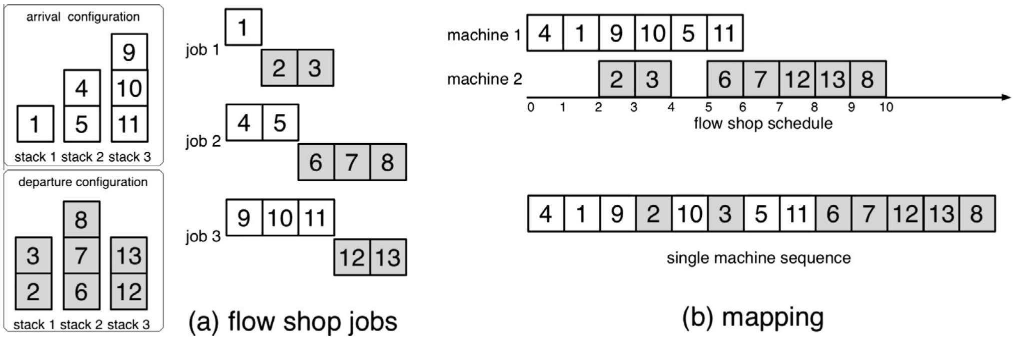

Fig. 11. Two machine flow shop.

Lemma 2. *The mapped single machine sequence is feasible.*

Proof. This is because there is no violation of precedence relations. □

Lemma 3. *In the single machine sequence, s_1 occurs $2k-1$ times, where k denotes the number of time units when both machines are simultaneously busy.*

Proof. Observe that s_1 (in the mapped single machine sequence) occurs when the processing crosses machines in the flow shop schedule. For a given flow shop schedule with k common busy time units, assuming there is a virtual "line" between the two machines, then the processing of jobs crosses this line $2k-1$ times. □

Lemma 4. *The nr-QCDCP-IR aims to maximize the number of k.*

Proof. By Lemma 1. □

Lemma 5. *The nr-QCDCP-IR is equivalent to the two machine flow shop makespan minimization problem.*

Proof. Observe that two machine flow shop makespan minimization problem is equivalent to maximize the number of time units when both machines are simultaneously busy. It is well known that preemption has no advantage for this flow shop problem. By Lemma 4, the proposition follows. □

Recall that Johnson's rule optimally solves two machine flow shop makespan minimization problem (Pinedo, 2012). Let $u(i)$ ($l(i)$) denote the processing time of the first (second) operation of job J_i.

Johnson's rule

(i) If $u(i) < l(i)$, assign job J_i to group A; otherwise, assign it to group B.
(ii) Arrange jobs in group A in non-decreasing order of $u(i)$, and jobs in group B in non-increasing order of $l(i)$.
(iii) Construct the final sequence in which all jobs in group A are followed by all jobs in group B.

Theorem 2. *The nr-QCDCP-IR can be optimally solved by Johnson's rule.*

Proof. By Lemma 5. □

References

Bell, M.G.H., Liu, X., Angeloudis, P., Fonzone, A., Hosseinloo, S.H., 2011. A frequency-based maritime container assignment model. Transp. Res. Part B 45 (8), 1152–1161.
Bell, M.G.H., Liu, X., Rioult, J., Angeloudis, P., 2013. A cost-based maritime container assignment model. Transp. Res. Part B 58, 58–70.
Bierwirth, C., Meisel, F., 2009. A fast heuristic for quay crane scheduling with interference constraints. J. Sched. 12 (4), 345–360.
Bierwirth, C., Meisel, F., 2010. A survey of berth allocation and quay crane scheduling problems in container terminals. Eur. J. Oper. Res. 202 (3), 615–627.

Bierwirth, C., Meisel, F., 2015. A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. Eur. J. Oper. Res. http://dx.doi.org/10.1016/j.ejor.2014.12.030.
Carlo, H.J., Vis, I.F.A., Roodbergen, K.J., 2013. Seaside operations in container terminals: literature overview, trends, and research directions. Flex. Serv. Manuf. J. http://dx.doi.org/10.1007/s10696-013-9178-3.
Choo, S., Klabjan, D., Simchi-Levi, D., 2010. Multiship crane sequencing with yard congestion constraints. Transp. Sci. 44 (1), 98–115.
Crainic, T.G., Kim, K.H., 2005. Intermodal transportation. In: Barnhart, C., Laporte, G. (Eds.), Transportation Handbooks in Operations Research and Management Science. North-Holland, Amsterdam, Netherlands.
Daganzo, C.F., 1989. Crane productivity and ship delay in ports. Transp. Res. Rec. 1251, 1–9.
Drewry Container Insight, 2014. <http://ciw.drewry.co.uk/release-week/2014-40>.
Goodchild, A.V., 2005. Crane Double Cycling in Container Ports: Algorithms, Evaluation, and Planning. Ph.D. thesis, Department of Civil and Environmental Engineering, University of California at Berkeley.
Goodchild, A.V., Daganzo, C.F., 2006. Double-cycling strategies for container ships and their effect on ship loading and unloading operations. Transp. Sci. 40 (4), 473–483.
Goodchild, A.V., Daganzo, C.F., 2007. Crane double cycling in container ports: planning methods and evaluation. Transp. Res. Part B: Methodol. 41 (8), 875–891.
Goodchild, A.V., McCall, J.G., Zumerchik, J., Lanigan Sr., J., 2011. Reducing train turn times with doubly cycling in new terminal designs. Transp. Res. Rec.: J. Transp. Res. Board 2238, 8–14.
Kim, K.H., Park, Y.-M., 2004. A crane scheduling method for port container terminals. Eur. J. Oper. Res. 156 (3), 752–768.
Lee, C-Y., Liu, M., Chu, C.B., 2014. Optimal algorithm for general quay crane double-cycling problem. Transp. Sci. http://dx.doi.org/10.1287/trsc.2014.0563.
Lim, A., Rodrigues, B., Xu, Z., 2007. A m-parallel crane scheduling problem with a non-crossing constraint. Naval Res. Logist. 54 (2), 115–127.
Liu, M., Zheng, F.F., Li, J.F., 2014. Scheduling small number of quay cranes with non-interference constraint. Optim. Lett. http://dx.doi.org/10.1007/s11590-014-0756-4.
Meisel, F., Wichmann, M., 2010. Container sequencing for quay cranes with internal-reshuffles. OR Spectrum 32 (3), 569–591.
Meng, Q., Wang, S.A., 2011. Liner shipping service network design with empty container repositioning. Transp. Res. Part E: Logist. Transp. Rev. 47 (5), 695–708.
Meng, Q., Wang, S.A., Andersson, H., Thun, K., 2014. Containership routing and scheduling in liner shipping: overview and future research directions. Transp. Sci. 48 (2), 265–280.
Pinedo, M., 2012. Scheduling: Theory, Algorithms, and Systems, fourth ed. Springer.
Port Technology International, 2015. Friday Focus: Maersk Triple-E Infographic. <http://www.porttechnology.org>.
Stahlbock, R., Voß, S., 2008. Operations research at container terminals: a literature update. OR Spectrum 30 (1), 1–52.
Wang, S.A., 2014. A novel hybrid-link-based container routing model. Transp. Res. Part E: Logist. Transp. Rev. 61, 165–175.
Wang, S.A., Liu, Z., Bell, M.G.H., 2015. Profit-based maritime container assignment models for liner shipping networks. Transp. Res. Part B 72, 59–76.
Wang, S.A., Meng, Q., 2012. Sailing speed optimization for container ships in a liner shipping network. Transp. Res. Part E 48 (3), 701–714.
Wang, S.A., Meng, Q., Liu, Z., 2013. Bunker consumption optimization methods in shipping: a critical review and extensions. Transp. Res. Part E 53, 49–62.
Wang, H., Wang, S.A., Meng, Q., 2014. Simultaneous optimization of schedule coordination and cargo allocation for liner container shipping networks. Transp. Res. Part E: Logist. Transp. Rev. 70, 261–273.
Zhang, H.P., Kim, K.H., 2009. Maximizing the number of dual-cycle operations of quay cranes in container terminals. Comput. Ind. Eng. 56 (3), 979–992.