Multiship Crane Sequencing with Yard Congestion Constraints

Multi-ship Crane Sequencing with Yard Congestion Constraints

Shawn Choo
PSA International Pte Ltd, PSA Singapore Terminals, Singapore (shawnc@psa.com.sg)

Diego Klabjan
Department of Industrial and Management Sciences, Northwestern University, Evanston, IL (d-klabjan@northwestern.edu)

David Simchi-Levi
Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, Cambridge, MA (dslevi@mit.edu)

Crane sequencing in container terminals determines the order of ship discharging and loading jobs that quay cranes (QCs) perform so that the duration of vessels' stay is minimized. The ship's load profile, berthing time, number of available bays and QCs are considered. More importantly, clearance and yard congestion constraints need to be included, which respectively ensure that a minimum distance between adjacent QCs is observed and yard storage blocks are not overly accessed at any point in time. In sequencing for a single ship, a mixed integer programming model is proposed and a heuristic approach based on the model is developed that produces good solutions. The model is then reformulated as a generalized set covering problem and solved exactly by branch-and-price. For multi-ship sequencing, the yard congestion constraints are relaxed in the spirit of Lagrangian relaxation so that the problem decomposes by vessel into smaller sub-problems solved by branch-and-price. An efficient primal heuristic is also designed. Computational experiments reveal that large-scale problems can be solved in a reasonable computational time.

---

1. Introduction

A container terminal serves as an interface between land and sea transportation. Its main functions are to receive export containers from shippers for loading onto vessels and to discharge import containers from vessels for picking up by consignees, Murthy et al. (2005). Terminals also have storage yards for temporary storage of containers. Container terminals are considered essential infrastructure because they are highly capital-intensive, and specialized equipment is needed to handle and transport containers within the port system, e.g., a quay crane (QC) can cost upwards of US$10 million.

In an increasingly competitive and global industry, ports have to ensure efficiency in their management of yard resources. Efficient port management involves a variety of interrelated operational decisions to achieve a range of goals, some of which include

the minimization of berthing time of vessels, resources needed for handling the workload, congestion on the roads, and efficient usage of limited yard storage space.

One of the main goals is to minimize the time duration the ship (which corresponds to a customer in the traditional setting) is berthed in the port, or vessel makespan. An industry estimate puts the cost of a ship being berthed at a port to US$1,000 an hour, Peterkofsky and Daganzo (1990). An important quayside factor, which directly impacts the vessel makespan is the way cranes are scheduled to load and discharge containers from the vessels, which is a step in stowage planning.

The objective of stowage planning is to achieve an efficient and smooth discharge, restowage, and loading of containers on vessels to obtain an expeditious on-time turnaround of vessels. It is carried out hours or days before the vessel's arrival and is a fundamental part of terminal management. The steps in stowage planning differ from port to port, but for most of them it covers import and export planning, input of stowage instructions, crane sequencing, yard slotting, and vessel stability checks, Zhang (2002).

In this work we focus on the crane sequencing problem, which is to partition all the loading and discharging jobs among the vessel's allocated QCs, and decide the order at which the jobs are to be executed. Before the berthing of the vessel, the shipping company usually provides a work instruction, called the load profile, which details the precise location on the ship and exact identity of the containers which are to be loaded or discharged. Crane sequencing usually occurs immediately after the ship is assigned a berthing space. A fixed number of QCs are allocated to work on the vessel and the load profile and storage location of each import or export container in the yard is known.

While the single ship problem has already been studied, we address the multi-ship problem where the ships are linked by means of yard congestion constraints. The discharging operations of ships need to be appropriately synchronized so as not to create congestions at the yard. We model the underlying problem as an integer program, which is solved by a combination of Lagrangian relaxation and column generation. Not surprisingly, it turns out that the makespan objective function poses a big challenge. A primal heuristic is also proposed. This state-of-the-art solution methodology is benchmarked against a commercial optimization solver. Much larger, realistic size problem instances were solved by using the presented methodology.

The presented work has several important contributions. First, we present the underlying nontrivial model, where the modeling challenges arise from crane sequencing and the makespan objective function. Second, we present a new column generation based algorithm for the single ship problem where first the model is reformulated as a generalized set covering type problem. The underlying model and algorithm are completely different from those presented by Daganzo (1989), where the single ship problem is solved by branch-and-bound. We also point out that our single ship problem is more complex since we capture several operational constraints not addressed so far. This is critical for a practical application in an industrial setting. These additional constraints capture operational practices of a mega-container terminal and they include QC clearance (QCs cannot be too close to each other due to safety reasons). Our third important contribution is in designing a solution methodology for solving the multi-ship problem. We relax in the Lagrangian relaxation spirit the yard congestion constraints. The underlying restricted master problem corresponds to several single ship problems, which are solved by branch-and-price.

In Section 2 we study the single ship problem. We propose the generalized set covering formulation and we outline the underlying branch-and-price algorithm. The extension to the multi-ship problem is presented in Section 3. The Lagrangian relaxation is given in this section and the underlying solution methodology is presented in the same section. Extensions considering some practical aspects to the basic model are discussed in Section 4. Section 5 reports a computational study. We conclude the introduction with a more detailed description of relevant port operations, basic principles in relevant optimization methodologies, and a literature review.

1.1. Relevant Port Operations

Most container terminals have two main operational interfaces, quayside and landside. Quayside activities deal with the loading and unloading operations of ships, while landside activities involve loading and unloading of containers on or off external trucks, trains or yard storage locations. Some of the equipment and resources put to use in both interfaces are shown in Figure 1. A typical QC has a width of 25.8 meters, and its usual performance is in the range of 22-30 containers an hour. It is common practice to allocate up to 5 QCs to large vessels, and up to 2,000 containers to be handled per vessel in large ports. QCs run on tracks parallel to the berth line; this horizontal move-

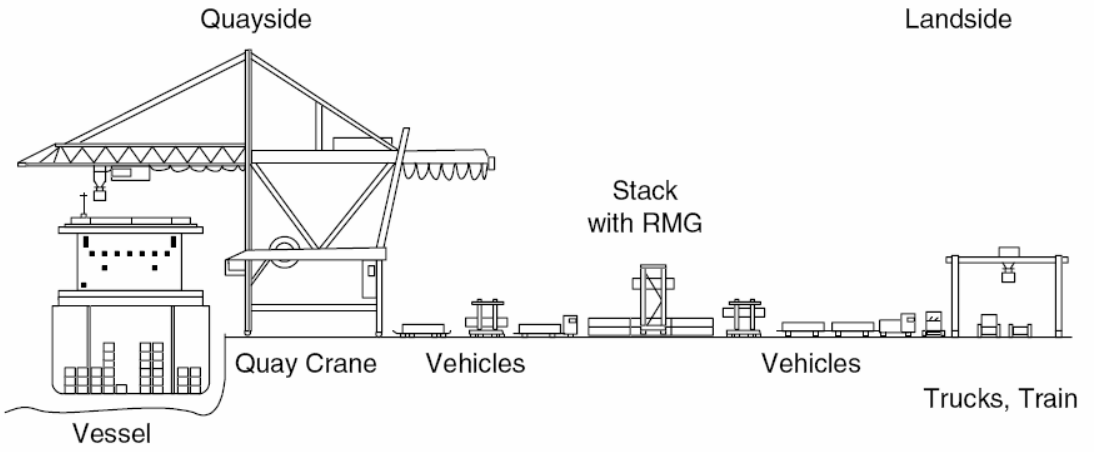

Figure 1: The two operational interfaces of a container terminal system (RMG stands for rail mounted gantry)

ment is known as gantrying. The transportation of containers between the yard storage locations and the quayside is carried out primarily by trucks or automated guided vehicles.

A container terminal has also a storage yard which is usually divided into rectangular regions, known as yard blocks. Each block has approximately 6 to 8 rows for storing containers in stacks, with an additional lane for truck passing. A row may have up to 20 stacks placed end-to-end, each of which can be up to 8 levels high. These blocks are served by yard cranes. Yard cranes remove and place containers from the stacks directly onto trucks, which park in the passing lane while the transfer occurs. Traffic congestion caused by high rates of loading and unloading containers from a particular block is a significant concern.

Upon a vessel's arrival at the terminal, it is assigned to a berth for loading and discharging of containers by QCs. Discharged containers are placed onto trucks by QCs for transportation to pre-determined storage locations in the yard, awaiting pickup from a local consignee or restowage onto another vessel. Yard cranes lift containers from the trucks onto their assigned stacks; the trucks are then recycled back into usage for other jobs.

1.2. Branch-and-price and Lagrangian Relaxation

Large-scale linear programs are often solved by delayed column generation. In this algorithm, at every iteration, only a subset of columns is considered. The problem with only a subset of columns is called the restricted master problem (RMP). In every iteration of the algorithm, first the RMP is solved. Let π be the optimal dual vector, which for ease of discussion we assume it exists. Next the so-called subproblem or pricing problem is solved. In solving the subproblem we identify a set *S* of columns with the lowest reduced cost with respect to π. If we cannot find a column with negative reduced cost, then we stop because π is an optimal dual solution to the original problem and together with the optimal primal solution to the RMP we have an optimal primal/dual pair. Otherwise, we append columns in *S* to the RMP and the entire procedure is iterated. After several iterations, when a large number of columns have been included in the RMP, columns with large reduced costs are removed from the RMP.

Branch-and-price is a solution methodology for solving large-scale linear mixed integer models, Barnhart et al. (1998). It is essentially branch-and-bound where every LP relaxation is solved by delayed column generation. Typically a specialized branching heuristic is employed.

Lagrangian relaxation, see e.g., Fisher (1985), Geoffrion (1974), is a different widely used technique for solving large-scale mixed integer programs. Suppose we can partition constraints into "easy" and "difficult". The concept behind this classification is that if the difficult constraints are removed, the resulting problem is easily solvable. In Lagrangian relaxation, difficult constraints get a linear penalty and are moved to the objective function. The resulting problem is called the Lagrangian relaxation and its objective value is a function of the penalties. Let us assume that we deal with a maximization problem. For any given values of penalties, the Lagrangian relaxation is computationally easy. Moreover, it always provides an upper bound on the optimal value. The goal now is to find the best upper bound, i.e., to minimize the value of the Lagrangian relaxation over all possible penalties. This is the Lagrangian dual problem, which is typically a nonlinear convex optimization problem. In practice it can be solved by a variant of a subgradient algorithm. One drawback of this approach is that there is no guarantee to find feasible solutions. They have to be constructed heuristically during the execution of the subgradient algorithm. The algorithm is very appealing because it is easy to implement and it can handle complex (difficult) side constraints.

1.3. Literature Review

The crane scheduling problem was first highlighted by Daganzo (1989), who proposed an exact linear integer programming formulation for loading ships. Available QCs are assigned to ship bays at discretized time periods. The problem with the objective of mi-

nimizing makespan is solved using enumerative techniques for up to 3 ships. Both the static case, where no other ships arrive in the planning horizon, and the dynamic case are considered. There are several important differences between this work and the presented work. First, we capture several requirements such as the clearance constraints and the yard congestion constraints, which are not captured by Daganzo. Second, our generalized set covering formulations are entirely new and this is the main reason why we are able to solve large-scale instances. Peterkofsky and Daganzo (1990) discuss a branch-and-bound algorithm based on the same formulation to give exact solutions in reduced time.

Kim and Park (2004) similarly discuss the crane scheduling problem, but they assume that there may be multiple tasks or container clusters within a single bay, as opposed to where a bay is considered the smallest positional unit. They study only the single ship problem without clearance constraints. They schedule tasks to QCs based on task precedence constraints. Branch-and-bound and heuristic search algorithms are proposed to obtain solutions to the problem.

Bish (2003) develops a heuristic algorithm based on formulating the problem as a three-fold transshipment problem – determining a storage location for each unloaded container, dispatching vehicles to containers and scheduling the loading and unloading operations of cranes. Since many operational constraints are not considered (e.g., clearance and yard congestion), the resulting model can be solved as an LP. Several very quick heuristics are proposed.

Crane scheduling at the yard is addressed in Cheung et al. (2002). The authors do not consider the ship side of operations but they provide a microeconomics modeling of yard operations, in particular crane movements. A Lagrangian relaxation and piecewise linear approximations are used as solution methodologies. While this work provides more details on the yard side than we do, it does not consider ship bays and the corresponding QCs. A similar problem is addressed in Zhang et al. (2002). Another work dealing with a microscopic view of crane movements is Kim and Kim (1999). They model the number of containers a transfer crane picks and the corresponding bay sequence assignment. Like in Cheung et al. (2002), the QC aspect of the problem is neglected.

An excellent survey on port operations and previous related work can be found in Steenken et al. (2004).

2. The Single Ship Problem

In this section we study the single ship problem. We first provide a compact formulation and then a formulation with an exponential number of columns. The latter is used in our multi-ship algorithm.

2.1. The Compact Formulation

For the single ship problem, we assume that no other vessels berth during the planning horizon, which is the maximum time in which all crane operations have to be completed. The entire planning horizon is discretized. The length of each interval is the amount of time needed for a QC to handle a standard 20-foot container, i.e., time needed to perform the smallest unit of work. QCs can only move and be assigned to a bay at discrete

intervals. The input data to the problem are: (1) the number of QCs allocated to work on the vessel, (2) the number of bays in the vessel, and (3) the vessel load profile. (A vessel is divided along its length into segments known as bays, which can be several rows of containers across and several tiers deep). The sequencing constraints for the single ship model are as follows.

- A QC can only be positioned and work at a single bay at any time.
- A QC must be positioned at least *r* bays away from any adjacent QCs on the left and right (for safety reasons). These are the clearance constraints.
- QCs cannot cross each other's path and therefore must be ordered by position at all times.

The following assumptions are made.

- QC gantrying time is small compared to the time it takes to handle a container and can therefore be ignored in the calculation of vessel's makespan.
- The number of allocated QCs is fixed for the entire duration of vessel's operations.
- All QCs are identical and have similar work rates.
- There is no delay in trucks delivering containers to the QCs.

We show in Section 4 how to modify the models and algorithms to relax some of these assumptions. For ease of exposition, we assume only ship discharging operations are handled. However, the same models apply to loading operations and any combination of the two. Since all QCs work rates are identical, we can assume that in each time period a single container is handled. In other words, after discharging a single container from a bay, QCs can be repositioned. The following notation is used throughout the paper.

Indices:

j Bay number, in increasing order of their location on the vessel, i.e., left to right;

k QC number, in increasing order of their relative position, i.e., left to right;

t Time period index, denoting the interval of time from t-1 to t;

Parameters:

C Number of allocated QCs;

H Number of bays in the vessel;

f_j Number of containers to be discharged in bay *j;*

T Total number of time periods in the planning horizon, which can be set to $\sum_{j=1}^{H} f_j$;

r QC clearance value, in terms of the number of bays;

The problem is to sequence QCs in such a way that the makespan of all operations is minimized. We have to obey all QC related constraints. In addition, all of the containers must be discharged.

The model employs the following decision variables (see Figure 2).

$x_{jk}(t)$ 1 if QC k is positioned at bay j at time period t, and 0 otherwise;

$\delta_{jk}(t)$ 1 if QC k is handling a container at bay j at time period t, and 0 otherwise;

$\gamma(t)$ work completion flag: 0 if all container jobs have not yet been completed at time period t and 1 otherwise;

We need both x and δ variables to model the fact that a QC can be positioned at a bay but its status is idle.

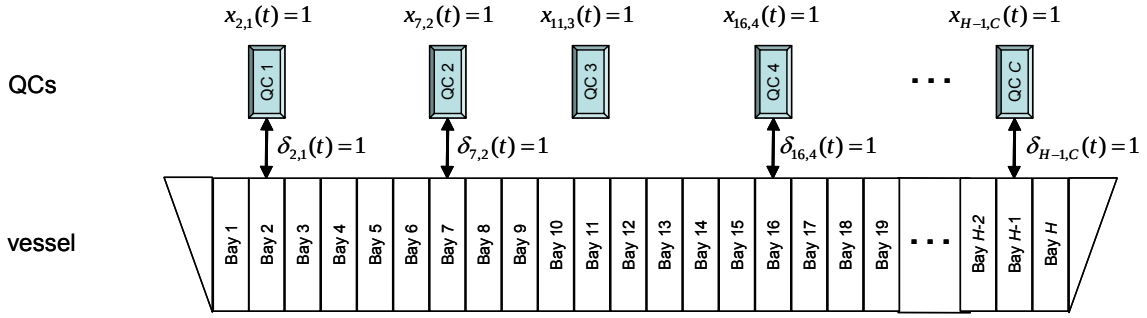

Figure 2: Interpretation of variables

The mathematical formulation for the single ship model is as follow.

$$\text{Maximize } \sum_{t=1}^{T} \gamma(t) \tag{1}$$

Subject to

$$\sum_{j=1}^{H} x_{jk}(t) = 1 \qquad k = 1,...,C; t = 1,...,T \tag{2}$$

$$\sum_{k=1}^{C} x_{jk}(t) \le 1 \qquad j = 1,...,H; t = 1,...,T \tag{3}$$

$$1 - x_{jk}(t) \ge \sum_{l=\max\{1,\, j-r\}}^{j-1} x_{lm}(t) \qquad j = 2,...,H; k = 1,...,C; m = 1,...,C; t = 1,...,T \tag{4}$$

$$1 - x_{jk}(t) \ge \sum_{l=j+1}^{\min\{j+r,\, H\}} x_{lm}(t) \qquad j = 1,...,H-1; k = 1,...,C; m = 1,...,C; t = 1,...,T \tag{5}$$

$$x_{jk}(t) \le \sum_{l=j+1}^{H} x_{l,k+1}(t) \qquad j = 1,...,H-1; k = 1,...,C-1; t = 1,...,T \tag{6}$$

$$x_{jk}(t) \le \sum_{l=1}^{j-1} x_{l,k-1}(t) \qquad j = 2,...,H; k = 2,...,C; t = 1,...,T \tag{7}$$

$$\sum_{k=1}^{C} \sum_{t=1}^{T} \delta_{jk}(t) = f_j \qquad j = 1,...,H \tag{8}$$

$$\delta_{jk}(t) \le x_{jk}(t) \qquad j = 1,...,H; k = 1,...,C; t = 1,...,T \tag{9}$$

$$\gamma(t) \le \frac{\sum_{k=1}^{C} \sum_{l=1}^{t} \delta_{jk}(l)}{f_j} \qquad j = 1,....,H; t = 1,...,T \tag{10}$$

$$x, \delta, \gamma \quad \text{binary} \tag{11}$$

Constraints (2) and (3) respectively ensure that all QCs cannot gantry away from the vessel at any time and that only one QC can be positioned at a bay at any time. Constraints (4) an (5) enforce the QC clearance condition, stating that if any QC is positioned at a particular bay, all other QCs are restricted from being positioned r bays to the left and right respectively. Constraints (6) and (7) describe the QC 'ordering' conditions, where a 'higher-numbered' or 'lower-numbered' QCs must be respectively positioned to the right and left of any QC. Constraint (8) states that all required container jobs must be completed within the planning horizon. Constraint (9) ensures that a QC must be positioned at a bay if it is working there. Constraint (10) defines the work completion flag for bay j. Objective function (1) ensures that when the value of the right-hand-side of (10) sums to 1 for all H bays at time period t, $\gamma(t)$ will be 1. If this is not the case, then the right-hand side will be strictly less than 1 and thus $\gamma(t) = 0$. Therefore the objective function evaluates the value of the vessel makespan. Clearly all variables are binary as stated in (11).

Constraints (4) and (5) can also be modeled in their segregated form $1 - x_{jk}(t) \geq x_{lm}(t)$, however this leads to many constraints (but a tighter LP relaxation). The presented aggregated constraints turned out to be more efficient within a commercial solver.

2.2. Branch-and-price

In this section, the compact single ship model, described in Section 2.1, is reformulated as a generalized set covering problem with an exponential number of variables. It is shown in Section 5.1 by means of computational experiments that this compact formulation is more efficient for large-scale problems.

Reformulation

The key principle of the reformulation is to encode an entire QC schedule in a give time step as a decision variable. Consider a feasible assignment of QC positions to bays (a QC position-to-bay assignment). Such an assignment must obey the clearance requirements and the ordering of QCs. We can encode a QC position-to-bay assignment as a *C*-tuple $(j_1, j_2, j_3, ..., j_C)$, which represents that QC 1 is positioned at bay j_1, QC 2 at bay j_2, etc. Let P be the set of all such feasible assignments. Feasibility is equivalent to requiring $j_i + r \leq j_{i+1}$ for $i = 1, 2, ..., C-1$.

We introduce a variable z_p for $p \in P$, which is the number of times the QC position-to-bay assignment *p* is selected. The formulation reads

$$\text{Minimize} \sum_{p \in P} z_p \tag{12}$$

Subject to

$$\sum_{p \in P: j \in p} z_p \geq f_j \qquad j = 1, ..., H \tag{13}$$

$$z \quad \text{nonnegative integer.} \tag{14}$$

The expression $j \in p$ represents the fact that bay j is assigned to a QC in assignment p. Formally, there exists q such that $j_q = j$ if $p = (j_1, j_2, j_3, ..., j_C)$.

Objective function (12) minimizes the total number of QC positions-to-bay assignments needed to handle all the jobs in the load profile. Constraints (13) impose the unloading requirements. They impose that for every bay j we must select at least f_j assignments.

From the formulation, given a feasible solution z, it is clear how to obtain the corresponding x values of the compact formulation. The corresponding δ's in the compact formulation can be obtained by starting with x's, and then arbitrarily setting them to zero in order to satisfy constraints (8) at equality. This procedure also yields the following statement.

Proposition: The LP relaxation of (12)-(14) is at least as strong as the LP relaxation of (1)-(11).

Proof. Given a feasible solution z to the LP relaxation of (12)-(14), for each $p \in P$ we first construct $\lfloor z_p \rfloor$ time periods with the corresponding binary x's and δ's as described above. The possible fractional value $z_p - \lfloor z_p \rfloor$ is then again assigned to x's and δ's in a similar fashion except that this time their values might be fractional. This shows how to obtain a corresponding feasible solution to the LP relaxation of (1)-(11) with no worse cost. □

The Branch-and-price Algorithm

In this section we discuss the most important components of the branch-and-price algorithm for our generalized set covering formulation. As discussed in Section 1.2, branch-and-price algorithms are branch-and-bound algorithms where LP relaxations are solved by delayed column generation.

In our branch-and-price, depth-first search is adopted so that integer solutions are found rapidly. Since the column costs are integral, we can prune nodes whose value of the LP relaxation is greater than an integer away from the best integer solution. This significantly speeds up the branch-and-price algorithm.

We next provide more details on obtaining an initial solution, pricing, and branching.

A Heuristic

A quick heuristic is developed for two purposes: as a basis for comparison against the results from a commercial solver and as a primal heuristic within the branch-and-price algorithm. It is a greedy heuristic based on selecting good QC position-to-bay assignments one period at a time.

Let $l_j(t)$ be the remaining workload at bay j and time t onwards, i.e., it is f_j minus everything that has already been unloaded from bay j before time t. Then a lower bound for the remaining makespan $RM(t)$ at time period t is

$$\max\left\{\left\lceil \sum_{j=1}^{H} l_j(t) \Big/ C \right\rceil, \max\{l_j(t) \mid j = 1, ..., H\}\right\} \le RM(t). \qquad (15)$$

It is clear that in instances where clearance constraints are not heavily restrictive, i.e., when QCs are not tightly 'packed' or when work load is well distributed, the minimum possible vessel makespan is going to be the first term in (15). However, since only one QC can work on a bay at any time, a possible value for $RM(t)$ could be the second term, or the bay with the heaviest remaining work load.

In the heuristic, in each time period, we make assignment decisions so that the lower bound on $RM(t+1)$ is as low as possible. We thus observe two sequencing principles which guide the heuristic towards a good solution: (1) the bay with the maximal remaining workflow $l_j(t)$ is handled, and (2) remaining QCs should not be idle; they are assigned to work on other bays with heavy work loads. This yields a strategy where QCs are selected based on the nonincreasing order of $l_j(t)$. Clearly we must satisfy all of the feasibility constraints.

The following describes the overall heuristic procedure.

Step 1: Let $l_j(1) = f_j$ for all j=1,…,H and set t=1.

Step 2: Rank all bays in terms of the remaining work load $l_j(t)$ for the current time period t.

Step 3: Assign QCs to bays for the current time period t based on the just computed order. Skip all bays that violate the clearance requirement.

Step 4: Compute $l_j(t+1)$ by subtracting from $l_j(t)$ the work performed in each bay at time period t.

Step 5: If $\sum_j l_j(t+1) = 0$, there is no remaining work and the algorithm terminates with t as the makespan. Otherwise, increment t by 1 and go to Step 2.

Within branch-and-price it is easy to adjust the heuristic by appropriately setting initial l_j values to reflect the current branching decisions.

Pricing

The pricing problem provides a column that prices out favorably or proves that none exists. The dual variables π associated with constraint (13) from the RMP are used to solve the pricing problem:

$$\text{1-Maximize} \sum_{j=1}^{H} \pi_j x_j \tag{16}$$

Subject to

$$\sum_{j=1}^{H} x_j = C \tag{17}$$

$$C\left(1-x_j\right) \geq \sum_{l=j+1}^{\min\{j+r,H\}} x_l \qquad j = 1,...,H-1 \tag{18}$$

$$C\left(1-x_j\right) \geq \sum_{l=\max\{1,j-r\}}^{j-1} x_l \qquad j = 2,...,H \tag{19}$$

$$x \quad \text{binary.} \tag{20}$$

The decision variable x_j is 1 if a QC is positioned at bay j and 0 otherwise. Constraints (18) and (19) impose clearance, while (17) ensures that all QCs are assigned. Objective function (16) arises from the calculation of the reduced cost.

Next we show how to reformulate the pricing problem as a shortest path problem. The nodes of the network correspond to bay/QC pairs, i.e., there is a node for each $(j,k), j=1,...,H, k=1,...,C$, which represents that QC k is assigned to bay j. There is an arc between (j,k) and $(q,k+1), k=1,...,C-1$ only if the clearance constraints are satisfied, i.e., $q \geq j+r$. We also add a source s connected to all nodes $(j,1)$ and a sink t connected from all nodes (j,C). Each arc from (j,k) to $(q,k+1)$ has cost $-\pi_j$, outgoing arcs from s get a zero cost, and the arc connecting (j,C) and t bears a cost of $-\pi_j$. It is easy to see that the pricing problem is equivalent to finding a shortest *s*-*t* path. Note that this network is acyclic and therefore a shortest path can be found by a single forward scan of all arcs. The complexity of the algorithm is $O(H^2 \cdot C)$, which is pseudo polynomial in the input size (recall that polynomial in our case means polynomial in H and $\log(C)$). In practice these two values are low and therefore the shortest path algorithm is very efficient.

Branching

A valid branching scheme partitions the solution space in such a way that the current fractional optimal solution of a node is excluded, integer solutions remain intact, and finiteness of the algorithm is ensured, see e.g., Lubbecke and Descrosiers (2005). Branch-and-price algorithms typically require customized branching in order to fulfill these conditions.

Computational experiments have shown that for this application standard branching on variables works efficiently. It is applied on the least fractional column. Note that such branching also stresses feasibility. The problem with variable branching is that during pricing a column that is already in the restricted master problem can be regenerated. This would yield a possible infinite branch-and-price algorithm. In our problem we observed that columns are very seldom regenerated. The main reason for such a favorable behavior lies in the fact that our variables are nonnegative integers (and not binary).

Nevertheless, columns can be regenerated and we cope with this in the following way. Consider the shortest path formulation of our pricing problem. Every integer solution to our problem implies a set of path in the network with nonnegative integer flow on each arc. Thus the total flow on each arc is a nonnegative integer. Using this observation we can branch as follows.

Consider a possible fractional solution z^* to the restricted master problem. Based on this solution we compute the flow on each arc in our shortest path network. If there is an arc e from (j,k) to $(q,k+1)$ whose total flow is fractional, we create two branches. Let $u_e^*(z^*)$ be the value of the flow on this arc. On one branch we add the constraint

$$\sum_{p \in P: e \in p} z_p \leq \lfloor u_e^*(z^*) \rfloor$$

while on the other branch we add

$$\sum_{p \in P: e \in p} z_p \geq \left\lceil u_e^*(z^*) \right\rceil .$$

Such branching clearly cuts off the current fractional solution and it does not exclude any feasible solution.

It remains to argue that given a fractional solution to the restricted master problem, we can always find such an arc e. As stated, this is not true, however we can address this by the following proposition.

Proposition: Let z^* be a solution to the restricted master problem whose flow arc values are all integral. Then there exists a feasible integral solution $\bar{z}$ with the same objective value.

Proof. Let us regard each column in z^* as an s-t path. We consider the maximum s-t flow problem with capacity on arc e equal to $u_e^*(z^*)$. Then z^* defines a flow in the network whose value equals to the corresponding value of the restricted master problem and all of the capacities are saturated.

From the maximum flow theory it is known that there exists a flow that can be decomposed into paths with integer flow on each one of them. This holds because all capacities are integers by definition and assumption. We denote by $\bar{z}$ this underlying set of paths and corresponding values. It is also clear that all arcs are saturated with respect to $\bar{z}$ and as a result the flow going through each node based on $\bar{z}$ is equal to the flow going through each node based on z^*. This implies that $\bar{z}$ is a feasible solution to our problem since constraints (13) require that the flow going through all nodes $\{(j,k)\}_k$ must be equal to or greater than f_j. It is also clear that the objective value of $\bar{z}$ equals to the objective value of z^*. □

This shows that the proposed branching rule is a valid branching rule. It is also easy to adjust the pricing problem to capture these branching decisions. By adding branching constraints we obtain new dual values corresponding to arcs in the pricing network. It is easy to adjust the shortest path pricing by adding these dual values as arc costs.

In our algorithm we branch on variables until we encounter a column that was regenerated. At this point we switch to the aforementioned branching. Even though branching on variables creates extremely unbalanced trees, it outperforms arc based branching.

3. The Multi-ship Problem

In this section we model several ships berthing at specific, preplanned times throughout the entire planning horizon. The objective remains to minimize the weighted cumulative makespan of all vessels, while ensuring that a new set of yard congestion constraints are adhered to. The modeling assumptions and sequencing constraints used in the single ship model are also applied here subject to each individual ship.

3.1. The Formulation

.In the multi-ship model, QC sequencing of a vessel would be independent of other vessels if not for the imposed additional yard congestion constraints. These constraints prevent the number of QCs handling containers that are slated for a particular yard storage location from exceeding a given quantity, known as the yard activity threshold. The aim is to limit the level of yard crane and truck activities in the yard blocks and hence prevent congestion and other operational inefficiencies. Thus, if the yard activity thresholds had been breached, QC jobs in various vessels would have had to be resequenced. New input parameters for the multi-ship model are:

- total number of ships berthing,
- for each vessel, the cost per time unit,
- vessel berthing time, or the time at which QC operations for the vessel may commence,
- yard block number for storing each handled container, and
- yard activity threshold for each yard block.

The complete multi-ship formulation is provided in Appendix A (see Figure 3 for variable definition). This formulation is a compact one, i.e., it has polynomially many variables. It is very hard to derive a formulation similar to the set covering formulation (12)-(14). An important observation about this problem and model is that the yard congestion constraints are the only constraints that link the ships together.

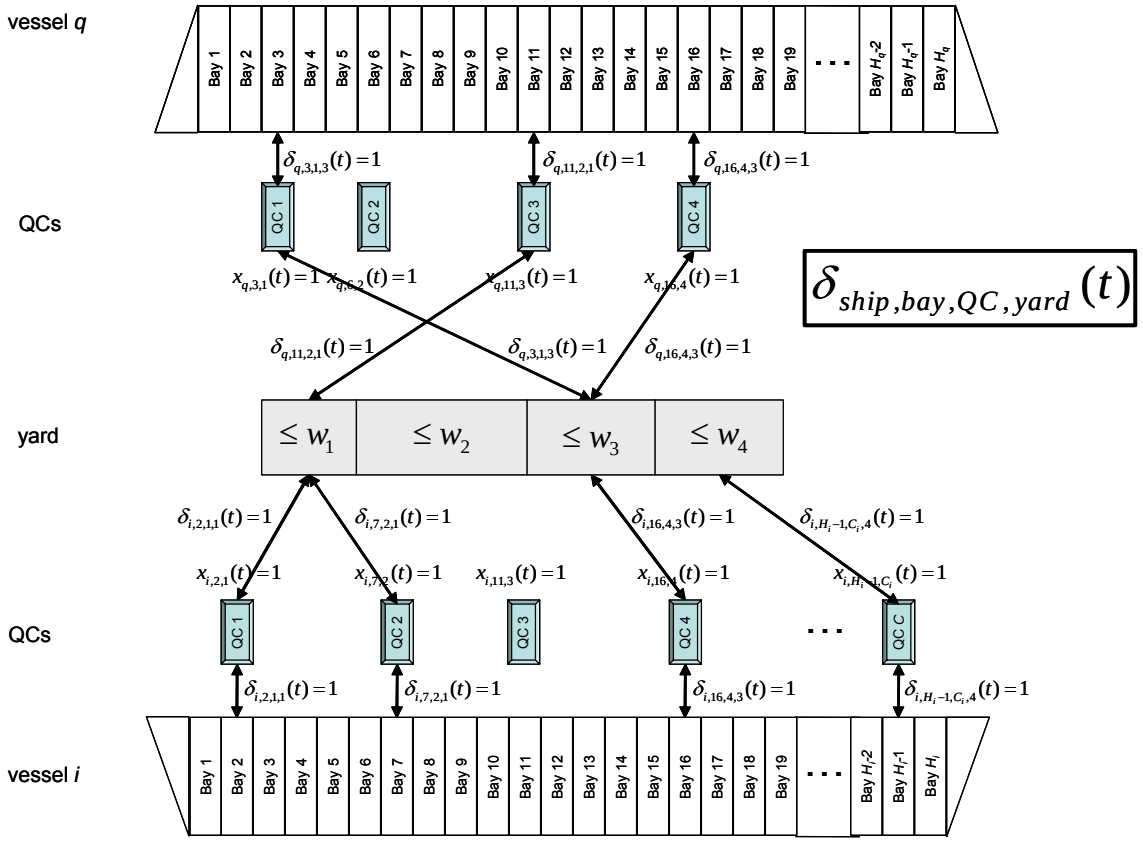

Figure 3: Interpretation of variables in the multi-ship case

We assume that the time spent in moving containers from vessels at the quayside to all yard blocks is the same. In practice, this time depends on the distance between the vessel and yard blocks, and on the congestion in the port road network, however, these dependencies are often insignificant in practice.

3.2. Solution Methodology

Based on the observation that yard congestion constraints are the only constraints linking ships together the model is amended to Lagrangian relaxation.

The yard congestion constraints, which link the decision variables associated with different vessels together, are relaxed and the resulting problem decomposes by ship. For given Lagrangian multipliers $\lambda = (\lambda_{zt})_{z,t}$ (z indexes yard locations) we define the corresponding Lagrangian relaxation as

$$L(\boldsymbol{\lambda}) = \text{Minimize}_{x,\delta,\gamma} -\sum_{i=1}^{S}\sum_{t=d_i}^{d_i+T_i-1} c_i\gamma_i(t) + \sum_{z=1}^{L}\sum_{t=1}^{T_{\max}}\lambda_{zt}\left(\sum_{i\in Q_t}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\delta_{ijkz}(t) - w_z\right), \tag{21}$$

where the minimum is subject to constraints (45)-(56) and (58)-(59). Here $\gamma_i(t)$ models the makespan of ship *i* and c_i is the per time unit cost of ship *i*. To tackle (21) note that it decomposes by ship and therefore the subproblems are reformulated into a set covering form and solved by branch-and-price. Due to the additional penalty term in the objective function the transformation is not straightforward.

The goal is to solve the Lagrangian dual problem

$$L^* = \text{Maximize}_{\boldsymbol{\lambda}\geq 0} L(\boldsymbol{\lambda}) \tag{22}$$

The subgradient method, first proposed by Held and Karp (1970), is applied to solve (22).

Solutions to (21), produced repeatedly (by branch-and-price) during the subgradient procedure, are rarely primal feasible to the original problem. They are used as good starting points for a heuristic approach to find a primal feasible solution.

Decomposition by Vessel into Subproblems

When the coupling yard congestion constraints are relaxed, the objective function and all of the nonrelaxed constraints are separated into S (the number of ships) independent minimization subproblems, each with objective function $L^i(\boldsymbol{\lambda})$ defined in (23) below. Expression (21) can be rewritten as

$$L(\boldsymbol{\lambda}) = \text{Minimize}_{x,\delta,\gamma} -\sum_{z=1}^{L}\sum_{t=1}^{T_{\max}}\lambda_{zt}w_z + \sum_{i=1}^{S}\underbrace{\left\{\sum_{t=d_i}^{d_i+T_i-1}\left(\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\sum_{z=1}^{L}\lambda_{zt}\delta_{ijkz}(t) - c_i\gamma_i(t)\right)\right\}}_{L^i(\boldsymbol{\lambda})}. \tag{23}$$

The constraints of the subproblems are similar to the single ship model, except that the variables δ's and the input vessel load profile f's are additionally indexed by z for the yard block number. The full formulation of the subproblem for ship i is given in Appendix B. Term $L^i(\boldsymbol{\lambda})$ in (23) is inflated from the single ship model objective (1) by the penalty term

$$\sum_{t=d_i}^{d_i+T_i-1}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\sum_{z=1}^{L}\lambda_{zt}\delta_{ijkz}(t). \qquad (24)$$

Although the subproblems are easier to solve compared to the original problem, they are still very difficult for large-scale problems. However, due to the relative similarity between these subproblems and the single ship model, we are able to take advantage of the efficient branch-and-price solution methodology developed for the single-ship model. The branch-and-price model needs to be adjusted to take into account the penalty term (24) in $L^i(\boldsymbol{\lambda})$. The presence of a time-indexed cost coefficient λ_{zt} means that the cost of each column differs by time period and a different pricing problem and model have to be solved. The expected increase in the problem size of the pricing problem suggests that certain problem specific measures, not used for the single ship problem, have to be developed to reduce the computational complexity of the resulting branch-and-price algorithm.

The Set Covering Formulation of the Lagrangian Subproblems

As in the single ship problem, the Lagrangian subproblem is formulated as a set covering problem. Let us focus on ship i. Here, because the cost of the QC activity in each period is affected by δ's, each variable represents the QC position-to-(bay,yard position) assignment for a particular period. Each column can be encoded as a *C*-tuple, where each coordinate represent a (bay,yard position) pair. For example, $((j_1,z_1),(j_2,z_2),...,(j_C,z_C))$ encodes that the first QC is positioned at bay j_1 and it is unloading a container heading for yard position z_1 (equivalently, $\delta_{ij_1 1z_1}(t)=1$). Because the work completion constraints have to be satisfied, the set partitioning structure has the right-hand side vector *f*, which is the vessel's load profile for each (*j*,*z*) pair. Identical columns may differ in cost if designated to different periods due to the λ_{zt} cost coefficient.

Let R be the set of all feasible QC work-assignments. For $r \in R$ we introduce binary variables $y_{r,t}$, which is 1 if QC assignment *r* is selected in time period *t*, and 0 otherwise. The set partitioning formulation reads

$$\text{Minimize} \sum_{t=d_i}^{d_i+T_i-1}\sum_{r\in R}\left(c_i+\sum_{z=1}^{L}\lambda_{zt}\right)y_{r,t} \qquad (25)$$

Subject to

$$\sum_{t=d_i}^{d_i+T_i-1}\sum_{r\in R:(j,z)\in r} y_{r,t} = f_{ijz} \quad j=1,...,H_i;\ z=1,...,L \qquad (26)$$

$$\sum_{r\in R} y_{r,t} \le 1 \qquad t=d_i,...,d_i+T_i-1 \qquad (27)$$

$$\sum_{r\in R} y_{r,t+1} - \sum_{r\in R} y_{r,t} \le 0 \qquad t=d_i,...,d_i+T_i-2 \qquad (28)$$

y binary.

Notation $(y,z)\in r$ denotes the fact that if $r=((j_1,z_1),(j_2,z_2),...,(j_C,z_C))$, then there exists *q* such that $j_q=j, z_q=z$.

Objective function (25) captures 2 parts, contribution to the makespan (1 for all columns) and a penalty term associated with the QC activity in the designated period. Constraint (26) captures the work completion requirement and (27) guarantees that in each time period we select a single QC work-assignment. Constraint (28) imposes that columns designated to earlier periods can be selected only if later periods are, ensuring the vessel makespan is evaluated correctly. They impose that there exists a period $\bar{t}$ such that a QC work assignment is selected for periods $t = d_i, ..., \bar{t}$ and $\sum_{r \in R} y_{r,t} = 0$ for $t = \bar{t} + 1, ..., d_i + T_i - 2$.

Typically set covering formulations are preferred over set partitioning formulations because their LP relaxations are numerically more stable (less degenerate) and thus easier to solve. In the multi-ship formulation given in Appendix A, suppose we allow work completion constraints to be exceeded; in other words, allow more work to be done than necessary. Assuming that K more jobs are performed than necessary, we can arbitrarily reset K values of δ's that are 1 to 0. QC position constraints are not violated by this operation because x's are independent of δ's. Neither are the QC work constraints and yard congestion constraints violated since the reduced workload makes them less tight. We conclude that since either cost or feasibility are not affected by the aforementioned operation, the set covering form of the original multi-ship model is equivalent to the set partitioning form, and thus the sign in (26) can be changed to equal to or greater than. From now on we assume that this is the case.

In the set covering problem (25)-(28), we observed by computational experiments that constraints (28) cause a high degree of fractionality in the LP relaxations. Strengthening them through segregation is not practical due to the difficulties in handling dual variables in the pricing problem and the large number of rows. To circumvent this we fix makespan and try all possible makespan values.

If makespan is fixed to T_{MS}, then (28) can be removed. The resulting problem reads

$$\text{Minimize} \sum_{t=d_i}^{d_i+T_{MS}-1} \sum_{r \in R} \left(\sum_{z=1}^{L} \lambda_{zt} \right) y_{r,t} \tag{29}$$

Subject to

$$\sum_{t=d_i}^{d_i+T_{MS}-1} \sum_{r \in R:(j,z) \in r} y_{r,t} \geq f_{ijz} \qquad j = 1, ..., H_i, z = 1, ..., L \tag{30}$$

$$\sum_{r \in R} y_{r,t} = 1 \qquad t = d_i, ..., d_i + T_{MS} - 1 \tag{31}$$

y binary.

Objective function (29) now includes only the penalty term costs since makespan is fixed, while (31) must be satisfied at equality to guarantee the makespan of T_{MS}. We next provide relevant details in solving (29)-(31).

Pricing and Branching

We first observe that for each time period t we have a separate pricing problem. Let now t be fixed. Let π and p be the dual values corresponding to (30) and (31), respectively. Then the pricing problem reads

$$p_t + \text{Minimize} \sum_{j=1}^{H_i} \sum_{z=1}^{L} (\lambda_{zt} - \pi_{jz}) \delta_{jz} \tag{32}$$

Subject to constraints (17)-(19), and

$$\delta_{jz} \le x_j \qquad j = 1,...,H_i; z = 1,...,L \tag{33}$$

$$\sum_z \delta_{jz} \le 1 \qquad j = 1,...,H_i \tag{34}$$

$$x, \delta \text{ binary.} \tag{35}$$

Objective function (32) captures the reduced cost of a column. As before, (17)-(19) enforce QC position constraints, while constraints (33) and (34) dictate where the QCs are actually working.

This pricing problem can be reformulated as a shortest path problem in the same spirit as in the single ship pricing problem. For every bay index $j = 1,...,H_i$ we define $z^*(j) = -\arg\max_{z=1,...,L} \left(\pi_{jz} - \lambda_{zt} \right)^+$. Observe that if x^*, δ^* is an optimal solution to (32)-(35), then we can assume that $\delta^*_{j,z^*(j)} = 1$ and $\delta^*_{jz} = 0$ for all $z, z \ne z^*(j)$ if $x^*_j = 1$, and $\delta^*_{jz} = 0$ for all $j = 1,...,H_i$ if $x^*_j = 0$. As a result the cost of the arc from (j,k) to $(q,k+1)$ is $-\max_{z=1,...,L} (\pi_{jz} - \lambda_{zt})^+$. Branching can also be defined as before based on the flow on arcs.

The pricing problem is solved repetitively for each time period t in the range $d_i \le t \le d_i + T_{MS} - 1$.

The Overall Branch-and-Price Algorithm

The overall algorithm for solving (25)-(28) is to solve (29)-(31) for every T_{MS} by branch-and-price. As we have already mentioned, it turns out that this is more efficient than solving (25)-(28) directly.

Before starting the entire procedure we first establish a lower bound on makespan by ignoring the penalty term $\sum_{z=1}^{L} \lambda_{zt}$ and simply minimizing the makespan, while adhering to clearance, work completion and other constraints. This clearly gives us a lower bound on T_{MS}. We also precompute an upper bound on the overall value by adding the penalty cost component $\sum_{z=1}^{L} \lambda_{zt}$ to each column in the computed minimum makespan solution. This clearly gives us a feasible solution with an appropriate cost. This upper bound is used in branch-and-price algorithms.

A Heuristic for Generation of Feasible Solutions

The heuristic described here attempts to correct yard congestion constraint violations within the Lagrangian framework while keeping the objective function deterioration small, i.e., to avoid excessively increasing vessel makespans. It provides an upper bound to the optimal cost of the multi-ship problem. This bound is also used as a termination criterion and in calculating the step sizes of the subgradient procedure described later. We assume we have a feasible solution to (21).

The heuristic first detects a yard location z and a time period t where infeasibility occurs. It then attempts to swap QC work with any other QCs working on the same bay in different time periods, if this is possible. If it is possible, the swap is performed and clearly the makespan is not increased. If this is not possible (due to the clearance requirements), the QC work is postponed to the end and the makespan is increased by one. These steps are repeated until all of the infeasibilities are removed. The heuristic is randomized at various points. For example, at each iteration within the heuristic, a different search for infeasible z and t is conducted, and QCs may choose a different QC to swap work with. For each subgradient iteration, the heuristic is run several times and the best upper bound Z_{UB} is stored.

Updating Lagrangian Multipliers

The subgradient algorithm searches for optimal Lagrangian multipliers λ_{zt}^{*} that maximize $L(\boldsymbol{\lambda})$. It is well known that $L(\boldsymbol{\lambda})$ is a concave function. An initial Lagrangian multiplier $\boldsymbol{\lambda}^0$ is selected and a sequence of $\boldsymbol{\lambda}$'s is determined by updating at each iteration the current value of $\boldsymbol{\lambda}$ with a step in the direction of the subgradient at the current iterate. If necessary, the resulting subgradient vector is projected back into the non-negative orthant. In our case the updates are as follows, if $\boldsymbol{\lambda}^n$ is the current multiplier at the n-th iteration:

$$G_{zt}(\boldsymbol{\lambda}^n) = \sum_i \sum_{j=1}^{H_i} \sum_{k=1}^{C_i} \delta_{ijkz}(t) - w_z \tag{36}$$

$$\lambda_{zt}^{n+1} = \max\left\{0, \lambda_{zt}^n + t^n G_{zt}(\lambda^n)\right\}. \tag{37}$$

Expression (36) defines the gradient, where δ is an optimal solution to the Lagrangian relaxation (21). The following step-size rule is used:

$$t^n = \frac{\zeta^n \left(Z_{UB} - L(\boldsymbol{\lambda}^n)\right)}{\sum_{z,t}\left(\sum_i \sum_{j=1}^{H_i} \sum_{k=1}^{C_i} \delta_{ijkz}(t) - w_z\right)^2}. \tag{38}$$

Justification for (38) is given in Wolfe and Crowder (1974). In each iteration $L(\boldsymbol{\lambda}^n)$ provides a lower bound on the optimal value of the multi-ship problem. The best lower bound encountered during the execution of the algorithm is denoted by Z_{LB}.

In our experiments, the sequence ζ^n is determined by starting with $\zeta^0 = 1$ and reducing ζ^n by half whenever $L(\boldsymbol{\lambda}^n)$ has failed to increase for 5 consecutive iterations. The following rules are used as termination criteria: (1) we reach 100 iterations, (2) the step-size parameter satisfies $\zeta^k \le 0.005$, (3) no improvement in $L(\boldsymbol{\lambda}^n)$ for 10 iterations, and (4) the optimality gap satisfies $\frac{Z_{UB} - Z_{LB}}{Z_{UB}} < 0.05$.

4. Extensions

In this section we present several enhancement and extensions to the presented models and algorithms. Since the main focus is on the multi-ship problem, we address them only in this context.

Initial Quay Crane Configuration

At the beginning of the time horizon, QCs are already located at certain bays (from the previous operations). In addition, for the same reason a QC might become available only at a certain time. Modeling such requirements requires only slight changes to the set covering formulation (29)-(31). From starting availabilities of QCs, it is easy to determine the number of available QCs in each time period *t*. We would then modify the definition of R to R_t, where in R_t we would no longer consider *C*-tuples, but the number of coordinates would equal to the number of available QCs in time period *t*. If the initial position of a QC needs to be imposed, we would further constrain R_t by fixing the bay of the newly added QC.

Multiple Operations

So far we assumed that discharging is the only operation. We next show how to incorporate loading. For simplicity, let us assume that each bay requires first discharging, which is then followed by loading. To this end, let L^D, L^L be the set of discharging, loading storage locations, respectively. We clearly have $L^D \bigcup L^L = \{1,2,...,L\}, L^D \bigcap L^L = \varnothing$. We need to impose that the discharging locations must be handled before the loading locations.

In the compact formulation of the multi-ship problem presented in Appendix A, a new set of constraints

$$\delta_{ijkz}(t) + \delta_{ij\bar{k}\bar{z}}(\bar{t}) \le 1 \qquad i = 1,...,S;\ j = 1,...,H_i; k = 1,...,C_i; \bar{k} = 1,...,C_i;$$
$$z \in L^L; \bar{z} \in L^D; t = d_i,...,d_i + T_i - 1; \bar{t} \ge t$$

is required to ensure that discharging is performed before loading. It states that if QC *k* is loading at time *t*, then at any later point in time $\bar{t}$ any other QC $\bar{k}$ cannot be discharging. The underlying Lagrangian relaxation from Appendix B requires the addition of the same constraint except that the ship index *i* is dropped.

The changes with respect to the set covering reformulation are more interesting. The model (25)-(28) needs to be augmented by the constraints

$$\sum_{\substack{r \in R \\ (j,z) \in r \\ z \in L^L}} y_{r,t} - \sum_{\substack{r \in R \\ (j,z) \in r \\ z \in L^D}} y_{r,\bar{t}} \le 0 \qquad t = d_i,...,d_i + T_i - 2; \bar{t} = t+1,...,d_i + T_i - 1;\ j = 1,...,H_i. \qquad (39)$$

If $\alpha_{jt\bar{t}}, \bar{t} > t$ are the dual values for (39), then it can be seen that the objective value (32) changes to

$$p_t + \text{Minimize} \sum_{j=1}^{H_i} \sum_{z=1}^{L} (\lambda_{zt} - \pi_{jz} + \beta_{jzt}) \delta_{jz},$$

where

$$\beta_{jzt} = \begin{cases} \sum_{\bar{t}:\bar{t}>t} \alpha_{jt\bar{t}} & z \in L^L \\ \sum_{\bar{t}:\bar{t}<t} \alpha_{j\bar{t}t} & z \in L^D. \end{cases}$$

It is convenient that the pricing problem is of the same complexity as the original pricing problem (32)-(35). Thus we firmly believe that this extension is computationally tractable.

We can similarly handle more than two operations. The concept of different operations allows also to model shifting. In shifting, a set of containers is temporarily unloaded from a bay, and then later loaded into a different bay of the same ship. The loading and unloading tasks of shifting can be modeled as two separate operations and then incorporated as described above.

Quay Crane Time

In many situations, QCs movement times are negligible with respect to the time to perform the underlying operation. Next we show how to handle substantial gantry time if need be.

Let us assume that configurations r and $\bar{r}$ are selected in two consecutive time periods. A QC requires a certain amount of time to move from one bay to another bay. Since r and $\bar{r}$ encode the actual positions of the QCs, it is easy to assign the required time $d(r,\bar{r})$ for the QCs to move from one configuration to the next one (it is the maximum time over all QCs). The new objective is now to minimize the cumulative sum over all selected configuration pairs of all $d(r,\bar{r})$ over all time periods.

To achieve this, we need to augment the decision variables in the set covering formulation (29)-(31). We define $y_{r,\bar{r},t}$ to be one if in time period *t* configuration r is selected and configuration $\bar{r}$ is selected in time period *t*+1. The new set covering model (including Lagrangian multipliers) now becomes

$$\text{Minimize} \sum_{t=d_i}^{d_i+T_{MS}-2} \sum_{\bar{r}\in R, r\in R} \left(d(r,\bar{r}) + \sum_{z=1}^{L} \lambda_{zt} \right) y_{r,\bar{r},t} \tag{40}$$

Subject to

$$\sum_{t=d_i}^{d_i+T_{MS}-2} \sum_{\substack{r\in R:(j,z)\in r \\ \bar{r}\in R}} y_{r,\bar{r},t} + \sum_{\substack{\bar{r}\in R:(j,z)\in r \\ r\in R}} y_{r,\bar{r},d_i+T_{MS}-1} \geq f_{ijz} \qquad j=1,...,H_i, z=1,...,L \tag{41}$$

$$\sum_{r\in R,\bar{r}\in R} y_{r,\bar{r},t} = 1 \qquad t = d_i,...,d_i+T_{MS}-2 \tag{42}$$

$$\sum_{\substack{r\in R,\bar{r}\in R \\ (j,z)\in r}} y_{r,\bar{r},t} \leq \sum_{\substack{r\in R,\bar{r}\in R \\ (j,z)\in \bar{r}}} y_{r,\bar{r},t-1} \qquad t = d_i+1,...,d_i+T_{MS}-2; \quad j=1,...,H; z=1,...,L \tag{43}$$

y binary.

In objective function (40) we now capture the time between two consecutive configurations and the corresponding contribution from Lagrangian multipliers. Constraints

(41) impose the usual requirement to perform all necessary operations. Note that the last time period needs to be handled separately. Based on (42) a single configuration must be selected in each time period. Finally, (43) impose basic compatibility among the assignment variables in two consecutive time periods. They state that if a QC is performing a certain (*j*,*z*) assignment in time period *t*, then the same (*j*,*z*) assignment must be present in the second configuration in time period *t*-1.

The pricing problem (32)-(35) needs to be appropriately modified to capture the new setting. Unfortunately, it becomes more complicated since it would include twice as many variables due to the fact that two consecutive time periods must be considered. We omit these details.

Dynamic Quay Crane Assignment to Ships

So far we assumed that QCs are assigned to ships a priori. Ideally, such decision making should be allowed and, furthermore, dynamically reconfigured in each time period. The proposed framework can be extended also to such situations.

Let $C = \sum_{i=1}^{S} C_i$ be the total number of available QCs. The main idea here is to let the QCs freely float among the ships. We define new assignment variables $\overline{x}_{ikt}$, which are 1 if QC *k* is assigned to ship *i* in time period *t*. The following constraints are then added to the multi-ship model presented in Appendix A:

$$\sum_{i=1}^{S} \overline{x}_{ik}(t) = 1 \qquad k = 1,...,C; t = 1,...,T \tag{44}$$

$$\overline{x}_{ik}(t) \le \sum_{j=1}^{H_i} x_{ijk}(t) \qquad k = 1,...,C; t = 1,...,T; i = 1,...,S.$$

We also redefine the basic assignment variables used in the multi-ship model in Appendix A as $x_{ijk}(t)$ is 1 if in time period *t* QC *k* is assigned to ship *i* and bay *j* on this ship. The remaining constraints remain virtually unchanged except that they apply over all QCs.

In addition to (51), now (44) also provide a linkage among the ships. Thus they have to be relaxed in the Lagrangian spirit. Our solution methodology is still applicable with only minor changes. Each configuration *r* now involves a varying number of QCs and the underlying Lagrangian multipliers of (44) are reflected in the cost.

5. Computational Study

We designed several computational experiments to test our proposed algorithms. The experiments were conducted on a Pentium IV 1.6GHz personal computer with 512MB of RAM and Linux as the operating system. The single and multi-ship models were implemented in OPL Studio 3.7 (www.ilog.com), which utilizes CPLEX 9.1 as the underlying LP and IP solver. The heuristic approach, the branch-and-price algorithm, and the subgradient procedure were implemented by using OPLScript.

In our exact IP model, all the constraints and the objective function are linear, therefore standard IP solvers, such as CPLEX, can be used. We primarily use them to benchmark the quality of solutions generated with the heuristic and the branch-and-

price framework. However, the number of constraints is very large even for problem instances of moderate size and therefore off-the-shelf solvers have severe limitations for our problems.

5.1. The Single Ship Problem

We first investigate the single ship problem. Two data sets were created: the first one with 12 small problem instances, and the second with 3 problem instances of realistic sizes. The problem instances are arranged in the order of increasing H, each with varying T, C, and r input parameters. For each problem instance, the workload is randomly distributed among the vessel bays. Table 1 provides the parameters of each problem instance in the two data sets where instances starting with SSP denote the realistic sets. The last two columns show the number of rows and variables of the compact formulation (1)-(11).

Table 1: Description of the single ship test problems

Instance code	*H*	*C*	*R*	*T*	No. of constraints	No. of variables
SS1-1	10	2	1	61	8,184	2,501
SS1-2	10	3	1	61	15,687	3,721
SS1-3	10	4	1	61	25,508	4,941
SS1-4	10	5	1	61	37,647	6,161
SS1-5	10	2	3	61	8,184	2,501
SS2-1	20	3	3	106	55,882	12,826
SS2-2	20	4	4	106	90,968	17,066
SS2-3	20	3	5	106	55,882	12,826
SS3-1	30	3	8	143	114,001	25,883
SS4-1	40	2	8	198	109,732	31,878
SS4-2	40	3	8	198	211,306	47,718
SS5-1	50	3	8	249	◊	◊
SSP-1	25	2	8	519	178,561	52,419
SSP-2	30	3	8	839	◊	◊
SSP-3	40	4	8	1,385	◊	◊

◊ Unknown – insufficient memory to load the problem into CPLEX

For each problem instance, 3 different experiments were carried out: the compact formulation solved by CPLEX, the heuristic presented in Section 2.2, and branch-and-price (B&P). Computational results of all these tests are shown in Table 2.

CPLEX obtained the optimal vessel makespan values for 11 out of the 12 small problem instances, with the last one failing due to insufficient memory. The branch-and-price algorithm was always able to find an optimal solution. The heuristic, on the other hand, generates optimal solutions for 9 of the 12 small test cases, with the rest having an optimality gap of at most 6.1%. For the realistic problem set, CPLEX fails to execute for the last two instances while branch-and-price finds an optimal solution. The heuristic was actually able to generate optimal solutions for all 3 realistic test cases. The overall quality of the heuristic solutions is high.

In terms of the computational time, the heuristic is the fastest one, as expected. Branch-and-price is also extremely fast with SS5-1 the only instance requiring more than a minute. In problem instances SS1-4, SS2-2, and SS3-1 integer solutions were obtained at the root node and no branching was necessary, thus computation time is almost negligible in these cases. As for the heuristic, the computational time never ex-

ceed 20 seconds even for the largest test case, SSP3. It is clear that branch-and-price significantly outperforms CPLEX. This is particularly pronounced in instances SS4-2 and SSP1. Note also the low running time of branch-and-price for all three realistic instances.

Table 2: Comparison of the three algorithms for the single ship instances

Instance code	Optimal makespan	Heuristic gap above optimal	CPLEX runtime (secs)	Heuristic runtime (secs)	B&P runtime (secs)	B&P no. of nodes
SS1-1	31	0	0.87	0.09	0.15	13
SS1-2	21	0	0.98	0.08	0.19	9
SS1-3	16	0	1.10	0.08	0.14	7
SS1-4	14	0	1.02	0.09	0.10	1
SS1-5	31	0	1.13	0.01	0.02	3
SS2-1	36	0	0.84	0.31	1.09	15
SS2-2	38	5.26	11.58	0.50	0.51	1
SS2-3	36	0	9.67	0.36	0.64	4
SS3-1	56	0	6.54	1.04	1.05	1
SS4-1	99	0	25.89	1.83	4.53	9
SS4-2	66	6.06	368.70	1.82	20.66	43
SS5-1	83	3.61	◊	2.99	105.29	99
SSP1	260	0	808.09	2.87	3.83	20
SSP2	317	0	◊	6.99	8.02	12
SSP3	412	0	◊	18.42	24.11	6

◊ Unknown – insufficient memory to load the problem into CPLEX

Next we demonstrate how changes in the number of bays, QCs, and time periods effect the running times. By default we set T=125, C=2, H=20, r=2, and next we vary them one by one. Both exact algorithms (CPLEX and branch-and-price) were always able to find an optimal solution. In the left figure in Figure 4 we range the number of bays H from 10 to 40 and we record the running times of the three algorithms. It is clear that this has a significant impact on CPLEX, but, on the other hand, negligible effect on the remaining two algorithms. The running time of branch-and-price barely increases with the increased number of bays.

In the right chart in Figure 4 we change the number of time periods. Due to large running times of CPLEX in comparison with branch-and-price, we report the running times on the logarithmic scale. The trend is very similar to the previous case. As the number of time periods increases, the running times of branch-and-price and the heuristic barely increase, while the increases in CPLEX are significant. For T=200, branch-and-price requires less than a second and CPLEX 167 seconds.

The impact of the number of QCs is depicted in Figure 5. The running times are shown on the logarithmic scale. To the contrary with the previous two cases, the running time of branch-and-price now increases with the increased number of QCs. CPLEX exhibits similar behavior. Nevertheless, the difference in the running time between these two algorithms is enormous for five QCs: CPLEX requires 447 seconds and branch-and-price a mere 45 seconds. The running time of the heuristic does not increase significantly.

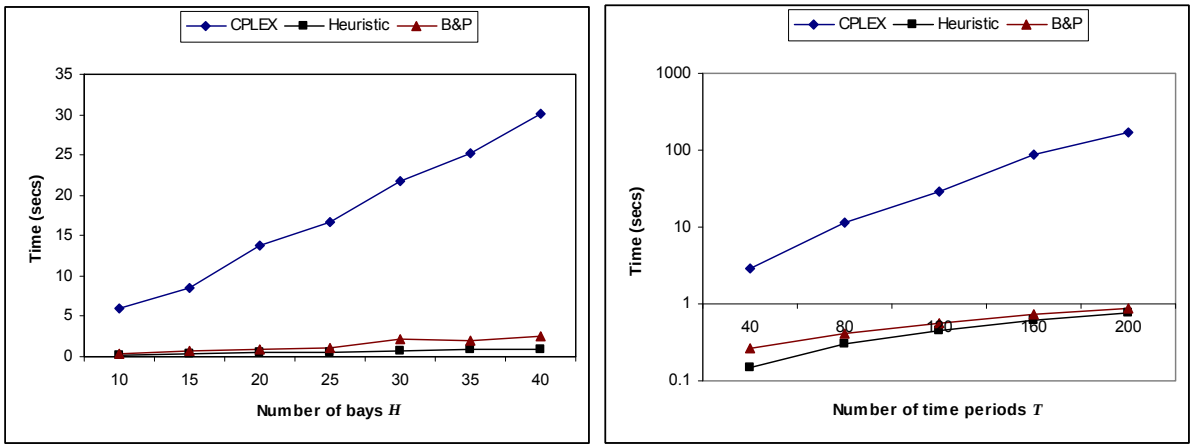

Figure 4: Sensitivity of H and T with respect to the running time

We can clearly conclude that increasing any of the three parameters increases significantly the running time of CPLEX. On the other hand, branch-and-price is robust with respect to the number of bays and time periods. Its running time increase is noticeable only with the increased number of QCs. As expected, the running of the heuristic is not significantly influenced by neither parameter.

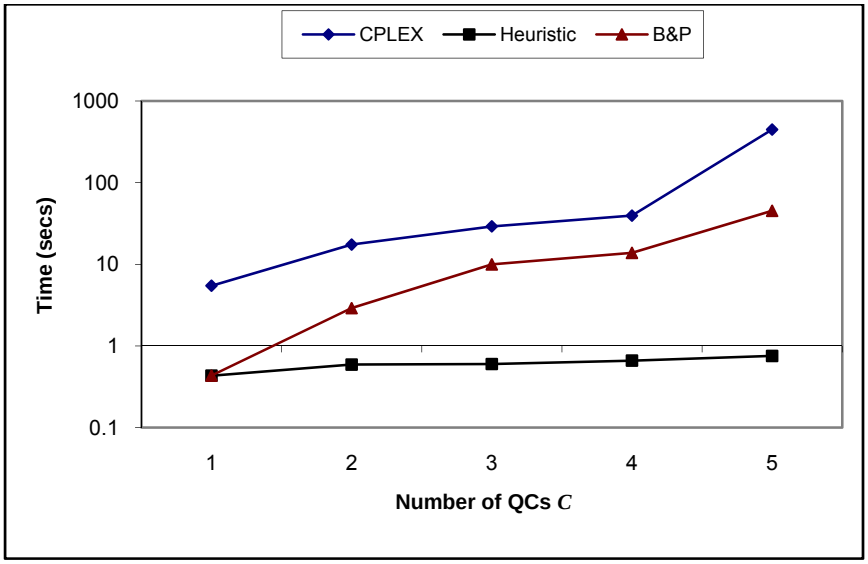

Figure 5: Sensitivity of C with respect to the running time

While Table 2 and the corresponding figures provide a thorough treatment of the computational results, they are unable to demonstrate how QC spatial constraints are met. An interface is written in Matlab to display a graphical and intuitive form of the solution. A graph of the vessel is plotted in Figure 6 with the bay location on the *x*-axis and time period on the *y*-axis.

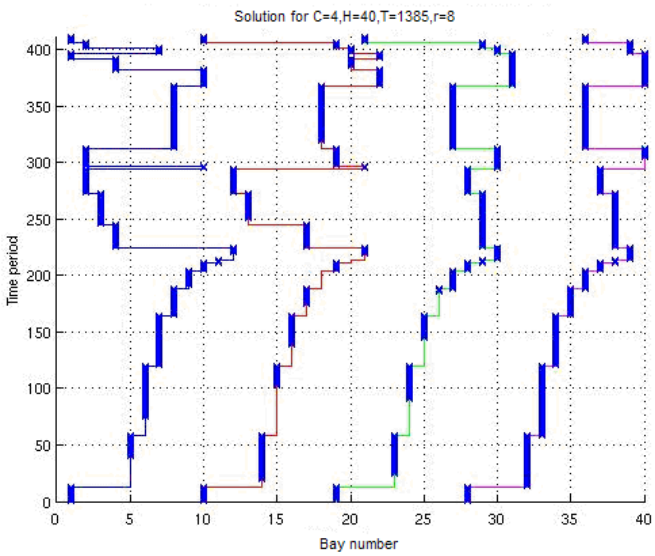

Figure 6: Branch-and-price solution of SSP3, with all QC spatial constraints being met

5.2. The Multi-ship Problem

It remains to discuss the multi-ship problem. A set of 7 different problem instances was created, arranged in the order of the increasing number of ships S, which varies from 2 to 8. They are described in Table 3 for 8 yard locations. The ships are weighted equally with respect to their importance towards makespan. Again, the container workload is randomly distributed among the vessel bays. In these problem instances we have an overlap in the berthing periods of the vessels (i.e., $[d_i, T_i]$ intervals defined in Appendix A overlap), so that the yard congestion constraints are binding.

Three further instances were created on each problem instance, labeled as 'easy', 'moderate' and 'hard'. The level of difficulty is determined by the value $\text{of } w_z$. In 'easy' cases the largest value of w_z is 8, for moderate cases the largest value is 2, and in hard cases it is always 1. For easy problems the yard congestion constraints barely restrict the feasible region. Often the primal heuristic performs no work at all as no infeasibilities are detected from the Lagrangian solutions. In contrast, for 'moderate' and 'hard' cases, much lower values of w_z are used. In Table 4 we compare CPLEX vs. our Lagrangian approach. Recall that L is the number of yard locations.

The column 'CPLEX makespan' shows the deviation of the CPLEX solution with respect to the best solution obtained by our Lagrangian approach, which is shown in the column 'Lag. makespan'. The column 'Optimality gap' gives the optimality gap of the Lagrangian approach. For the 'easy' cases, a single Lagrangian iteration is needed since most of the yard congestion constraints are non binding. For all 7 problem instances, the subgradient algorithm runs faster than CPLEX. Both algorithms solve all of instances except the last one by CPLEX to provable optimality. We can clearly see that the Lagrangian approach outperforms CPLEX in all cases with respect to the running time, and clearly there is a big advantage of using Lagrangian for the last instance MSD8.

Table 3: Description of the multi-ship test problems

Instance code	S	H_i	C_i	r	d_i	T_i	No. of constraints	No. of variables
MSD2	2	10,10	2,2	2	1,4	32,44	18,423	9,196
MSD3	3	10,10,15	2,2,3	3	1,5,10	32,44,73	76,161	43,394
MSD4	4	10,10,15, 15	2,2,3,3	3	1,1,8,14	32,44,73, 50	110,715	63,694
MSD5	5	10,10,15, 15,20	2,2,3,3, 4	3	1,8,20, 40,65	32,44,73, 50,79	204,757	120,653
MSD6	6	10,10,15, 15,20,25	1,1,2,2, 2,3	5	1,8,20, 40,65,65	32,44,73, 50,79,89	231,990	128,932
MSD7	7	10,10,15, 15,20,25, 25	1,1,2,2, 2,3,3	6	1,8,20, 40,65,65, 41	32,44,73, 50,79,89, 111	359,729	203,968
MSD8	8	10,10,15, 15,20,25, 25,25	1,1,2,2, 2,3,3,3	6	1,8,20, 40,65, 65,41,10	32,44,73, 50,79,89, 111,80	451,849	258,048

For the ‘moderate’ cases, the Lagrangian approach solves 4 out of the 7 instances to optimality. A maximum of 4 subgradient iterations are required to reduce the duality gap below 1%. In these cases moderate infeasibilities in the Lagrangian solutions are easily corrected by the primal heuristic to produce high quality upper bound values. Instance MSD4 is the only instance where CPLEX outperforms our Lagrangian approach. Note that for this instance the running time of CPLEX is lower, and, even though the computed makespan is the same for both algorithms, CPLEX has no optimality gap at the end. The last two instances are particularly interesting since CPLEX runs out of memory while the Lagrangian approach finds solutions within 1% in less than 5 minutes.

For the ‘hard’ cases, CPLEX outperforms the Lagrangian approach only for the first instance. Both algorithms find the same solution, however Lagrangian requires much more time and it does not establish provable optimality. In the remaining 7 instances Lagrangian clearly dominates CPLEX. A maximum duality gap of 16.07% (in instance MSD4) is found among all instances. The running times are much longer than in the ‘easy’ and ‘medium’ cases mostly due to a larger number of subgradient iterations. In MSD3 and MSD4 CPLEX finds inferior makespan within a time limit of one hour, which is particularly pronounced in the MSD4 instance.

It is clear that the Lagrangian approach substantially outperforms CPLEX. Out of 24 instances only in two instances CPLEX fared better.

Figure 7 shows the gap between the lower and the upper bound with the subgradient algorithm as the iterations progress. The standard tailing effect is observed, in particular with respect to the Lagrangian dual lower bound.

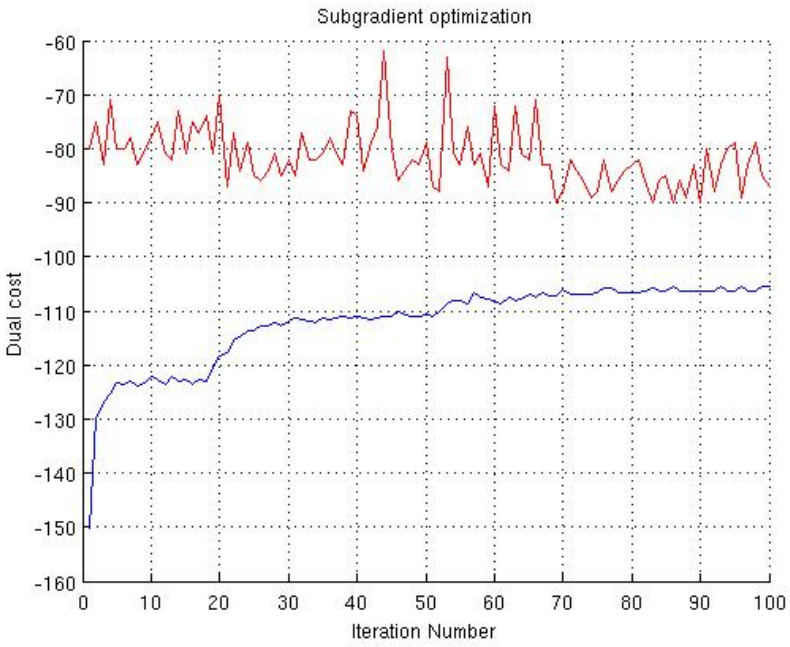

Figure 7: Lower and upper bounds of subgradient optimization for MSD4 (hard case)

Table 4: Comparison of CPLEX vs. our Lagrangian approach for multi-ship instances

Level of difficulty	Instance code	*L*	CPLEX makespan	Lag. makespan	Optimality gap	CPLEX runtime (secs)	Lag. runtime (secs)	No. of Lag. iterations
	MSD2	5	0	40	0	3	2	1
	MSD3	8	0	74	0	15	11	1
	MSD4	8	0	106	0	25	17	1
'Easy'	MSD5	8	0	162	0	60	39	1
	MSD6	8	0	146	0	171	121	1
	MSD7	8	0	210	0	473	242	1
	MSD8	8	⌂	262	0	⌂	295	1
	MSD2	5	0	40	0	8	4	2
	MSD3	8	0	74	0	17	24	2
	MSD4	8	0	106	0.97%	53	71	4
'Moderate'	MSD5	8	0	162	0	71	39	2
	MSD6	8	0	146	0	242	121	2
	MSD7	8	⌂	210	0.96%	⌂	240	1
	MSD8	8	⌂	260	0.77%	⌂	294	1
	MSD2	5	0	39	2.68%	13	2,743	85
	MSD3	8	+2◊	71	5.42%	3,600◊	3,003	100
	MSD4	8	+14◊	91	16.07%	3,600◊	3,543	100
'Hard'	MSD5	8	⌂	158	2.02%	⌂	3,295	69
	MSD6	8	⌂	141	3.86%	⌂	5,976	97
	MSD7	8	⌂	192	12.26%	⌂	3,342	20Δ
	MSD8	8	⌂	246	15.41%	⌂	4,332	20Δ

◊ CPLEX time limit of 1 hour exceeded
⌂ insufficient memory to run the problem with CPLEX
Δ subgradient terminated prematurely due to high running time per iteration

6. Conclusions

In this work we have articulated formal models and algorithms for the single and multi-ship models with the latter taking into account the possibility of congestion forming in the yard from high loading and discharging activities. We have also incorporated the important QC clearance requirements into our models. For the single-ship model a novel branch-and-price approach is developed, which solves the single ship model to optimality. This is used in lieu of the direct approach by solving the problem using a commercial solver. Branch-and-price saves a tremendous amount of computational effort. For the multi-ship model we proposed a method based on a combination of Lagrangian relaxation, the subgradient algorithm, and a primal heuristic. We noted that relaxing the yard congestion constraints allows the separation of the original problem into independent subproblems having a similar structure as the single ship model. This created the opportunity to develop a modified branch-and-price method, originally proposed for the single ship model, to solve the Lagrangian relaxation in each iteration of the subgradient algorithm. Good primal solutions were achieved by effectively utilizing information generated by Lagrangian relaxation. Computational results varied considerably depending on the value of the yard activity threshold. When this value was high, no duality gap was observed. However, for lower threshold values we have to tolerate manageable levels of suboptimality. Nevertheless, in all cases our proposed Lagrangian relaxation technique is the recommended approach in terms of computational efficiency and solution quality for solving multi-ship problems.

Appendix A: The Multi-ship Formulation

The following notation is used in the multi-ship formulation.

Indices:

i Ship number, in no particular berthing order;
j Bay number, in increasing order of their relative location on the vessel (i.e., left to right);
k QC number, in increasing order of their relative position (i.e., left to right);
z Yard storage location number;
t Time period index, denoting the interval of time from t-1 to t;

Parameters:

S Number of ships berthing during the master planning horizon;
Q_t The set of ships berthed at time t;
C_i Number of QCs allocated to ship i;
H_i Number of bays in ship i;
L Number of container storage locations in the yard;
f_{ijz} Number of containers to be discharged from bay j of ship i headed for storage location z;
d_i Berth time of ship i; vessel i cannot be handled before this time;
c_i The per time unit cost of ship i during discharging;

T_i Number of time periods in the individual planning horizon of ship i; (it can be set to $\sum_{j=1}^{H}\sum_{z=1}^{L} f_{ijz}$);

T_{max} Total number of time periods in the master planning horizon, i.e., $\max_i\{d_i + T_i - 1\}$;

r QC clearance value for all vessels, in terms of the number of bays;

w_z Yard activity threshold; it is the maximum number of QCs allowed to work on containers headed for storage location z at any time;

Decision variables:

$x_{ijk}(t)$ 1 if QC k is positioned at bay j of ship i at time period t and 0 otherwise;

$\delta_{ijkz}(t)$ 1 if QC k is discharging a container headed for storage location z at bay j of ship i at time period t and 0 otherwise;

$\gamma_i(t)$ work completion flag: 0 if all container jobs in ship i have not yet been completed at time period t and 1 otherwise;

The formulation is as follows.

$$\text{Minimize } -\sum_{i=1}^{S}\sum_{t=d_i}^{d_i+T_i-1} c_i\gamma_i(t) \tag{45}$$

Subject to

$$\sum_{j=1}^{H_i} x_{ijk}(t) = 1 \qquad i=1,...,S; k=1,...,C_i; t=d_i,...,d_i+T_i-1 \tag{46}$$

$$\sum_{k=1}^{C_i} x_{ijk}(t) \le 1 \qquad i=1,...,S;\ j=1,...,H_i; t=d_i,...,d_i+T_i-1 \tag{47}$$

$$C_i\left(1-x_{ijk}(t)\right) \ge \sum_{l=\max\{1,\,j-r\}}^{j-1}\sum_{m=1}^{C_i} x_{ilm}(t) \qquad i=1,...,S;\ j=2,...,H_i; k=1,...,C_i; t=d_i,...,d_i+T_i-1 \tag{48}$$

$$C_i\left(1-x_{ijk}(t)\right) \ge \sum_{l=j+1}^{\min\{j+r,H_i\}}\sum_{m=1}^{C_i} x_{ilm}(t) \qquad i=1,...,S;\ j=1,...,H_i-1; k=1,...,C_i; t=d_i,...,d_i+T_i-1 \tag{49}$$

$$x_{ijk}(t) \le \sum_{l=j+1}^{H} x_{il,k+1}(t) \qquad i=1,...,S;\ j=1,...,H_i-1; k=1,...,C_i-1; t=d_i,...,d_i+T_i-1 \tag{50}$$

$$x_{ijk}(t) \le \sum_{l=1}^{j-1} x_{il,k-1}(t) \qquad i=1,...,S;\ j=2,...,H_i; k=2,...,C_i; t=d_i...d_i+T_i-1 \tag{51}$$

$$x_{ijk}(t) = 0 \qquad i=1,...,S; k=2,...,C_i;\ j=1,...,k-1; t=d_i,...,d_i+T_i-1 \tag{52}$$

$$x_{ijk}(t) = 0 \qquad i=1,...,S; k=1,....,C_i-1;\ j=H_i-(C_i-k)+1,...,H_i; t=d_i,...,d_i+T_i-1 \tag{53}$$

$$\sum_{k=1}^{C_i}\sum_{t=d_i}^{d_i+T_i-1} \delta_{ijkz}(t) = f_{ijz} \qquad i=1,...,S;\ j=1,...,H_i; z=1,...,L \tag{54}$$

$$\delta_{ijkz}(t) \le x_{ijk}(t) \qquad i=1,...,S;\ j=1,...,H_i; k=1,...,C_i; z=1,...,L; t=d_i,...,d_i+T_i-1 \tag{55}$$

$$\sum_{z=1}^{L} \delta_{ijkz}(t) \le 1 \qquad i=1,...,S;\ j=1,...,H_i; k=1,...,C_i; t=d_i,...,d_i+T_i-1 \tag{56}$$

$$\sum_{i\in Q_t}\sum_{j=1}^{H_i}\sum_{k=1}^{C_i}\delta_{ijkz}(t)\le w_z \qquad z=1,...,L;t=1,...,T_{\max} \tag{57}$$

$$\gamma_i(t)\le \frac{\sum_{k=1}^{C_i}\sum_{l=d_i}^{t}\delta_{ijkz}(l)}{f_{ijz}} \qquad i=1,...,S;\ j=1,...,H_i;z=1,...,L;t=d_i,...,d_i+T_i-1 \tag{58}$$

$$x,\delta,\gamma \text{ binary} \tag{59}$$

The individual planning horizon of vessel i ranges from d_i to $d_i + T_i$ - 1, and for each vessel the QC positions and work constraints are applied in every time period within this range. All of the QC position constraints (46)-(53) are similar to the single-ship model position constraints described in Section 2.1, and they are applied to each vessel, indexed by i. QC work constraints (54)-(56) are also similar except for the additional index for storage location. The added work constraint (56) ensures that a QC cannot discharge more than one container from a bay at any time.

Constraint (57) sums up the total amount of QC work performed on containers headed for a particular storage location z over all active vessels at a particular time period t. It ensures that it does not exceed the yard activity threshold w_z. The value w_z may be dissimilar among the different storage locations due to different types of yard cranes and the size of the traffic lanes. Each of the yard congestion constraints depends on location z and time t. For ease of referencing them they are denoted by (z,t).

Constraint (58) captures the work completion flag; containers in all storage locations in each bay of vessel i must be entirely handled before $\gamma_i(t)$ can be set to 1. The objective function evaluates the sum of the weighted makespan of each vessel. The makespan of vessel i in an optimal solution is $T_i - \sum_{t=d_i}^{d_i+T_i+1}\gamma_i(t)+1$.

Appendix B: The Lagrangian Subproblem Formulation

The Lagrangian subproblem reads

Minimize $L^i(\boldsymbol{\lambda})$

Subject to constraints (2)-(7) and

$$\sum_{t=d_i}^{d_i+T_i-1}\delta_{jz}(t)=f_{ijz} \qquad j=1,...,H_i;z=1,...,L$$

$$\delta_{jkz}(t)\le x_{jk}(t) \qquad j=1,...,H_i;k=1,...,C_i;z=1,...,L;t=d_i,...,d_i+T_i-1$$

$$\sum_{z=1}^{L}\delta_{jkz}(t)\le 1 \qquad j=1,...,H_i;k=1,...,C_i;t=d_i,...,d_i+T_i-1$$

$$\gamma(t)\le \frac{\sum_{k=1}^{C}\sum_{l=d}^{t}\delta_{jkz}(l)}{f_{jz}} \qquad j=1,...,H_i;z=1,...,L;t=d_i,...,d_i+T_i-1$$

$$x,\delta,\gamma \text{ binary.}$$

References

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., and Vance, P. (1998). Branch-and-Price: Column generation for solving huge integer programs. *Operations Research*, **46**, 316-329.

Bish, E. (2003). A multiple crane constrained scheduling problem in a container terminal. *European Journal of Operational Research*, **144,** 83-107.

Cheung, R., Li, C.-L., and Lin, W. (2002). Interblock crane deployment in container terminals. *Transportation Science*, **36**, 79-93.

Daganzo, C. (1989). The crane scheduling problem. *Transportation Research: Part B*, **23,** 159-175.

Fisher, M. (1985). An applications oriented guide to Lagrangian relaxation. *Interfaces*, **15**, 10-21.

Geoffrion, A. (1974). Lagrangean relaxation for integer programming. *Mathematical Study*, **2**, 82-114.

Held, M., and Karp, R. (1970). The travelling salesman problem and minimum spanning trees. *Operations Research*, **18**, 1138-1162.

Kim, K.H., and Kim, K.Y. (1999). An optimal routing algorithm for a transfer crane in port container terminals. *Transportation Science*, **33**, 17-33.

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

Lubbecke, M., and Descrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**, 1007-1023.

Murty K., Liu, J., Wan, Y., and Linn, R. (2005). A decision support system for operations in a container terminal. *Decision Support Systems*, **39**, 309-332.

Peterkofsky, R., and Daganzo, C. (1990), A branch-and-bound solution method for the crane scheduling problem. *Transportation Research: Part B*, **24**,159-172.

Steenken, D., Voß, S., and Stahlbock, R. (2004). Container terminal operation and operations research – a classification and literature overview. *OR Spectrum*, **26**, 3-49.

Wolfe, P., and Crowder, H. (1974), Validation of subgradient optimization. *Mathematical Programming*, **6**, 62-88.

Zhang, C. (2002). A heuristic for real-time container load sequencing. *Master's Thesis*, HPCES, Singapore-MIT Alliance.

Zhang, C., Wan, Y.-W., Liu, J., and Linn, R.J. (2002). Dynamic crane deployment in container storage yards. *Transportation Research: Part B*, **36**, 537–555.