The Quay Crane Scheduling Problem With Stability Constraints

Zizhen Zhang, Ming Liu, Chung-Yee Lee, and Jiahai Wang, *Member, IEEE*

***Abstract*— The quay crane scheduling problem (QCSP) is one of the most important problems for the operations at container ports. The QCSP aims to decide a QC schedule for loading and unloading containers so as to minimize the vessel turnaround time. The QCSP is subject to various kinds of constraints, e.g., task precedence constraints and QC noninterference constraints. This paper extends the QCSP by taking into consideration the stability constraints, which are crucial for the safety reason but often omitted in the existing literature. We provide a mathematical model for the QCSP with stability constraints (QCSPSCs). A bicriteria evolutionary algorithm is proposed to solve the QCSPSC. The algorithm consists of a sliding-window heuristic to fix the schedule, which violates the stability constraints. Extensive experiments are conducted to demonstrate the effectiveness of the algorithm. The computational results of the traditional QCSP and the QCSPSC are also compared and analyzed.**

***Note to Practitioners*—Stability is a vital factor for the safety of the ship and the goods inside. At a terminal, when a berthed container ship receives loading and unloading operations performed by quay cranes (QCs), it is required that the ship stability criterion does not exceed a certain threshold. In practice, ship operation managers are keen in minimizing the ship's turnaround time via an efficient QC schedule for loading and unloading containers without leaning the ship to one side. The schedule is then implemented by automatic scheduling systems or simulation tools in the terminal routines. This paper presents some practical stability constraints and integrates them into the traditional QC scheduling model. The results show that by considering the stability constraints, the ship will become more stable, but the turnaround time will not be affected significantly. We believe that the introduction of ship stability constraints can not only fill in the gap between the research on QC scheduling and the real-world ship handling practices but also enrich the studies on logistics planning and port operations.**

Manuscript received September 14, 2016; revised November 13, 2017; accepted January 11, 2018. Date of publication February 15, 2018; date of current version July 2, 2018. This paper was recommended for publication by Associate Editor M. P. Fanti and Editor M. P. Fanti upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grant 71601191, Grant 61673403, and Grant U1611262, and in part by the Research Grants Council of Hong Kong through the Theme-Based Research Grant under Grant T32-620/11. *(Corresponding author: Chung-Yee Lee.)*

Z. Zhang and J. Wang are with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510275, China, and also with the Guangdong Key Laboratory of Big Data Analysis and Processing, Sun Yat-sen University, Guangzhou 510275, China.

M. Liu is with the School of Economics and Management, Tongji University, Shanghai 200092, China.

C.-Y. Lee is with the Department of Industrial Engineering and Logistics Management, The Hong Kong University of Science and Technology, Hong Kong (e-mail: cylee@ust.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2018.2795254

***Index Terms*— Evolutionary algorithm, quay crane (QC) scheduling, sliding windows, stability constraints.**

I. Introduction

CONTAINER ports play crucial roles in global maritime logistics industries. According to UNCTAD [1], world container port throughput has increased by an estimate of 5.1% to 684.4 million Twenty-foot Equivalent Unit (TEUs) in 2014. The world fleet has more than doubled since 2001, reaching 1.75 billion deadweight tons in 2014. The key factors for the effective utilization of container ports are primarily determined by the efficiency of the stacking and the transportation of import/export containers to/from the ship's side [2]. To boost the productivity and enhance the port's competitive advantage, many automatic planning systems have been developed to provide customized solutions (e.g., [3]–[5]). Then, the next step is to introduce optimization techniques to improve the solution quality.

In general, container ports have different types of equipments and various kinds of resources in the daily operations. One of the most expensive and important equipment is the quay crane (QC), which performs the loading and unloading operations between ships and the wharf apron. This paper studies an extension of the so-called QC scheduling problem (QCSP), which aims at deciding a QC movement sequence for loading and unloading containers so as to minimize the operational cost or turnaround time. Fig. 1 illustrates a scenario of QCs scheduling. Upon a ship's arrival, it is first assigned to a berth for container loading and unloading. Because container ships are commonly large, the port may launch more than one QC working simultaneously. The ship is divided longitudinally into *bays*. Each QC serves one or several bays. Each bay is exclusively served by only one QC. A QC can perform the *gantry travel* moving between bays. It is commonly assumed that: 1) QCs are on the same track and thus cannot cross each other and 2) two QCs should not work too closely for safety reasons. The above two requirements are known as the *noninterference* constraints. A bay consists of several containers in *slots* indexed by their *row* numbers and *tier* numbers. A stowage plan designates the loading plan for export containers and unloading plan for import containers. For the sake of efficiency in loading and unloading, a bunch of export containers, which are adjacent to each other and have the same destination port, are grouped as a cluster. Likewise, a collection of adjacent import containers with the same origin

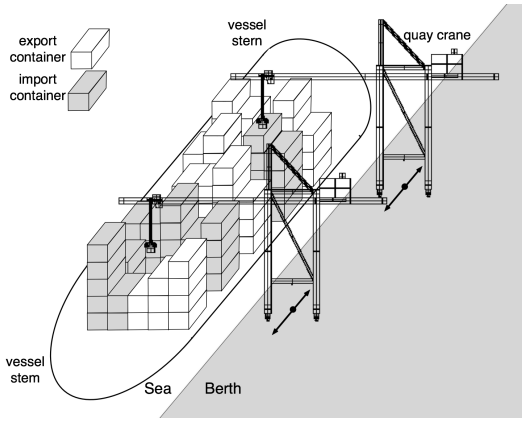


Fig. 1. Scenario of the QC scheduling.

port are also clustered. In the QCSP, a *task* is often defined as a loading or unloading operation for a cluster rather than a single container. Note that unloading tasks should always precede loading tasks in the same stack, which are known as the *precedence* constraints. In a nutshell, the scheduling of QCs is equivalent to finding an arrangement of tasks to QCs subject to the specified constraints.

The QCSP is a well-studied problem that attracts much academic attention. However, most of the existing studies on the QCSP only consider the attributes of tasks (e.g., task types, task durations, and precedence relation of tasks) or QCs (e.g., QC interference, initial positions of QCs, and QC travel times) [6]. Seldom considers the attributes of ships, particularly the stability of ships. The stability is the ability to keep the ships in the upright position. It is vital for the safety of ships no matter in voyage or in port. When a ship is served by QCs in port, loading or unloading a container will arise a heeling moment. Then, the ship will tend to roll toward a new stabilized state. This motion can badly interfere with other concurrent loading or unloading operations. What is worse, a dissatisfactory QC schedule may lead to a temporary condition that the ship leans to one side, potentially increasing the risk of capsizing.

In this paper, we take the stability of ships into consideration and investigate a QCSP with stability constraints (QCSPSC). The contributions of our works are threefold. First, we systematically introduce the stability constraints, which narrow the gap between the research on QCSP and the real-world practice. Second, we show that the proposed bicriteria framework with sliding-window heuristic produces promising solutions. Moreover, these techniques can be generalized to handling other versions of QCSPSC. Third, we conducted extensive experiments on the modified benchmark instances to evaluate our algorithm. The data set and comprehensive experimental results can serve as a baseline for future studies.

The remainder of this paper is organized as follows. Section II briefly reviews the relevant literature on the QCSP and the QCSPSC. In Section III, we provide a formal definition and a mathematical model of the QCSPSC. For solving this problem, a bicriteria evolutionary algorithm (BiEA) is devised and described in Section IV. To evaluate the proposed approach, Section V reports a series of experiments that we conducted based on the modified benchmark instances. As a comparison, the results of the QCSP without stability constraints are also provided. Finally, we conclude this paper in Section VI.

II. Literature Review

The logistics management of containers mainly consists of container terminal management (e.g., [7]–[11]) and container ship routing and scheduling (e.g., [12]–[16]). As the QCSP, which falls in the scope of container terminal management, has been studied for about two decades, we just briefly review the most relevant results in the following. The interested reader may refer to some excellent survey papers (e.g., [2], [6], [17], and [18]).

Daganzo [19] first studied static and dynamic QC work schedules with the objective of minimizing the total weighted departure time of vessels. To assign QCs to ship bays of multiple vessels, they proposed an exact branch-and-bound algorithm for the small instances and a heuristic approach for the practical large instances. An improved branch-and-bound algorithm was developed in a following paper [20]. However, some practical constraints, e.g., the interference of QCs, were simplified in these pioneering works.

Lim *et al.* [21] studied the QCSP with spatial constraints. The spatial constraints include the noncrossing constraint, the neighborhood constraint (or called the safety margin constraint), and the job-separation constraint. They proposed dynamic programming algorithms, a probabilistic tabu search, and a squeaky wheel optimization heuristic for finding solutions of the problem with different spatial constraints. Lee *et al.* [22] focused on the QCSP with interference constraints. They proved the NP-completeness of the problem by reducing it to the PARTITION problem. Genetic algorithms were proposed for finding near-optimal solutions. The interference constraints are also considered in some yard crane scheduling problems, when two or more yard cranes sharing the same traveling lane take place [23].

Kim and Park [24] considered the QCSP with interference constraints and precedence constraints. They proposed a branch-and-bound method and a greedy randomized adaptive search procedure (GRASP). They showed that the branch-and-bound method performs better in terms of the solution quality, but the GRASP yields less computational times. Moccia *et al.* [25] strengthened the model of [24]. They developed a branch-and-cut algorithm incorporating several families of valid inequalities. Their algorithm significantly improved the solutions for a set of QCSP benchmark data. Later, Sammarra *et al.* [26] presented a tabu search for the same QCSP model. The neighborhood of the tabu search is defined by performing the swapping move and the insertion move. The results showed that the tabu search can outperform the GRASP. Compared with the branch-and-cut algorithm, the tabu search can provide a good compromise between solution quality and computational times. For the same problem, Bierwirth and Meisel [27] employed a branch-and-bound algorithm to search a best solution within the

scope of unidirectional schedules, which only allow QCs move along the same direction. Chung and Choy [28] adopted a modified genetic algorithm with problem-specific crossover and mutation operators. They showed that their algorithm is very efficient, while the solution qualities are comparable with other existing approaches. A genetic algorithm is also used by Tavakkoli-Moghaddam *et al.* [29] for solving the QCSP and QC assignment problem. Recently, following [27], Legato and Trunfio [30] developed a local branching-based algorithm for the QCSP under unidirectional schedules.

Liu *et al.* [31] considered a complicated version of the traditional QCSP, which includes initial crane positions, moving speed, and interference condition for the cranes. They devised a heuristic decomposition approach to break down the problem into two interrelated smaller models, i.e., the vessel-level and the berth-level models. Later, Legato *et al.* [32] constructed a richer model for the QCSP based on [31]. They explicitly considered the speed and the time windows of different QCs. They focused on the unidirectional schedules. They developed a branch-and-bound algorithm for small instances and a timed Petri net approach for large problems. Meisel [33] also studied the QCSP by taking the crane time windows into account. The time windows may restrict the availability of cranes at a vessel. This paper provided a mathematical formulation and a tree-search-based heuristic solution method for solving the problem. Meisel and Bierwirth [34] used a unified approach for evaluating the performance of different QCSP models and solution procedures. They provided a QCSP data generator along with computational results, which can serve as a benchmark for the related research.

Goodchild and Daganzo [35] initiated the study of the stack-based QCSP model. Their work considers one QC to process the container stacks within one bay. According to the vessel physical structure, precedence-related tasks are defined for each container stack. Zhang and Kim [36] further explored the cases where deck covers are involved, which make the problem complicated. They developed a fast heuristic method. Recently, the computational difficulty of stack-based QCSP with deck covers was overcome by Lee *et al.* [9]. Meisel and Matthias [37] considered the QCSP on the basis of single containers. They provided an idea allowing reposition of containers in the bay instead of temporarily unloading them, and developed an efficient GRASP approach. Liu *et al.* [10] further studied this container-based QCSP model. They presented a mixed integer linear programming formulation and a very fast constructive heuristic method to solve the instances with practical sizes.

The stability of ships is very important in port operations. There are several works on the stowage planning problem considering ship stability constraints (e.g., [38]–[42]). The stowage planning decides the containers to be placed to what locations in the ship. For the stowage planning problem, the stability constraints are applied to ensuring the balance of the ship after it departs the port. In contrast, the QCSP requires that ship stability constraints must be satisfied during the whole process of QC operations.

To the best of our knowledge, there are relatively few works on the QCSP under the concern of stability. Wang *et al.* [43] first considered the stability of ships in the QCSP, but their approach was not extensively tested to demonstrate its effectiveness. Al-Dhaheri *et al.* [44] proposed an mixed integer programming model for the QCSP that takes into account the preemption, noncrossing, safety margin, QC traveling time, QC initial position, and vessel stability. In their paper, the stability is measured by the shift of the vessel's center of gravity, while horizontal and longitudinal stabilities are not considered. This paper is further extended by Al-Dhaheri and Diabat [45], which developed a Lagrangian relaxation-based algorithm to obtain satisfactory solutions for large-scale instances within acceptable times. Ursavas [46] presented a decision support system for the complex crane scheduling problem, in which the noncrossing restrictions, dynamic crane assignment policy, and vessel stability issues are considered.

III. Problem Definition

In this section, we first introduce some basic assumptions of the QCSP. Subsequently, a mixed integer linear programming model of the QCSP is provided. We then discuss the ship stability constraints and also include them in the model to formulate the QCSPSC.

A. Assumptions of the QCSP

As described in Section I, the scheduling of QCs is a very complex process in practice. To simplify the problem, the following assumptions are proposed by Kim and Park [24], and they are also made in this paper.

1) The objective of the problem is to minimize the makespan, i.e., the last completion time among all the tasks.
2) A task must be continuously served by only one QC during the whole process. The transport equipments on the yard are always ready for delivering/providing the containers.
3) The task is bay-based and is classified according to their positions (on the deck or in the hold), and their operation types (loading or unloading). The precedence constraints define the following priority levels in a bay: 1) unloading operations on the deck; 2) unloading operations in the hold; 3) loading operations in the hold; and 4) loading operations on the deck.
4) The noninterference constraints are considered, and the minimum safety margin between QCs is denoted as g bays, where g is a small and fixed integer.
5) The movement speed of QCs between two adjacent bays is the same.

Fig. 2 illustrates an example instance of the QCSP. In this example, there are six tasks located in different locations of the ship. For a task, the first element in the parentheses is the processing time and the last element is the operation type, where the symbol "↑" indicates an unloading operation and the symbol "↓" indicates a loading operation. The initial positions of QC1 and QC2 are at Bays 1 and 2, respectively. The traveling time of a QC between two adjacent bays is assumed to be one time unit. The safety margin is $g = 1$.

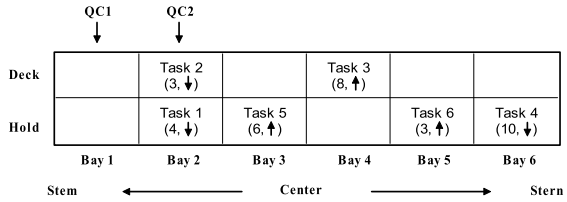


Fig. 2. Example instance of the QCSP.

An optimal solution of the problem is shown in Fig. 3, where QC1 sequentially processes tasks 5 and 3, and QC2 processes task 1, 2, 6, and 4. The optimal makespan is 24.

B. Mathematical Formulation of the QCSP

The QCSP has previously been formulated as mixed integer linear programming models (see [24] and [25] for more details). These models can be regarded as three-index formulations of the problem, as they introduce a set of binary variables x_{ijk} to indicate the order relations between task i and j on QC k. In this section, we reformulate the problem by a two-index model as follows.

Input Parameters:

- p_{ik} Processing time of task i by QC k.
- l_i Location of task i.
- q_k Initial position of QC k.
- t_{ij} Traveling time of the QC between the locations of task i and task j.
- t'_{ik} Traveling time of QC k moving from its initial position to the location of task i.
- g Minimum safety margin between QCs.

Sets:

- N Set of tasks, also interchangeably referred to as the number of tasks.
- Q Set of quay cranes, also interchangeably referred to as the number of quay cranes.
- Φ Set of precedence relations, i.e., $\Phi = \{ (i, j) \,|\, l_i = l_j$ and task i precedes task $j\}$.
- Θ Set of combinations of two task i, j and two QCs k, l which can potentially effect a conflict if i is assigned to k and j is assigned to l, i.e., $\Theta = \{ (i, j, k, l) \,|\, i \neq j, k \neq l, l_i \leq l_j, q_k \geq q_l\}$.

Indices:

- i, j Index of tasks, $i, j \in N$.
- k, h Index of quay cranes, $k, h \in Q$.

Decision Variables:

- $C_{\max}$ Makespan, i.e., the last completion time of all tasks.
- s_i Starting time of task i.
- c_i Completion time of task i.
- x_{ik} Binary variable which is equal to 1 if task i is assigned to QC k, and 0 otherwise.
- y_{ij} Binary variable which is equal to 1 if task j starts after the completion of task i, i.e., $s_i \geq c_j$, and 0 otherwise.

Two-Index Formulation for the QCSP:

$$\text{(QCSP)} \quad \min \ C_{\max} \tag{1}$$

$$\text{s.t. } C_{\max} \geq c_i, \quad i \in N \tag{2}$$

$$\sum_{k \in Q} x_{ik} = 1, \quad i \in N \tag{3}$$

$$s_i + \sum_{k \in Q} p_{ik} x_{ik} = c_i, \quad i \in N \tag{4}$$

$$y_{ij} + y_{ji} \leq 1, \quad i, j \in N, \ i \neq j \tag{5}$$

$$t'_{ik} x_{ik} \leq s_i, \quad i \in N, \ k \in Q \tag{6}$$

$$c_i \leq s_j, \quad (i, j) \in \Phi \tag{7}$$

$$y_{ij} = 1, \quad (i, j) \in \Phi \tag{8}$$

$$y_{ji} = 0, \quad (i, j) \in \Phi \tag{9}$$

$$(c_i + t_{ij}) - s_j \leq \mathcal{M}\ (2 + y_{ji} - x_{ik} - x_{jk}), \quad i, j \in N, \ i \neq j, \ k \in Q \tag{10}$$

$$(c_j + t_{ji}) - s_i \leq \mathcal{M}\ (2 + y_{ij} - x_{ik} - x_{jk}), \quad i, j \in N, \ i \neq j, \ k \in Q \tag{11}$$

$$(c_i + g) - s_j \leq \mathcal{M}\ (2 + y_{ji} - x_{ik} - x_{jl}), \quad (i, j, k, l) \in \Theta \tag{12}$$

$$(c_j + g) - s_i \leq \mathcal{M}\ (2 + y_{ij} - x_{ik} - x_{jk}), \quad (i, j, k, l) \in \Theta \tag{13}$$

$$c_i - s_j \leq \mathcal{M}\ (2 + y_{ji} - x_{ik} - x_{jl}), \quad i, j \in N, \ i \neq j, \ k, l \in Q, \ k \neq l, |l_i - l_j| \leq g \tag{14}$$

$$c_j - s_i \leq \mathcal{M}\ (2 + y_{ij} - x_{ik} - x_{jl}), \quad i, j \in N, \ i \neq j, \ k, l \in Q, \ k \neq l, |l_i - l_j| \leq g \tag{15}$$

$$x_{ik} \in \{0, 1\}, \quad i \in N, \ k \in Q \tag{16}$$

$$y_{ij} \in \{0, 1\}, \quad i, j \in N \tag{17}$$

$$s_i, c_i \geq 0, \quad i \in N. \tag{18}$$

In the above model, $\mathcal{M}$ is a sufficiently large number. The objective function (1) is to minimize the makespan of QCs, which is defined by Constraint (2). Constraint (3) ensures that each task is assigned to only one QC. Constraint (4) defines a time window $[s_i, c_i)$ (left-closed and right-open) for each task i. Constraint (5) limits the relative relation of time windows of two tasks. Constraint (6) restricts the earliest starting times of QCs. Constraints (7)–(9) are precedence constraints. If task i precedes task j, the completion of task i should be earlier than the beginning of task j, and the variables y_{ij} and y_{ji} can be determined. Constraints (10) and (11) guarantee that a QC cannot process two tasks simultaneously and have enough time to move from one task location to another task location. Constraints (12) and (13) are noncross constraints (a part of noninterference constraints). If $(i, j, k, l) \in \Theta$ (which means that a crossing between QC k and QC l takes place), then these constraints can help to avoid QC collisions by imposing a safety interval between task i and task j. Constraints (14) and (15) are safety margin constraints (another part of noninterference constraints). Two tasks, which locate too closely, should not have an overlap interval. Constraints (16)–(18) are variable range constraints.

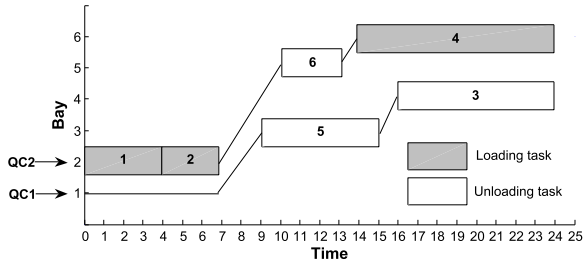


Fig. 3. Optimal solution to the QCSP instance.

The two-index formulation introduces less binary variables than the three-index formulation. It performs slightly faster than the three-index formulation by commercial IP solvers (e.g., ILOG Cplex). However, it is still difficult to solve on those middle and large size QCSP instances.

C. Stability Constraints

In practice, an important issue of the QCSP is to keep the ship stabilized during the unloading and/or loading process. Note that in contrast to stowage planning problems, the ship stability requirement should be satisfied during the whole process, not merely applied to the final layout of the ship.

Generally, ship stability consists of vertical stability, traverse stability, and longitudinal stability [47]. Important factors for measuring vertical stability and traverse stability are *metacentric height* (the distance between the center of gravity of a ship and its metacenter) and *angle of list* (the degree to which a ship is leaning to one side). Because the task is bay-based, it is implicitly assumed that vertical stability and traverse stability can be optimized during the process of a single task, e.g., loading from bottom to top, from middle to two sides in each bay. In this paper, we only focus on the longitudinal stability of the ship.

The longitudinal stability (see Fig. 4) is related to the *trim* value, which is defined as the difference between the drafts forward and aft. The trim value, denoted by t, is given in (19), where W is the weight of the empty ship, l is the length of the ship, w_i is the weight of container i, $x_i - x_0$ is the horizontal distance between the container i and the center of gravity, and GM_L is the distance between the center of gravity G and the longitudinal metacenter M_L. The approximated equation is obtained from $GM_L \simeq BM_L = ((W \cdot l^3)/(12(W + \sum_i w_i)))$ when the box-shaped ship is considered

$$t = \frac{l \sum_i w_i (x_i - x_0)}{\left(W + \sum_i w_i\right) \cdot GM_L} \simeq \frac{12}{W \cdot l^2} \sum_i w_i (x_i - x_0). \quad (19)$$

Trim by the stern is conventionally defined as positive. Thus, unloading near the stem or loading near the stern will increase the trim value. On the other hand, loading near the stem or unloading near the stern will decrease the trim value.

Based on the above discussion, we can incorporate the following constraints into the QCSP model to form the QCSPSC model.

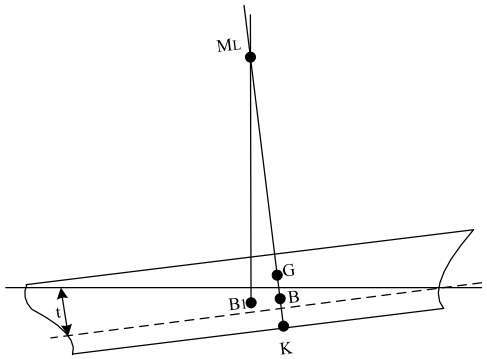


Fig. 4. Ship longitudinal stability factor (trim).

Notations:

Input Parameters:

d_i Contribution of trim value (per unit time) of task i.
H Stability threshold.
T_0 Upper bound on the scheduling horizon.

Sets:

T Set of the time horizon, i.e., $T = \{0, \ldots, T_0\}$.

Indices:

h, t Index of time horizon, $h, t \in T$.

Decision Variables:

u_{it} Binary variable which is equal to 1 if task i is being processed at time t, and 0 otherwise.

Constraints:

$$s_i \le t u_{it} + \mathcal{M}\,(1 - u_{it}), \quad i \in N,\ t \in T \quad (20)$$

$$t u_{it} \le c_i - 1, \quad i \in N,\ t \in T \quad (21)$$

$$\sum_{t \in T} u_{it} = \sum_{k \in Q} p_{ik} x_{ik}, \quad i \in N \quad (22)$$

$$\left| \sum_{h=0}^{t} \sum_{i \in N} d_i u_{ih} \right| \le H, \quad t \in T \quad (23)$$

$$u_{it} \in \{0, 1\}, \quad i \in N,\ t \in T. \quad (24)$$

Constraints (20)–(24) assure the ship stability within the threshold H during the whole scheduling process. A trivial upper bound T_0 can be set to $\max_{i \in N, k \in Q}\{t'_{ik}\} + (N - 1) \max_{i,j \in N}\{t_{ij}\} + \sum_{i \in N} (\max_{k \in Q} p_{ik})$, which approximates the longest time for one QC processing all the tasks. Constraints (20) and (21) state that if time t outsides the window $[s_i, c_i)$, then the variable u_{it} must equal 0. Otherwise, u_{it} is forced to equal 1 by Constraint (22). Constraint (23) is the stability constraint, where $\sum_{h=0}^{t} \sum_{i \in N} d_i u_{ih}$ corresponds to the trim of the ship at time t. Because Constraint (23) is nonlinear, we linearize them by the following two constraint sets:

$$\sum_{h=0}^{t} \sum_{i \in N} d_i u_{ih} \le H, \quad t \in T \quad (25)$$

$$\sum_{h=0}^{t} \sum_{i \in N} d_i u_{ih} \ge -H, \quad t \in T. \quad (26)$$

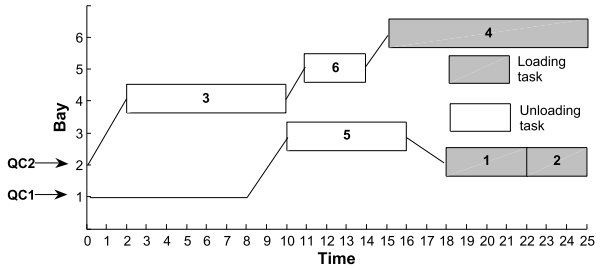


Fig. 5. Optimal solution to the QCSPSC instance.

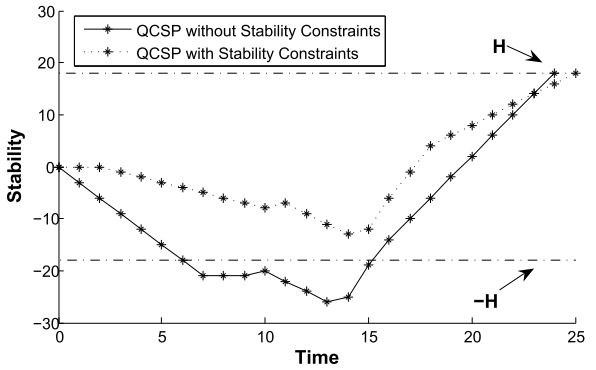


Fig. 6. Stability–time plot for the optimal solution of the QCSP with/without stability constraints.

We also take the instance given in Fig. 2 as a QCSPSC example. Assume that: 1) the ship is on even keel in initial; 2) the weight of containers in a task is proportional to the task processing time; 3) the horizontal positions of the six bays are set to $(-5, -3, -1, +1, +3, +5)$; and 4) the constant coefficient in (19) is neglected. Then, the contribution of trim values of the six tasks is $(-12, -9, -8, +50, +6, -9)$, and thus, the increment of trim is $+18$ in the final layout. If we require that the stability threshold H is also 18, then the optimal solution of the QCSPSC is 25 (see Fig. 5 for the detailed solution), which is 1 unit time larger than the optimal solution of the QCSP. The stability–time plot is illustrated in Fig. 6. It can be seen from the optimal solution of the QCSP without stability constraints that the ship is trimming by the stem severely in the early stage [the maximum absolute trim (MAT) value is 26], while the ship maintains much stabilized conditions from the optimal solution of the QCSPSC.

IV. Solution Approach

The QCSPSC is an NP-hard problem [27] and is generally much more difficult to solve than the QCSP. Moreover, the QCSPSC may even have no feasible solutions when the stability threshold is relatively small. Preliminary experiments show that the QCSPSC model is very difficult to solve by general IP solvers. To address the difficulty, we present a BiEA for seeking promising solutions of the QCSPSC.

A. Overview of the Approach

It is obvious that a feasible solution of the QCSPSC is also feasible to the QCSP, i.e., the feasible region of the QCSP encompasses the entire feasible solutions of the QCSPSC. The basic idea of our approach is to generate a set of solutions to the QCSP with relaxing the stability constraints, and then try to repair them to satisfy the stability requirements. To do so, we treat the QCSPSC as a bicriteria problem [48]. The first criterion is the minimization of the makespan, which is also the objective of the QCSPSC. The second criterion is the minimization of the MAT during the scheduling process, which guarantees the stability constraints.

TABLE I
Dominance Relations Between Two Solutions S_1 and S_2

	$f_m(S_1) < f_m(S_2)$	$f_m(S_1) = f_m(S_2)$	$f_m(S_1) > f_m(S_2)$
$f_t(S_1) = f_t(S_2) = H$	$\prec$	=	$\succ$
$f_t(S_1) = f_t(S_2) < H$	$\prec$	=	$\succ$
$H < f_t(S_1) = f_t(S_2)$	$\prec$	=	$\succ$
$f_t(S_1) < f_t(S_2) < H$	$\prec$	$\prec$	$\succ$
$f_t(S_1) < f_t(S_2) = H$	$\prec$	$\prec$	$\succ$
$f_t(S_2) < f_t(S_1) < H$	$\prec$	$\succ$	$\succ$
$f_t(S_2) < f_t(S_1) = H$	$\prec$	$\succ$	$\succ$
$f_t(S_2) < H < f_t(S_1)$	$\succ$	$\succ$	$\succ$
$f_t(S_2) = H < f_t(S_1)$	$\succ$	$\succ$	$\succ$
$H < f_t(S_2) < f_t(S_1)$	$\succ$	$\succ$	$\succ$
$f_t(S_1) < H < f_t(S_2)$	$\prec$	$\prec$	$\prec$
$f_t(S_1) = H < f_t(S_2)$	$\prec$	$\prec$	$\prec$
$H < f_t(S_1) < f_t(S_2)$	$\prec$	$\prec$	$\prec$

Formally, let $S \in \mathbb{S}$ be a feasible solution of the QCSP, where $\mathbb{S}$ is the solution space of the QCSP. We denote by $f_m(S)$ and $f_t(S)$ the makespan and the MAT with respect to the solution S, respectively. The QCSPSC can be simply restated as follows:

$$\min \ f_m(S) \tag{27}$$

$$\text{subject to } f_t(S) \leq H \tag{28}$$

$$S \in \mathbb{S}. \tag{29}$$

The bicriteria problem tries to minimize $f_m(S)$ and $f_t(S)$ simultaneously. Because our approach tries to generate many QCSP solutions, we need a mechanism to compare the quality of two solutions $S_1, S_2 \in \mathbb{S}$. The definition of the dominance relation is introduced as follows.

Definition 1: A solution $S_1 \in \mathbb{S}$ is said to (strictly) dominate another solution $S_2 \in \mathbb{S}$, denoted by $S_1 \prec S_2$ (or $S_2 \succ S_1$), if one of the following conditions hold:

1) $f_m(S_1) < f_m(S_2)$ and $f_t(S_1) \leq f_t(S_2)$;
2) $f_m(S_1) < f_m(S_2)$ and $f_t(S_2) \leq f_t(S_1) \leq H$;
3) $f_m(S_1) = f_m(S_2)$ and $f_t(S_1) < f_t(S_2)$;
4) $f_m(S_1) > f_m(S_2)$ and $f_t(S_1) \leq H < f_t(S_2)$;
5) $f_m(S_1) > f_m(S_2)$ and $H < f_t(S_1) < f_t(S_2)$.

The rationale behind the definition lies in the following observations: 1) feasible solutions are better than infeasible solutions; 2) feasible solutions with smaller objectives are preferred; and 3) infeasible solutions with better stability are more likely to become feasible after slightly perturbing the solution, and thus are preferred. Table I summarizes the dominance relations between two solutions with respect to different relations of f_m and f_t.

The dominance relations are very important in maintaining elite solutions during the evolutionary process. For solving

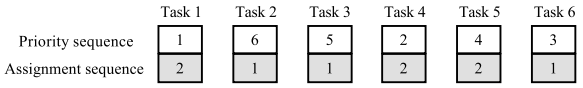


Fig. 7. Example of the encoding scheme.

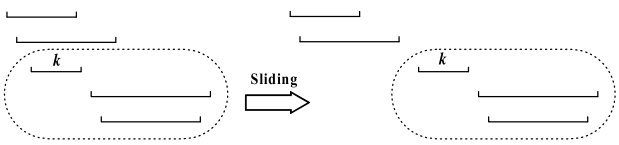


Fig. 8. Example of the sliding-window operation.

the QCSPSC, we adopt a relatively standard evolutionary framework. The evolutionary algorithm is a population-based metaheuristic method, which is able to seek optimal or near-optimal solutions in the solution space. Our algorithm (see Algorithm 1) consists of an initial population generation phase, a crossover, a local search, and a bicriteria-based selection procedure. The algorithm is essentially a hybrid genetic algorithm or memetic algorithm [49].

Algorithm 1: BiEA Framework for the QCSPSC

1: $P_0 \leftarrow$ INITIAL_POPULATION( );
2: $t \leftarrow 0$;
3: **while** termination criteria not reached **do**
4: $Q_t \leftarrow$ CROSSOVER(P_t);
5: $Q_t \leftarrow$ LOCAL-SEARCH(Q_t);
6: $P_{t+1} \leftarrow$ SELECTION(Q_t); ▷ Use bi-criteria domination
7: $t \leftarrow t + 1$;
8: **end while**

One main component in the evolutionary algorithm is the representation of individuals. Encoding individuals in different representation ways may have different impacts on the efficiency of evolutionary operators and convergence of the algorithm. In Section IV-B, we introduce a priority-assignment-based encoding scheme to express a solution of the QCSP without stability constraints. We then devise a sliding-window heuristic algorithm in Section IV-C aiming to repair the solution that violates the stability constraints. The detailed implementations of the algorithm are presented in Section IV-D.

B. Solution Encoding and Decoding

Similar to the previous work [28], we introduce the priority-assignment-based encoding scheme. For a given QCSP solution, we use a priority sequence to indicate the starting order of each task and use an assignment sequence to represent the assignment of tasks to QCs. Fig. 7 shows an example of the encoding scheme. In this example, "Task 1" is the first task to be processed by "QC 2," "Task 2" is the last task to be processed by "QC 1," and so on.

On the other hand, given the priority sequence (denoted by the array pr) and the assignment sequence (denoted by the array as), we need to decode them to obtain a QCSP solution. The decoding of an individual is essentially a greedy procedure that tries to assign tasks as early as possible. Note that the priority sequence may be invalid, as the precedence constraints have defined partial priorities for several tasks. To correct the priority sequence, a simple reparation procedure is carried out to swap the elements in the priority sequence. Specifically, if $(i, j) \in \Phi$ and $pr[i] > pr[j]$, swap the values of $pr[i]$ and $pr[j]$.

Then, we only need to consider the noninterference constraints in decoding. The decoding procedure is expressed in Algorithm 2. In this algorithm, $QC\_pos[i]$ means the current position of QC i, which is set to q_i in initial (line 3). $QC\_time[i]$ denotes the available starting time of QC i, which is set to 0 in the beginning (line 4). The order of the task in process must be in accordance with the priority sequence (line 7). Suppose that task k is to be processed by QC q (lines 7 and 8). The algorithm checks whether QC q crosses some other QC j (line 10). If so, QC q is not suitable for processing task k and task k is assigned to its closest QC (line 14). To process the task, the QC should move from its current position to the task location (line 16), incurring the QC traveling time (line 15). When the QC is in position, it may not be able to start due to potential violations of the safety margin constraints. The QC needs to wait for the completion of its interfering QCs (lines 17–20). Because the priority sequence requires increasing starting times for processing the corresponding tasks, $latest\_time$ records the latest starting time (lines 21 and 22) and $QC\_time[q]$ is updated accordingly (line 23).

Finally, we introduce a number of *activities* to represent a QCSP solution. An activity is a quad-tuple defined as an assignment of a task to a QC within a specified time period. Algorithm 2 can generate a feasible QCSP solution (line 26).

Algorithm 2 executes in $O(NQ)$ time. Thus, it is quite efficient in decoding an individual and at the same time, evaluating the solution (the makespan value).

C. Sliding-Window Heuristic Algorithm

The decoding procedure (Algorithm 2) guarantees to find a feasible QCSP solution with respect to the given individual. However, it appears that the solution may be infeasible to the QCSPSC. This is because the MAT [i.e., $f_t(S)$] may exceed the safety threshold. In order to repair the solution to make it satisfy the stability constraints, we devise a sliding-window heuristic algorithm.

As a QCSP solution is represented by a set of activities, the basic idea of the heuristic algorithm is to slide the time windows of some activities so as to reduce the MAT meanwhile maintaining the relative order of the tasks and the feasibility of the solution to the QCSP. To achieve the goal, we first sort the activities according to the increasing order of their starting times. Proposition 1 allows us to perform a type of sliding-window operation.

Proposition 1: For a set of N sorted activities and a given index k, simultaneously slide all the activities $j \geq k$ to the right by the same time units, the resultant solution is also feasible to the QCSP.

Algorithm 2: A Greedy Algorithm for Decoding an Individual

Input:
Assignment sequence array: $as[i], i \in N$;
Priority sequence array: $pr[i], i \in N$;
Task locations: $l_i, i \in N$;
Processing time: $p_{ik}, i \in N, k \in Q$;
Initial QC positions: $q_k, k \in Q$;
Output:
Activity array: $a[1], \ldots, a[N]$.

1: **for** $i \leftarrow 1$ to Q **do** ▷ set initial QC positions and times
2: $QC\_pos[i] \leftarrow q_i$;
3: $QC\_time[i] \leftarrow 0$;
4: **end for**
5: $latest\_time \leftarrow 0$;
6: **for** $i \leftarrow 1$ to N **do**
7: $k \leftarrow$ the index of task with $pr[i]$;
8: $q \leftarrow as[k]$;
9: **for** $j \leftarrow 1$ to Q **do**
10: **If** QC q crosses QC j **then**
11: $q' \leftarrow$ record the QC which is closest to task k;
12: **end for**
13: **If** there is some q' recorded **then**
14: $q \leftarrow q'$; ▷ let QC q' handle task k
15: $QC\_time[q] \leftarrow QC\_time[q]$ + travelling time between $QC\_pos[q]$ and l_k;
16: $QC\_pos[q] \leftarrow l_k$;
17: **for** $j \leftarrow 1$ to Q **do**
18: **If** (QC q is interfered with QC j) and $(QC\_time[j] > QC\_time[q])$ **then**
19: $QC\_time[q] \leftarrow QC\_time[j]$;
20: **end for**
21: **If** $QC\_time[q] > latest\_time$ **then**
22: $latest\_time \leftarrow QC\_time[q]$;
23: $QC\_time[q] \leftarrow latest\_time + p_{i,q}$;
24: $a[i] = \langle k, q, latest\_time, QC\_time[q] \rangle$;
25: **end for**
26: **return** a;

Fig. 8 illustrates an example of such sliding-window operation. The proof of Proposition 1 is straightforward. Because the relative time windows of activity group $1, 2, \ldots, k-1$ and relative time windows of activity group $k, k+1, \ldots, N$ remain the same, the precedence constraints and interference constraints are satisfied within the group. Moreover, the activities in the latter activity group are deferred, so they will not interfere with the activities in the former activity group.

The next step is to decide which activities to slide for reducing the MAT. To do so, we first find the activities which cross the timepoint to the MAT. Only sliding the time windows of one or multiple of these activities may reduce the MAT. Let us take Fig. 9(a) as an example. There are four activities in total. The number listed above the segment indicates the trim contribution of the corresponding task in a time period. The change of trim and trim value during the scheduling horizon is also presented. It can be observed that the first three activities cross the timepoint. Subsequently, we test the

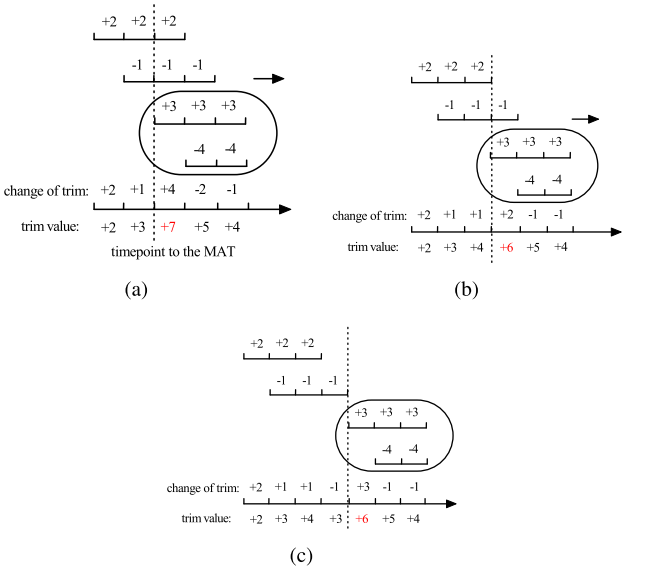


Fig. 9. Example of the sliding-window operation for reducing the MAT.

sign of trim contribution of these activities. If the sign is the same with the sign of trim value, sliding the time window of the corresponding activity will potentially reduce the MAT. In Fig. 9(a), the trim value corresponding to the timepoint is $+7$, and both the first and the third activities have plus trim contributions, the same sign with the trim value. If we perform the sliding-window operation with the index $k = 3$ for one time unit, we can obtain a QCSP solution shown in Fig. 9(b), where the MAT is decreased to 6.

It is worth noting that there are two cases that have no impacts on reducing the MAT: 1) performing the sliding-window operation with $k = 1$ and 2) sliding the activity which has already lied outside the time window of its preceding activities. Fig. 9(c) shows that the time window of the third activity is outside the time window of the first activity and the second activity.

Based on the above discussion, we propose a sliding-window heuristic algorithm (Algorithm 3). The algorithm first calculates the $change\_of\_trim$ array (lines 1–5) and the $trim$ array (lines 6–10). Next, the algorithm repeatedly finds the timepoint to the MAT (line 12) and record the corresponding trim value (line 13). If the MAT does not exceed the safety threshold H, a feasible QCSPSC solution is found (lines 14 and 15). Otherwise, the algorithm finds the activity which crosses the timepoint and has the same sign of trim contribution with the sign of max_tm, and then performs the sliding-window operation (lines 16–21).

The sliding-window operation is depicted in Algorithm 4. It simultaneously slides the time windows of activities $i, i+1, \ldots, N$ to the right by one time unit.

The sliding-window heuristic algorithm is guaranteed to terminate in finite steps, as the sliding-window operation will eventually separate all activities in finite steps. The worst case time complexity of the algorithm is $O(N \sum_{i=1}^{N} \max_{k \in Q} p_{ik})$.

D. Implementation of the Bicriteria Evolutionary Algorithm

The proposed BiEA to the QCSPSC combines a genetic algorithm with a local search procedure. Most components of

Algorithm 3 : Sliding-Window Heuristic Algorithm

Input:
A set of N activities, sorted by the starting times: $a[1], \ldots, a[N]$;
Trim contribution (per unit time) of each task: $tc[1], \ldots, tc[N]$;
Makespan: M.
Output:
Updated activities: $a[1], \ldots, a[N]$.

for $i \leftarrow 1$ to N **do**
  for $j \leftarrow a[i].start$ to $a[i].end - 1$ **do**
    $change\_of\_trim[j] \leftarrow change\_of\_trim[j] + tc[a[i].task]$;
  end for
end for
$tm \leftarrow 0$;
for $j \leftarrow 0$ to M **do**
  $tm \leftarrow tm + change\_of\_trim[j]$;
  $trim[j] \leftarrow tm$;
end for
while true **do**
  $k \leftarrow \arg\max\{|trim[j]| \mid j = 0, \ldots, M\}$;
  $max\_tm \leftarrow trim[k]$;
  If $|max\_tm| \leq |H|$ **then**
    return a; ▷ successfully fix the schedule
  for $i \leftarrow 2$ to N **do** ▷ the first activity is excluded
    if $(a[i].start \leq k)$ and $(a[i].end - 1 \geq k)$ and $(a[i].start < \min\{a[j].end \mid j < i\})$ and $(\text{sign}(tc[a[i].task]) = \text{sign}(max\_tm))$ **then**
      SLIDING_WINDOW$(trim, i, M)$;
      break ;
    end if
  end for
  If the function SLIDING_WINDOW( ) not called **then**
    return a; ▷ failed to fix the schedule
end while

Algorithm 4: SLIDING_WINDOW Procedure

$M \leftarrow M + 1$;
$trim[M] \leftarrow trim[M-1]$;
while $i \leq N$ **do**
  for $j \leftarrow a[i].start$ to $a[i].end - 1$ **do**
    $trim[j] \leftarrow trim[j] - tc[a[i].task]$;
  end for
  $a[i].start \leftarrow a[i].start + 1$;
  $a[i].end \leftarrow a[i].end + 1$;
  $i \leftarrow i + 1$;
end while

the algorithm are standard in the literature but designed in a problem-specific way.

1) Initial Population Generation: The initial population P_0 is made up of $|P|$ different individuals encoded by the priority-assignment-based scheme, where $|P|$ is the population size. For each individual, the priority sequence is a random generated permutation of $1, 2, \ldots, N$. The assignment sequence consists of N elements, each is a random integer between 1 and Q.

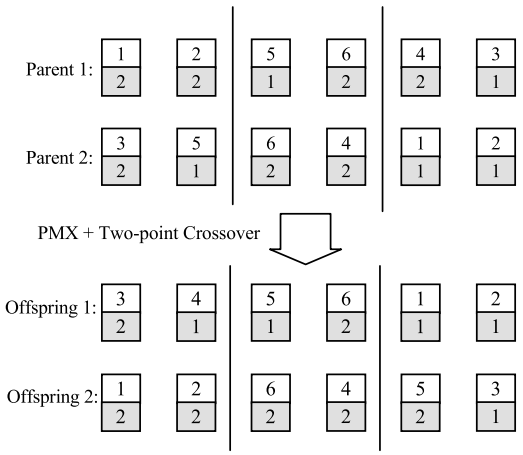


Fig. 10. Example of the two-point partially mapped crossover operator.

2) Crossover Operator: In the evolutionary algorithm, the crossover operator is used to produce new generations. In our algorithm, two parents are randomly selected and two offsprings are generated. A total of $|P|$ offsprings are produced finally.

Because an individual is represented by the combination of priority sequence and assignment sequence, it is impossible to directly apply the traditional crossover operators, e.g., two-point crossover, order-based crossover, and partially mapped crossover. To deal with this issue, we propose a two-point partially mapped crossover operator, which is similar to the one in [50].

The two-point partially mapped crossover operator is devised as follows: 1) two cut points in the range of 1 to N are randomly generated; 2) the two-point crossover to the generated cut points is performed on two parent assignment sequences, producing two offspring assignment sequences; and 3) the partially mapped crossover to the generated cut points is performed on two parent priority sequences, obtaining offspring priority sequences. Fig. 10 shows an example of the operator. For an individual, the blocks in white constitute the priority sequence, while the blocks in dark compose the assignment sequence.

3) Local Search Procedure: After the crossover operator, the local search operators are introduced to locally improve the quality of the generated offsprings. In our algorithm, we simply adopt the *exchange* operator and the *reverse* operator. Because an individual is encoded by the priority sequence and the assignment sequence, there are a total of four combinations for different operators on different sequences. One combination is randomly chosen, and then, the corresponding operation is performed. The local search procedure executes in $Iter$ iterations for each offspring, where $Iter$ is a user-defined parameter.

Note that both the priority sequence and the assignment sequence have N elements. The *exchange* operator randomly generates two indices x and y, such that $1 \leq x < y \leq N$, and then swap the corresponding elements. Similarly, the *reverse* operator reverses the segment of elements between x and y.

TABLE II
COMPARED ALGORITHMS ON THE QCSP INSTANCES

Literature	Year	Algorithm	Machine configuration
[24]	2004	B&B	Pentium 2 466 MHz, 64 MB RAM
[24]	2004	GRASP	Pentium 2 466 MHz, 64 MB RAM
[25]	2006	B&C	Pentium 4 2.5 GHz, 512 MB RAM
[26]	2007	TS	Pentium 4 2.66 GHz, 256 MB RAM
[27]	2009	UDS	Pentium 4 2.8 GHz, memory not provided
[28]	2012	GA	Intel Core 2 Quad 2 GHz, 2 GB RAM
[32]	2012	LTM	Intel T9900 3.07 GHz, memory not provided
-	-	BiEA	Intel Xeon 2.66 GHz, 8 GB RAM

TABLE III
PARAMETER SETTING FOR BIEA

Parameter	Value
Population size ($\|P\|$)	100
Recombination probability (P_r)	1.00
Mutation probability (P_m)	0.05
Local search iteration ($Iter$)	1000
Tournament size (τ)	50
No. generations (G)	1000

4) Selection Procedure and Termination Criterion: In our evolutionary algorithm, the tournament selection is applied to preserve elite individuals. The selection procedure involves running $|P|$ tournaments. In each tournament, a number of individuals (determined by the tournament size τ) in the population are chosen and compete with each other based on the dominance relations mentioned in Section IV-A. The winner of each tournament is selected for the next generation.

The algorithm terminates when the number of generations achieves a predefined value G.

V. COMPUTATIONAL EXPERIMENTS

To test the performance of the proposed BiEA, we conducted experiments on a series of test instances. The code was implemented in GNU C++, using the "−O2" option. All experiments were run on a Linux server with Intel Xeon E5430 2.66-GHz processor and 8-GB RAM.

A. Experimental Setup

The QCSPSC instances can be constructed by modifying the traditional QCSP instances. Originally, Kim and Park [24] proposed a set of QCSP benchmark instances, among which 37 small-scale instances (k13–k49) have been widely studied by researchers (e.g., [24]–[26], [28]). These instances are classified into four groups. Each group has an identical number of tasks and QCs. The remaining instances (k50–k102) in six groups are large-scale instances, which have been used by Bierwirth and Meisel [27] and Legato *et al.* [32].

Our experiments consisted of two parts. In the first part, we ran BiEA on the QCSP instances. This could be achieved by simply setting the safety threshold H to infinity. In order to evaluate the algorithm, we compared the results obtained by our algorithm with other algorithms. Table II summarizes different compared algorithms with their machine configurations, where B&B, GRASP, B&C, TS, UDS, GA, and LTM correspond to branch-and-bound, GRASP, branch-and-cut, tabu search, unidirectional scheduling heuristic, genetic algorithm, and the hybrid approach by Legato *et al.* [32], respectively. The comparison results are described in Section V-B.

The second part of the experiments assessed the quality of the solutions by BiEA on the QCSPSC instances. To construct QCSPSC instances from the QCSP instances, we followed the manner presented in the end of Section III-C and restated as follows: 1) set the ship on even keel in initial; 2) the total weight of containers in a task is set to the task processing time; 3) if an instance contains B bays, the horizontal positions of the ith bay (numbered from stem to stern) are set to $2i-B-1$; 4) the constant coefficient in (19) is neglected; and 5) the stability threshold is set to the absolute increment of trim in the final layout. The detailed results and analysis are presented in Section V-C.

TABLE IV
RESULTS OF THE QCSP WITHOUT STABILITY CONSTRAINTS ON SMALL-SCALE INSTANCES

Instance	GA	UDS	TS	B&C	GRASP	B&B	BiEA
Group A ($N=10, Q=2$):							
k13	453	453	453	453	453	453	453
k14	546	546	546	546	546	546	546
k15	513	513	513	513	516	513	513
k16	312	312	312	312	321	321	312
k17	453	453	453	453	456	456	453
k18	375	375	375	375	375	375	375
k19	543	543	543	543	552	552	543
k20	399	399	399	399	480	480	399
k21	465	465	465	465	465	465	465
k22	537	540*	537	537	720	720	537
Group B ($N=15, Q=2$):							
k23	576	576	582	576	591	576	576
k24	669	666	669	666	675	669	666
k25	744	738	741	738	741	738	738
k26	645	639	639	639	651	639	639
k27	660	657	657	660	687	657	657
k28	531	531	531	531	549	537	531
k29	810	807	810	807	819	807	807
k30	897	891	891	891	906	891	891
k31	570	570	570	570	570	570	570
k32	594	591	591	591	597	591	591
Group C ($N=20, Q=3$):							
k33	603	603	603	603	666	603	603
k34	717	717	735	717	762	717	717
k35	690	684	690	684	699	690	684
k36	*636*	678	681	678	708	720	678
k37	522	510	519	510	540	516	510
k38	618	618	618	618	660	633	618
k39	519	513	519	513	579	552	513
k40	567	564	567	564	597	576	567
k41	588	588	594	588	642	654	588
k42	576	573*	576	570	666	588	570
Group D ($N=25, Q=3$):							
k43	897	876	879	897	942	951	876
k44	855	822	834	822	858	879	822
k45	864	834	852	840	873	861	840
k46	723	690	690	690	735	708	690
k47	819	792	792	792	807	912	792
k48	663	639	663	645	669	669	639
k49	915	894	912	927	972	915	900
Average	623.4	617.8	621.6	619.5	649.9	640.5	618.1

TABLE V
SUMMARIZED RESULTS OF THE QCSP WITHOUT STABILITY CONSTRAINTS ON LARGE-SCALE INSTANCES

Group	Instance	UDS	LTM	BiEA
D $(N = 25, Q = 3)$	k50–k52	833.0	833.0	835.0
E $(N = 30, Q = 4)$	k53–k62	730.2	730.2	733.2
F $(N = 35, Q = 4)$	k63–k72	863.7	863.7	871.2
G $(N = 40, Q = 5)$	k73–k82	753.0	753.0	768.3
H $(N = 45, Q = 5)$	k83–k92	889.8	889.8	917.4
I $(N = 50, Q = 6)$	k93–k102	817.8	812.1	843.0
Average		812.2	811.1	827.2

After some preliminary experiments, the final parameters with respect to BiEA are listed in Table III. In fact, there is still room for fine-tuning the parameters on different instance groups, but we find that such setting is already appropriate in seeking promising QCSP and QCSPSC solutions.

B. Numerical Results for the QCSP

The results of 37 small-scale instances by different algorithms are reported in Table IV. The value under the second column to the eighth column indicates the best solution found by the corresponding algorithm. The LTM approach did not provide the detailed results of instances k13–k49, so its results are not shown.

We can observe from Table IV that the BiEA outperforms other approaches except for UDS in terms of the solution quality. It is worth noting that GA attained the smallest value among all the approaches on instance k36. However, such a value is incorrect as it is below the lower bound generated by other approaches (see [25]). The UDS approach takes the lead in most of the instances, but it obtained slightly larger results on k22 and k42 (marked with asterisks) than B&C and BiEA did. This is because the interference constraints defined by UDS require that QCs have to keep a safety margin at any time, while other approaches assumed that QCs have to keep a safety margin only if they are processing tasks. More discussions on interference constraints can be found in [27]. Overall, the BiEA achieves 35 out of 37 best known solutions on these instances. These results imply that the BiEA is a competitive approach, although it is not specifically designed for solving the QCSP.

We also tried to solve the large-scale QCSP instances using the BiEA. The results are compared with two leading approaches (i.e., UDS and LTM); they are summarized in Table V. Table V shows that the results obtained by the BiEA on k50–k102 are inferior to those given by UDS or LTM. The average gap is 1.81% and 1.94% to UDS and LTM, respectively.

We do not report the computational times in Tables IV and V. This is because different algorithms were executed on different machines and different random seeds potentially affected running times for those metaheuristic algorithms. In fact, all the solutions were obtained by the corresponding algorithms in reasonable computational times (less than 1 h).

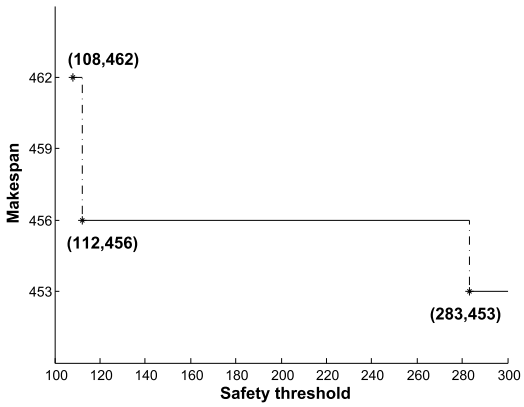


Fig. 11. Relations between safety threshold and makespan on instance k13.

C. Numerical Results for the QCSPSC

We ran our BiEA approach on the QCSPSC instances. For each instance, ten independent runs with different random seeds were carried out. Because previous works (e.g., [44] and [45]) cannot be directly applied for solving our QCSPSC, and preliminary experiments show that ILOG Cplex is not able to solve the IP model on any QCSPSC instances, we only compare the QCSPSC results generated by the BiEA with the QCSP results. Obviously, the QCSP solution can provide a lower bound for the corresponding QCSPSC solution.

The results on small-scale instances (k13–k49) are presented in Table VI. In Table VI, the number under the heading "$|H_{\text{final}}|$" is the absolute trim value in the final layout with respect to the corresponding instance. The constructed QCSPSC instance requires that the trim values during the scheduling horizon should not exceed $|H_{\text{final}}|$, so the columns with the heading "BiEA ($H = |H_{\text{final}}|$)" correspond to the results of the QCSPSC, while the columns with the heading "BiEA ($H \to +\infty$)" correspond to the results of the QCSP without stability constraints. The value under the heading "trim" is the actual trim value to the MAT (can be positive or negative). If the absolute of the value is larger than $|H_{\text{final}}|$, an asterisk ("*") is marked beside the value indicating that the solution is infeasible to the QCSPSC. The columns "min," "max," and "avg" represent the minimum, maximum, and average makespan of the ten solutions, respectively. The column "time(s)" gives for the average running time (in seconds) of the ten runs. The column "diff." calculates the relative difference between the minimum values obtained for the QCSP and the QCSPSC.

Table VI shows that originally, there are 19 solutions to the QCSP, which are also feasible to the QCSPSC. By considering the stability constraints, BiEA is able to find 35 feasible solutions to the QCSPSC. Only the instances k23 and k33 have no feasible solutions found. The optimal solution to the QCSP provides a lower bound for the QCSPSC. On average, the best found solution to the QCSPSC has an increase of 1% over the best found solution to the QCSP. Besides, the computational times do not increase significantly. These facts demonstrate the effectiveness of the proposed algorithm: ensuring the ship stability leads to only a small increase of the makespan.

TABLE VI
RESULTS OF THE QCSPSC ON SMALL-SCALE INSTANCES

Instance	$\|H_{final}\|$	BiEA ($H \to +\infty$)					BiEA ($H = \|H_{final}\|$)					
		min	max	avg	time(s)	trim	min	max	avg	time(s)	trim	diff.
Group A ($N = 10, Q = 2$):												
k13	108	453	453	453.0	209.8	-283*	462	462	462.0	223.5	-108	1.95%
k14	1473	546	546	546.0	264.7	-1473	546	546	546.0	259.0	-1473	0.00%
k15	325	513	513	513.0	210.6	-325	513	513	513.0	265.6	-325	0.00%
k16	661	312	312	312.0	195.6	661	312	312	312.0	196.5	661	0.00%
k17	865	453	453	453.0	228.8	865	453	453	453.0	231.5	865	0.00%
k18	875	375	375	375.0	231.6	-907*	378	378	378.0	247.4	-875	0.79%
k19	206	543	543	543.0	234.4	-290*	558	558	558.0	267.5	-206	2.69%
k20	387	399	399	399.0	228.7	-387	399	399	399.0	218.2	-387	0.00%
k21	90	465	465	465.0	204.8	-185*	471	471	471.0	279.3	-90	1.27%
k22	504	537	537	537.0	240.8	-504	537	537	537.0	252.5	-504	0.00%
Group B ($N = 15, Q = 2$):												
k23	76	576	576	576.0	308.2	368*	585	597	590.7	381.3	292*	1.54%
k24	602	666	666	666.0	348.6	688*	669	669	669.0	409.8	-602	0.45%
k25	1594	738	738	738.0	399.9	1594	738	738	738.0	421.9	1594	0.00%
k26	1046	639	639	639.0	369.0	1166*	642	642	642.0	390.5	1046	0.47%
k27	782	657	657	657.0	341.0	-782	657	657	657.0	379.0	-782	0.00%
k28	1826	531	531	531.0	339.2	1826	531	531	531.0	364.4	1826	0.00%
k29	624	807	810	807.3	374.7	1118*	810	810	810.0	467.2	624	0.37%
k30	1024	891	891	891.0	427.4	-1704*	936	936	936.0	631.8	-1024	4.81%
k31	972	570	570	570.0	359.9	972	570	570	570.0	373.8	972	0.00%
k32	612	591	591	591.0	345.1	612	591	591	591.0	366.9	612	0.00%
Group C ($N = 20, Q = 3$):												
k33	78	603	603	603.0	562.8	-863*	639	648	643.7	879.2	193*	5.63%
k34	1991	717	729	721.7	600.2	1991	723	729	726.0	743.5	1991	0.83%
k35	466	684	684	684.0	572.6	-904*	690	699	695.0	993.8	466	0.87%
k36	836	678	678	678.0	594.2	-1450*	681	690	684.3	805.3	836	0.44%
k37	179	510	510	510.0	522.1	802*	531	555	536.3	720.7	179	3.95%
k38	594	618	618	618.0	556.1	658*	621	633	626.3	829.6	594	0.48%
k39	1901	513	513	513.0	541.1	1901	513	513	513.0	550.8	1901	0.00%
k40	769	567	567	567.0	536.0	-769	567	567	567.0	690.4	-769	0.00%
k41	498	588	588	588.0	549.6	545*	597	597	597.0	731.9	498	1.51%
k42	3413	570	570	570.0	562.9	-3413	570	573	572.7	611.8	-3413	0.00%
Group D ($N = 25, Q = 3$):												
k43	1490	876	879	877.7	769.8	2022*	885	894	890.0	986.3	1490	1.02%
k44	3012	822	831	824.3	752.4	-3012	831	846	838.7	956.3	-3012	1.08%
k45	1900	840	852	845.3	724.3	-2020*	846	852	848.7	1002.7	-1900	0.71%
k46	2760	690	690	690.0	695.9	2760	696	708	703.0	745.4	2760	0.86%
k47	5500	792	801	793.0	731.8	5510*	801	807	804.0	926.3	5500	1.12%
k48	2500	639	666	646.7	648.1	2500	651	666	657.3	749.9	2500	1.84%
k49	1930	900	906	901.3	778.8	-1930	900	912	909.3	1110.2	-1930	0.00%
Average		618.1	620.3	618.7	447.6		624.3	628.6	626.4	558.4		1.00%

TABLE VII
SELECTED RESULTS OF THE QCSPSC ON LARGE-SCALE INSTANCES

Group	Instance	$\|H_{final}\|$	BiEA ($H \to +\infty$)		BiEA ($H = \|H_{final}\|$)	
			makespan	trim	makespan	trim
E	k55	493	696	642*	699(↑0.4%)	493(↓23.2%)
E	k58	433	789	-537*	798(↑1.1%)	-433(↓19.4%)
G	k73	272	891	977*	915(↑2.6%)	-495*(↓49.3%)
G	k74	794	858	2586*	879(↑2.4%)	-794(↓69.3%)
G	k77	1416	714	6016*	729(↑2.1%)	1416(↓76.5%)
G	k79	929	759	1351*	780(↑2.7%)	929(↓31.2%)
H	k91	1022	867	1242*	891(↑2.7%)	1022(↓17.7%)
I	k93	158	843	1716*	879(↑4.1%)	619*(↓63.9%)
I	k95	1263	849	1628*	867(↑2.1%)	-1263(↓22.4%)
I	k98	3598	762	-3635*	801(↑4.9%)	-3598(↓1.0%)
I	k99	7761	867	7837*	891(↑2.7%)	7761(↓1.0%)
I	k102	3020	921	3282*	954(↑3.5%)	3020(↓8.0%)

Table VII gives the selected results of the QCSPSC on large-scale instances (k50–k102). If the QCSP solution found is already feasible to the QCSPSC, the corresponding result is not selected. Finally, there are a total of 12 instances selected. We take instance k58 as an example. When the stability constraints are not considered, a solution was found with its makespan and stability equal to 789 and -537, respectively. This solution is infeasible to the QCSPSC, because $|-537| > |H_{\text{final}}| = 433$. If the stability constraints are considered, a feasible solution can be obtained. The makespan of the solution increases by 1.1% and the MAT decreases by 19.4%. Still, BiEA cannot find any feasible QCSPSC solutions on instances k73 and k93. The results suffice to demonstrate that by considering the stability constraints, the ship will become stable, but the turnaround time will not be affected significantly.

We also conduct additional experiments on the instance k13. By increasing the safety threshold H from $|H_{\text{final}}| = 108$ to infinity, the optimal makespan found by the algorithm decreases from 462 to 453. We plot the relations between safety threshold and makespan, as shown in Fig. 11. The plot can provide a Pareto front for users to trade off the makespan and the safety threshold between the solutions.

VI. Conclusion

The QCSP is an important and a well-studied problem in the operations of container terminals. It involves the task precedence constraints and QC noninterference constraints to ensure safe crane movements. In this paper, we extend the QCSP by taking into consideration the ship stability constraints, which are vital for the safety of ships. We formulate the QCSPSC into a mixed integer linear programming model. As the model is generally difficult to solve by integer programming solvers, a BiEA is proposed for finding promising solutions of the QCSPSC. One major contribution of this paper is the introduction of the sliding-window heuristic for fixing the schedule which violates the stability constraints. Computational results on the modified benchmark data set show that the proposed algorithm can produce high quality solutions, which are averagely about 1% greater than the best solutions of the QCSP.

In the literature, there exists several versions of the QCSP [18]. For example, the QCSP with complete bays requires that each bay is exclusively served by one QC, while the QCSP with bay areas enables the cranes to share the workload of bays. Our optimization techniques (e.g., the bicriteria method and the sliding-window heuristic) can potentially be extended to deal with different versions of the QCSPSCs, given that a proper heuristic method has been designed for generating feasible QCSP solutions.

In practice, the scheduling of QCs is a complex process subject to many constraints. Ship stability is one type of constraints that is close to the realistic logistics environment, so it deserves our attention. Ship stability generally consists of vertical stability, traverse stability, and longitudinal stability. This paper only focuses on ensuring the longitudinal stability. The vertical stability and traverse stability are implicitly guaranteed during the process of a single task. To extend this paper, future research can be conducted in ensuring more dimensions of the ship stability.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments. They would also would like to thank Prof. Y.-M. Park for providing the QCSP instances.

References

[1] *Review of Maritime Transport 2014*, UNCTAD, Geneva, Switzerland, 2014.

[2] R. Stahlbock and S. Voß, "Operations research at container terminals: A literature update," *OR Spectr.*, vol. 30, no. 1, pp. 1–52, 2008.

[3] C.-I. Liu, H. Jula, and P. A. Ioannou, "Design, simulation, and evaluation of automated container terminals," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 1, pp. 12–26, Mar. 2002.

[4] K. G. Murty, J. Liu, Y.-W. Wan, and R. Linn, "A decision support system for operations in a container terminal," *Decision Support Syst.*, vol. 39, no. 3, pp. 309–332, 2005.

[5] M. P. Fanti, G. Stecco, and W. Ukovich, "Scheduling internal operations in post-distribution cross docking systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 1, pp. 296–312, Jan. 2016.

[6] C. Bierwirth and F. Meisel, "A follow-up survey of berth allocation and quay crane scheduling problems in container terminals," *Eur. J. Oper. Res.*, vol. 244, no. 3, pp. 675–689, 2015.

[7] P. Chen, Z. Fu, A. Lim, and B. Rodrigues, "Port yard storage optimization," *IEEE Trans. Autom. Sci. Eng.*, vol. 1, no. 1, pp. 26–37, Jul. 2004.

[8] L. Zhen, L. H. Lee, E. P. Chew, D.-F. Chang, and Z.-X. Xu, "A comparative study on two types of automated container terminal systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 1, pp. 56–69, Jan. 2012.

[9] C.-Y. Lee, M. Liu, and C. Chu, "Optimal algorithm for the general quay crane double-cycling problem," *Transp. Sci.*, vol. 49, no. 4, pp. 957–967, 2014, doi: 10.1287/trsc.2014.0563.

[10] M. Liu, F. Chu, Z. Zhang, and C. Chu, "A polynomial-time heuristic for the quay crane double-cycling problem with internal-reshuffling operations," *Transp. Res. E, Logistics Transp. Rev.*, vol. 81, pp. 52–74, Sep. 2015.

[11] W. Zhu, H. Qin, A. Lim, and H. Zhang, "Iterative deepening A* algorithms for the container relocation problem," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 4, pp. 710–722, Oct. 2012.

[12] Q. Meng, S. Wang, H. Andersson, and K. Thun, "Containership routing and scheduling in liner shipping: Overview and future research directions," *Transp. Sci.*, vol. 48, no. 2, pp. 265–280, 2013.

[13] Q. Meng and S. Wang, "Liner shipping service network design with empty container repositioning," *Transp. Res. E, Logistics Transp. Rev.*, vol. 47, no. 5, pp. 695–708, 2011.

[14] S. Wang and Q. Meng, "Liner ship fleet deployment with container transshipment operations," *Transp. Res. E, Logistics Transp. Rev.*, vol. 48, no. 2, pp. 470–484, 2012.

[15] Q. Meng, S. Wang, and C.-Y. Lee, "A tailored branch-and-price approach for a joint tramp ship routing and bunkering problem," *Transp. Res. B, Methodol.*, vol. 72, pp. 1–19, Feb. 2015.

[16] J.-X. Dong, C.-Y. Lee, and D.-P. Song, "Joint service capacity planning and dynamic container routing in shipping network with uncertain demands," *Transp. Res. B, Methodol.*, vol. 78, pp. 404–421, Aug. 2015.

[17] D. Steenken, S. Voß, and R. Stahlbock, "Container terminal operation and operations research—A classification and literature review," *OR Spectr.*, vol. 26, no. 1, pp. 3–49, 2004.

[18] C. Bierwirth and F. Meisel, "A survey of berth allocation and quay crane scheduling problems in container terminals," *Eur. J. Oper. Res.*, vol. 202, no. 3, pp. 615–627, 2010.

[19] C. F. Daganzo, "The crane scheduling problem," *Transp. Res. B, Methodol.*, vol. 23, no. 3, pp. 159–175, 1989.

[20] R. I. Peterkofsky and C. F. Daganzo, "A branch and bound solution method for the crane scheduling problem," *Transp. Res. B, Methodol.*, vol. 24, no. 3, pp. 159–172, 1990.

[21] A. Lim, B. Rodrigues, F. Xiao, and Y. Zhu, "Crane scheduling with spatial constraints," *Naval Res. Logistics*, vol. 51, no. 3, pp. 386–406, 2004.

[22] D.-H. Lee, H. Q. Wang, and L. Miao, "Quay crane scheduling with non-interference constraints in port container terminals," *Transp. Res. E, Logistics Transp. Rev.*, vol. 44, no. 1, pp. 124–135, 2008.

[23] W. C. Ng, "Crane scheduling in container yards with inter-crane interference," *Eur. J. Oper. Res.*, vol. 164, no. 1, pp. 64–78, 2005.

[24] K. H. Kim and Y.-M. Park, "A crane scheduling method for port container terminals," *Eur. J. Oper. Res.*, vol. 156, no. 3, pp. 752–768, 2004.

[25] L. Moccia, J.-F. Cordeau, M. Gaudioso, and G. Laporte, "A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal," *Naval Res. Logistics*, vol. 53, no. 1, pp. 45–59, 2006.

[26] M. Sammarra, J.-F. Cordeau, G. Laporte, and M. F. Monaco, "A tabu search heuristic for the quay crane scheduling problem," *J. Scheduling*, vol. 10, nos. 4–5, pp. 327–336, 2007.

[27] C. Bierwirth and F. Meisel, "A fast heuristic for quay crane scheduling with interference constraints," *J. Scheduling*, vol. 12, no. 4, pp. 345–360, 2009.

[28] S. H. Chung and K. L. Choy, "A modified genetic algorithm for quay crane scheduling operations," *Expert Syst. Appl.*, vol. 39, no. 4, pp. 4213–4221, 2012.

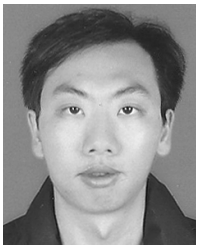
[29] R. Tavakkoli-Moghaddam, A. Makui, S. Salahi, M. Bazzazi, and F. Taheri, "An efficient algorithm for solving a new mathematical model for a quay crane scheduling problem in container ports," *Comput. Ind. Eng.*, vol. 56, no. 1, pp. 241–248, 2009.

[30] P. Legato and R. Trunfio, "A local branching-based algorithm for the quay crane scheduling problem under unidirectional schedules," *4OR*, vol. 12, no. 2, pp. 123–156, 2014.

[31] J. Liu, Y.-W. Wan, and L. Wang, "Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures," *Naval Res. Logistics*, vol. 53, no. 1, pp. 60–74, 2006.

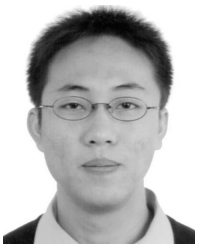
[32] P. Legato, R. Trunfio, and F. Meisel, "Modeling and solving rich quay crane scheduling problems," *Comput. Oper. Res.*, vol. 39, no. 9, pp. 2063–2078, 2012.

[33] F. Meisel, "The quay crane scheduling problem with time windows," *Naval Res. Logistics*, vol. 58, no. 7, pp. 619–636, 2011.
[34] F. Meisel and C. Bierwirth, "A unified approach for the evaluation of quay crane scheduling models and algorithms," *Comput. Oper. Res.*, vol. 38, no. 3, pp. 683–693, 2011.
[35] A. V. Goodchild and C. F. Daganzo, "Double-cycling strategies for container ships and their effect on ship loading and unloading operations," *Transp. Sci.*, vol. 40, no. 4, pp. 473–483, 2006.
[36] H. Zhang and K. H. Kim, "Maximizing the number of dual-cycle operations of quay cranes in container terminals," *Comput. Ind. Eng.*, vol. 56, no. 3, pp. 979–992, 2009.
[37] F. Meisel and W. Matthias, "Container sequencing for quay cranes with internal reshuffles," *OR Spectr.*, vol. 32, no. 3, pp. 569–591, 2010.
[38] A. Imai, K. Sasaki, E. Nishimura, and S. Papadimitriou, "Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks," *Eur. J. Oper. Res.*, vol. 171, no. 2, pp. 373–389, 2006.
[39] M. Y. H. Low, M. Zeng, W. J. Hsu, S. Y. Huang, F. Liu, and C. A. Win, "Improving safety and stability of large containerships in automated stowage planning," *IEEE Syst. J.*, vol. 5, no. 1, pp. 50–60, Mar. 2011.
[40] P. H. Hernández *et al.*, "An ant colony algorithm for improving ship stability in the containership stowage problem," in *Advances in Soft Computing and Its Applications*. Berlin, Germany: Springer, 2013, pp. 93–104.
[41] Z. Zhang and C.-Y. Lee, "Multiobjective approaches for the ship stowage planning problem considering ship stability and container rehandles," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 10, pp. 1374–1389, Oct. 2016.
[42] J. Christensen and D. Pacino, "A matheuristic for the cargo mix problem with block stowage," *Transp. Res. E, Logistics Transp. Rev.*, vol. 97, pp. 151–171, Jan. 2017.
[43] J. Wang, H. Hu, and Y. Song, "Optimization of quay crane scheduling constrained by stability of vessels," *Transp. Res. Rec. J. Transp. Res. Board*, vol. 2330, pp. 47–54, Aug. 2013.
[44] N. Al-Dhaheri, A. Jebali, and A. Diabat, "The quay crane scheduling problem with nonzero crane repositioning time and vessel stability constraints," *Comput. Ind. Eng.*, vol. 94, pp. 230–244, Apr. 2016.
[45] N. Al-Dhaheri and A. Diabat, "A Lagrangian relaxation-based heuristic for the multi-ship quay crane scheduling problem with ship stability constraints," *Ann. Oper. Res.*, vol. 248, nos. 1–2, pp. 1–24, 2017.
[46] E. Ursavas, "Crane allocation with stability considerations," *Maritime Econ. Logistics*, vol. 19, no. 2, pp. 379–401, 2017.
[47] B. Barrass and C. D. R. Derrett, *Ship Stability for Masters and Mates*. London, U.K.: Butterworth, 2011.
[48] R. B. Shepard, "Multi-objective, multi-criteria decision-making," in *Quantifying Environmental Impact Assessments Using Fuzzy Logic* (Springer Series on Environmental Management). New York, NY, USA: Springer, 2005, pp. 167–177.
[49] P. Moscato and C. Cotta, "A gentle introduction to memetic algorithms," in *Handbook of Metaheuristics*. Boston, MA, USA: Springer, 2003, pp. 105–144.
[50] H. Qin, Z. Zhang, Z. Qi, and A. Lim, "The freight consolidation and containerization problem," *Eur. J. Oper. Res.*, vol. 234, no. 1, pp. 37–48, 2014.



Zizhen Zhang received the B.S. and M.S. degrees from the Department of Computer Science, Sun Yat-sen University, Guangzhou, China, in 2007 and 2009, respectively, and the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2014.

He is currently an Associate Professor with Sun Yat-sen University. His research interests include computational intelligence and its applications in production, transportation, and logistics.



Ming Liu received the Ph.D. degree in industrial engineering from the Ecole Centrale Paris, Châtenay-Malabry, France, in 2009, and the Ph.D. degree in management science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2010.

He is currently an Associate Professor with Tongji University, Shanghai, China. His research interests include the problems of machine scheduling and container terminal logistics.

Chung-Yee Lee received the B.S. degree in electronic engineering and the M.S. degree in management sciences from National Chiao Tung University, Hsinchu, Taiwan, in 1972 and 1976, respectively, the M.S. degree in industrial engineering from Northwestern University, Evanston, IL, USA, in 1980, and the Ph.D. degree in operations research from Yale University, New Haven, CT, USA, in 1984.

He is currently a Chair Professor and a Cheong Ying Chan Professor of Engineering with the Department of Industrial Engineering and Logistics Management, The Hong Kong University of Science and Technology (HKUST), Hong Kong, where he served as the Department Head from 2001 to 2008. He is the Founding and Current Director of the Logistics and Supply Chain Management Institute, HKUST. He has published more than 150 papers in refereed journals. His current research interests include logistics and supply chain management, scheduling, and inventory management.

Dr. Lee is a Fellow of the Institute of Industrial Engineers, USA, and the Hong Kong Academy of Engineering Sciences.

Jiahai Wang (M'07) received the Ph.D. degree from Toyama University, Toyama, Japan, in 2005.

In 2005, he joined Sun Yat-Sen University, Guangzhou, China, where he is currently an Associate Professor. His main research interests include computational intelligence and its applications.