A Branch-and-Cut Algorithm for the Quay Crane Scheduling Problem in a Container Terminal

Luigi Moccia[*][†], Jean-François Cordeau[*],
Manlio Gaudioso[†] and Gilbert Laporte[*]

April 4, 2005

Abstract

The quay crane scheduling problem consists of determining a sequence of unloading and loading movements for cranes assigned to a vessel, in order to minimize the vessel completion time as well as the crane idle times. Idle times originate from interferences between cranes since these roll on the same rails and a minimum safety distance must be maintained between them. The productivity of container terminals is often measured in terms of the time necessary to load and unload vessels by quay cranes, which are the most important and expensive equipment used in ports. We formulate the quay crane scheduling problem as a vehicle routing problem with side constraints, including precedence relationships between vertices. For small size instances our formulation can be solved by CPLEX. For larger ones we have developed a branch-and-cut algorithm incorporating several families of valid inequalities which exploit the precedence constraints between vertices.

Keywords: *maritime container terminal; quay crane scheduling; branch-and-cut.*

[*]Canada Research Chair in Distribution Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

[†]Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, 87036 Rende (CS) - Italy.

1 Introduction

In maritime transportation the use of containers for general cargo has steadily increased over the last 20 years. Containers are large metal boxes made in standard dimensions and measured in multiples of twenty feet called "twenty foot equivalent units" (TEUs). The world container port throughput for 2002 reached 266.3 million TEUs, an increase of 22.5 million TEUs from the level of 243.8 million TEUs reached in 2001. For statistics on maritime transport we refer the reader to UNCTAD (2004). In this industry the hub and spoke arrangement is widely adopted. Deep sea vessels, also called mother vessels, operate between a limited number of transshipment terminals (hubs). Smaller vessels (feeders) link the hubs with the other ports (spokes). This network topology results in the consolidation of capacity along the routes linking the hubs and in their growth. In recent years, mother vessels have strongly increased in size, attaining up to 8000 TEUs. Transshipment ports are large intermodal platforms, and a limited number of them handle an important share of the world traffic. Thus, in 2002 the first 20 container ports handled 48% of the total traffic. Ultra-large container vessels cut down transport cost. However, hub ports are forced to invest heavily to accommodate these ships by deepening and widening channels and constructing new berthing services of sufficient depth and length. These trends require a continuous improvement in managerial practices at transshipment terminals which can be viewed as large material handling systems. Advanced communication and information technologies already exist and the next step will be the introduction of customized optimization techniques. Customization is a key factor since maritime terminals differ from each other in their layouts and material handling equipments. The need for an optimal management of logistics activities at modern container terminals is well recognized. For a recent overview and classification of the various equipments and decision problems in such systems, see Vis and Koster (2003) and Steenken et al. (2004). A review of operational research issues in maritime logistics, focused on ship routing and scheduling, was presented by Christiansen et al. (2004).

The productivity of container terminals is often measured in terms of the time necessary to load and unload vessels by quay cranes (QCs), which are the most important and expensive equipment used in ports. The *Quay Crane Scheduling Problem* (QCSP) consists of determining a sequence of unloading and loading movements for cranes assigned to a vessel in order to minimize the vessel completion time as well as the crane idle times. Idle times originate from interferences between cranes since these roll on the same rails, and thus cannot cross each other, and from the need to maintain a minimum safety distance between them.

In our study we consider as an input the number of quay cranes assigned to the vessels. Of course this decision affects the vessel completion time. In a complex system like a transshipment port the decision making process is often hierarchical, and the quay cranes assignment problem is solved before the QCSP. The number of quay cranes assigned to a vessel often depends on contracts between the terminal and shipping companies. The input data for the QCSP consists of the vessel stowage plan, of the loading plan, of the ready time of each crane, and of a yard map showing the storage locations of containers to be loaded on the vessel.

A first classification of the containers to be handled in a given ship bay is made according to their positions, on the deck or in the hold, and of their kind of operations: loading or unloading. Unloading always precedes loading: the deck is unloaded before the hold, the hold is then loaded, then the deck. Containers having the same size, origin and destination are assigned to neighbor slots in the vessel to simplify operations at the origin and destination ports. Containers located in a given ship bay can be divided into different tasks when they share the following attributes: ship bay number, position (deck or hold), operation (loading or unloading), size, destination port for outbound containers, or origin port for inbound containers. We assume that quay cranes have the same productivity for the same task, expressed in moves per hour. In the machine scheduling terminology, they can be viewed as identical machines executing non-preemptable tasks. Precedence relationships between tasks are the result of their position in the ship and of their associated operation type, as described above. Other precedence constraints can be imposed to take into account vessel load balance during the operations. Given the task positions in the vessel layout and the quay crane characteristics, certain pairs of tasks cannot be performed simultaneously. A safety distance between movements must be observed to avoid collisions, which translates into additional constraints between pairs of tasks. Depending on the yard layout and technology, some tasks may not be executed simultaneously to avoid workload peaks in specific areas of the yard. Container terminals in the Asia-Pacific region rely on the "Indirect Transfer System" (ITS) in which containers in the yard are stacked in compact sections. A dedicated gantry crane, called transtainer, moves them from and to the vehicles which link the yard with the quay and the gates. The ITS minimizes yard surface requirements, but poses specific constraints on the QCSP because tasks that interact with the same transtainer cannot be performed simultaneously. European and North-American container terminals are based on the "Direct Transfer System" (DTS) in which vehicles (e.g. straddle carriers) used to transfer containers between the yard, the quay and the gates are also capable of moving them between the slots in the yard. The DTS requires a larger surface because dedicated lanes in the yard are required for the vehicles to access the slot positions. However, in this case, the quay crane scheduling interaction with the yard is less critical.

Our purpose is to model the QCSP as a mixed integer linear program and to solve it exactly by means of a branch-and-cut algorithm. The remainder of the article is organized as follows. The next section briefly reviews relevant work on the QCSP and closely related problems. Section 3 defines the QCSP formally and introduces a mixed integer linear formulation. Section 4 describes the valid inequalities used in the branch-and-cut algorithm which is then described in Section 5, along with separation heuristics and preprocessing techniques. Computational experiments are reported in Section 6, followed by conclusions in Section 7.

2 Literature Review

The QCSP was first introduced by Daganzo (1990) who assumes the ships to be partitioned into bays and defines a task as the loading and unloading of all containers of a given bay. As a result, no precedence constraints are considered between groups of containers in the same bay. A task is preemptable since it involves different operations which, in our study, are represented as different tasks. Cranes can be moved freely, but no interference among QCs is modelled, and more than one vessel is considered. The paper presents an algorithm for determining the number of cranes to be assigned to bays of multiple vessels. Under similar assumptions, Peterkofsky and Daganzo (1990) provide a branch-and-bound algorithm to determine the departure times of multiple vessels and the number of cranes to be assigned to individual ship bays in a specific time interval. The objective function to be minimized is the sum of the delay costs.

More recently Kim and Park (2004) have assumed that there may be multiple tasks associated with a ship bay, and thus the loading and unloading operations associated with the same bay are divided into different tasks. We have made the same assumptions in Section 1. Kim and Park present a mixed integer formulation of the QCSP which cannot be solved in reasonable time, and therefore the authors propose a reduction of the solution space, and then an exact method capable of solving small and medium size instances. For larger instances a heuristic algorithm based on a greedy randomized adaptive search procedure (GRASP) is used to quickly identify feasible solutions.

Lim et al. (2004) study the same multi-vessel QCSP as Daganzo (1990) but integrate spatial constraints (non-crossing and minimum safety distance between QCs), and non-simultaneity between tasks. However, precedence constraints are not considered and the scheduling problem is modelled without integrating it over time, i.e. it is seen as a static crane-to-task matching. A probabilistic tabu search and a "squeaky wheel" optimization heuristic are applied to the problem, while dynamic programming algorithms can solve simplified variants.

The single crane version of the QCSP is a particular case of the *Precedence Constrained Traveling Salesman Problem* (PC-TSP) in which every vertex may have multiple predecessors and successors. The asymmetric PC-TSP has been studied, among others, by Balas et al. (1995) and Ascheuer et al. (2000) who proposed several families of valid inequalities based in large part on the strengthening of existing inequalities for the asymmetric TSP. This problem has applications, for example, in the scheduling of flexible manufacturing systems (see, e.g., Ascheuer 1996). Some of the proposed PC-TSP inequalities also apply to the QCSP. In particular, the predecessor and successor inequalities proposed by Balas et al. will be used in our algorithm. The QCSP also shares some characteristics with the *Vehicle Routing Problem with Pickup and Delivery* (VRP-PD) and the *Dial-a-Ride Problem* (DARP). However there is a difference between the *time precedence* constraint in the QCSP, where a QC can start a task only if the preceding tasks are already completed by *any* QC, and the *route precedence* constraint in the VRP-PD or the DARP, where a delivery node can be visited by a vehicle only after the pickup node has been visited by the *same* vehicle.

3 Mathematical models

We introduce our main notation in Section 3.1, followed by the Kim and Park (2004) mixed integer formulation in Section 3.2, by some observations in Section 3.3, and by an improved model in Section 3.4.

3.1 Notation

We are given a set of n tasks, $\Omega = \{1, ..., n\}$, and a set of q quay cranes, $K = \{1, ..., q\}$. A set Φ of ordered task pairs (i, j) describes the precedence relationships, i.e., i must precede j whenever $(i, j) \in \Phi$. Similarly tasks i and j cannot be performed simultaneously whenever $(i, j) \in \Psi$. Note that $\Phi \subseteq \Psi$. The input data for the model are:

$p_i, \forall i \in \Omega,$ the processing time of task i,

$r_k, \forall k \in K,$ the earliest available time of QC k,

$l_i, \forall i \in \Omega,$ the location of task i expressed by a ship bay number,

$l_k^0, \forall k \in K,$ the starting position of QC k expressed by a ship bay number,

$l_k^T, \forall k \in K,$ the final position of QC k expressed by a ship bay number,

$t,$ the travel time between two consecutive bays,

$t_{ij}, \forall i, j \in \Omega,$ the travel time of a QC from location l_i to location l_j, $t_{ij} = t \times |l_i - l_j|$,

$t_{0j}^k, \forall j \in \Omega, k \in K,$ the travel time of a QC k from location l_k^0 to location l_j, $t_{0j}^k = t \times |l_k^0 - l_j|$

$t_{iT}^k, \forall i \in \Omega, k \in K,$ the travel time of a QC k from location l_i to location l_k^T, $t_{iT}^k = t \times |l_i - l_k^T|$ ($t_{iT}^k = 0, \forall i \in \Omega$ means that we do not consider the quay crane travel time after the completion of the last task),

$\alpha_1,$ the weight assigned to the makespan component of the objective function,

$\alpha_2,$ the weight assigned to the total completion time component of the objective function.

We also define additional sets:

$\Phi^{'} = \{(i,j) \in \Phi : l_i = l_j\},$ the subset of precedences related to the loading, unloading, hold, deck priorities,

$\pi(i) = \{j \in \Omega : (j,i) \in \Phi\},$ a set of tasks that must be completed before i starts,

$\sigma(i) = \{j \in \Omega : (i,j) \in \Phi\},$ a set of tasks that can only start after i is completed,

$\pi^{'}(i), \sigma^{'}(i),$ defined as above but on the set $\Phi^{'}$ instead of Φ,

$\Upsilon(i,j) = \{l \in \Omega : l \in \pi^{'}(j), l \in \sigma^{'}(i)\},$ a set of tasks belonging to the same ship bay that can start only after i is completed and must be completed before j starts.

We denote by $\zeta(i,j)$ a lower bound on the difference between the starting time of a task j and the completion time of a task i which uses the precedence relationships inside a ship bay:

$$\zeta(i,j) = \begin{cases} 0 & \text{if } \Upsilon(i,j) = \emptyset \\ \sum_{l \in \Upsilon(i,j)} p_l & \text{otherwise} \end{cases}, \forall i,j \in \Omega.$$

Similarly $\zeta(0,j)$ represents a straightforward lower bound on the starting time of a task j when j has predecessors on the same ship bay:

$$\zeta(0,j) = \begin{cases} 0 & \text{if } \pi^{'}(j) = \emptyset \\ \sum_{l \in \pi^{'}(j)} p_l & \text{otherwise} \end{cases}, \forall j \in \Omega.$$

3.2 The Kim and Park formulation

In the Kim and Park formulation the problem is modelled on a graph $G = (V, A)$, where $V = \Omega \cup \{0, T\}$, 0 and T being initial and final states of the cranes and $A \subseteq V \times V$. For notational convenience define $\Omega^0 = \Omega \cup \{0\}$ and $\Omega^T = \Omega \cup \{T\}$. The formulation uses the following variables:

$x^k_{ij} \in \{0, 1\}\ \forall (i, j) \in A, k \in K,$ if QC k performs task j immediately after task i, then $x^k_{ij} = 1$; otherwise $x^k_{ij} = 0$; if $i = 0$ and $x^k_{ij} = 1$, then task j is the first task of QC k; similarly if $j = T$ and $x^k_{ij} = 1$, then task i is the last task of QC k;

$D_i\ \forall i \in \Omega,$ the completion time of task i;

$z_{ij} \in \{0, 1\}\ \forall i, j \in \Omega,$ if task j starts later than the completion time of task i then $z_{ij} = 1$, otherwise $z_{ij} = 0$;

$C^k\ \forall k \in K,$ the completion time of QC k;

$W,$ the makespan, i.e., the earliest time at which all QCs can complete their work.

The Kim and Park model, denoted by $F1$, is as follows:

$$\text{minimize } \alpha_1 W + \alpha_2 \sum_{k \in K} C^k \tag{1}$$

subject to

$$C^k \leq W \qquad \forall k \in K, \tag{2}$$

$$\sum_{j \in \Omega} x^k_{0j} = 1 \qquad \forall k \in K, \tag{3}$$

$$\sum_{i \in \Omega} x^k_{iT} = 1 \qquad \forall k \in K, \tag{4}$$

$$\sum_{k \in K} \sum_{i \in \Omega} x^k_{ij} = 1 \qquad \forall j \in \Omega, \tag{5}$$

$$\sum_{j \in \Omega^T} x^k_{ij} - \sum_{j \in \Omega^0} x^k_{ji} = 0 \qquad \forall i \in \Omega, \forall k \in K, \tag{6}$$

$$D_i + t_{ij} + p_j - D_j \leq M(1 - x^k_{ij}) \qquad \forall i, j \in \Omega, \forall k \in K, \tag{7}$$

$$D_i + p_j \leq D_j \qquad \forall (i,j) \in \Phi, \tag{8}$$

$$D_i + p_j - D_j \leq M(1 - z_{ij}) \qquad \forall i, j \in \Omega, \tag{9}$$

$$z_{ij} + z_{ji} = 1 \qquad \forall (i,j) \in \Psi, \tag{10}$$

$$\sum_{v=1}^{k} \sum_{u \in \Omega^0} x^v_{uj} - \sum_{v=1}^{k} \sum_{u \in \Omega^0} x^v_{ui} \leq M(z_{ij} + z_{ji}) \qquad \forall i, j \in \Omega, l_i < l_j, \forall k \in K, \tag{11}$$

$$D_j + t^k_{jT} - C^k \leq M(1 - x^k_{jT}) \qquad \forall j \in \Omega, \forall k \in K, \tag{12}$$

$$r_k - D_j + t^k_{0j} + p_j \leq M(1 - x^k_{0j}) \qquad \forall j \in \Omega, \forall k \in K, \tag{13}$$

$$x^k_{ij} \in \{0, 1\} \qquad \forall k \in K, \ \forall (i,j) \in A, \tag{14}$$

$$z_{ij} \in \{0, 1\} \qquad \forall k \in K, \ \forall i, j \in \Omega, \tag{15}$$

$$C^k, D_i \geq 0 \qquad \forall i \in \Omega, \forall k \in K, \tag{16}$$

where M is a sufficiently large constant.

The objective function (1) minimizes a weighted sum of W, the vessel completion time or makespan, and of $\sum_{k \in K} C^k$, the sum of the quay crane completion times. The two parts of the objective function will lead to the same solution if there is only one quay crane, since in this case $W = C^1$. However, when multiple quay cranes are available, minimizing over W or over $\sum_{k \in K} C^k$ usually results in different scheduling decisions. A quay crane completion time is made up of idle times, travelling times and handling times. Since we assume deterministic handling times, $\sum_{k \in K} C^k$ will have a constant part, the sum of the task handling times, and a variable part, the sum of the travelling and idle times, which depends on the scheduling decision. Among solutions with the same makespan the terminal management is interested in solutions that also minimize quay crane idle times in order to increase the utilization factor of this expensive material handling equipment. To this end we can set $\alpha_1 >> \alpha_2$, since reducing the makespan is more important. We also note that quay crane idle times are prominent with respect to the quay crane travelling times.

The makespan is defined by constraints (2). Constraints (3) and (4) ensure, respectively, that each crane k leaves its initial state 0 and ends at its final state T. Each task j is assigned to one and only one crane k by constraints (5). Constraints (6) define the flow balance for each task and each crane. Constraints (7) determine the completion time for each task and eliminate subtours. Precedence relationships are enforced by constraints (8). Constraints (9) define variables z_{ij}. Constraints (10) guarantee that tasks i and j cannot be performed simultaneously when $(i, j) \in \Psi$. The non-crossing requirement among QCs is modelled by constraints (11). If tasks i and j, with $l_i < l_j$, are performed simultaneously, then $z_{ij} + z_{ji} = 0$. Assuming that both QCs and tasks are ordered in increasing order of their relative bay locations in the ship, if $k_1 < k_2$, QC k_1 performs tasks j and QC k_2 performs task i, then a crossing between the two quay cranes will take place. However, in such a case, $\sum_{v=1}^{k} \sum_{u \in \Omega^0} x_{uj}^v - \sum_{v=1}^{k} \sum_{u \in \Omega^0} x_{ui}^v = 1$, which cannot be allowed because of constraints (11) and the fact that $z_{ij} + z_{ji} = 0$. The completion time of each QC is defined by constraints (12), while constraints (13) enforce time variables associated with the first task of each QC.

3.3 Observations on the Kim and Park formulation

In this section we present several strengthenings of formulation $F1$.

3.3.1 Consistency of the variable z_{ij}

If task j starts later than the completion time of task i, then $z_{ij} = 1$. In this situation the left-hand side of (9) is negative and the constraint is satisfied for $z_{ij} = 0$. The formulation can therefore be strengthened by adding the following constraints:

$$D_j - p_j - D_i \leq M z_{ij} \qquad \forall i, j \in \Omega. \tag{17}$$

To avoid QC collisions constraints (9) and (17) need some refinements. For example, consider two tasks i and j on the same bay, i.e. $l_i = l_j$. Collisions between QCs working on tasks belonging to the same bay are avoided because of constraints (10) forcing the non-simultaneity of the associated tasks. Suppose that task i is completed at time D_i by QC k which leaves the bay moving left; also suppose that QC $h, h > k$, has already completed a task u in a bay located right to the bay where are located i and j. If $t_{uj} \leq D_i - D_u$ then the QC h can start task j as soon as QC k completes task i, i.e. $D_j - p_j = D_i$ and no interference is detected by the model since z_{ij} can take the value 1; however a collision does happen. Under this condition, task j should not start before time $D_i + t$ which will allow the QC that has processed task i to move safely to another bay, assuming that the safety distance between QCs is equal to the length of a ship bay. This can be achieved by forcing the variable z_{ij} to be equal to 0 if $D_j - p_j = D_i$ and the QC processing j comes from a neighbor bay. Then, when $l_i = l_j$, constraints (9) and (17) should be restated as follows:

$$D_i + p_j - D_j + \sum_{k \in K} \sum_{u \in \Omega^0, l_u \neq l_i} tx_{uj}^k \leq M(1 - z_{ij}) \qquad \forall i, j \in \Omega, l_i = l_j, \tag{18}$$

$$D_j - p_j - D_i - \sum_{k \in K} \sum_{u \in \Omega^0, l_u \neq l_i} tx_{uj}^k \leq M z_{ij} \qquad \forall i, j \in \Omega, l_i = l_j. \tag{19}$$

We note also that precedence constraints (8) can be replaced by the following variable fixing:

$$z_{ij} = 1, z_{ji} = 0 \qquad \forall (i, j) \in \Phi. \tag{20}$$

And, because $\Phi \subseteq \Psi$, we can avoid writing constraints (10) when $(i, j) \in \Phi$:

$$z_{ij} + z_{ji} = 1 \qquad \forall (i, j) \in \Psi \setminus \Phi. \tag{21}$$

When the precedence between two tasks i and j is transitively derived, the associated precedence constraint between them is unnecessary, but to improve the linear relaxation, we can incorporate in constraints (18) a lower bound of the time between the completion of the task i and the start of the task j whenever $\Upsilon(i, j) \neq \emptyset$. Note that in this case $(i, j) \in \Phi$, $z_{ij} = 1$ and the right-hand side of (18) is equal to zero. We can add to the left-hand side of (18) the quantity $\zeta(i, j)$ which, as mentioned above, defines a lower bound on the difference between the starting time of a task j and the completion time of a task i. Furthermore we can also add the quantity $\sum_{k \in K} \sum_{u \in \Omega^0, l_u \neq l_i} \sum_{l \in \Upsilon(i,j)} tx_{ul}^k$ representing a lower bound on the QC waiting times between the starting of the tasks in $\Upsilon(i, j)$. For notational convenience let $\Omega_i = \{u \in \Omega^0 : l_u \neq l_i\}$ and $\tilde{\Upsilon}(i, j) = \{\Upsilon(i, j) \cup \{j\}\}$. Then constraints (18) become:

$$D_i + p_j + \zeta(i, j) - D_j + \sum_{k \in K} \sum_{u \in \Omega_i} \sum_{l \in \tilde{\Upsilon}(i,j)} tx_{ul}^k \leq M(1 - z_{ij}) \qquad \forall i, j \in \Omega, l_i = l_j. \tag{22}$$

3.3.2 Non-crossing constraints

The large constant M is unnecessary in constraints (11) since the maximum value of the left-hand side is equal to 1. Moreover constraints (11) are only relevant when $z_{ij} + z_{ji} = 0$, and we can therefore avoid writing these constraints for $(i, j) \in \Psi$. We also note that in constraints (11) the term $\sum_{v=1}^{k} \sum_{u\in\Omega^0} x_{ui}^v$ can be written as $\sum_{v=1}^{k-1} \sum_{u\in\Omega^0} x_{ui}^v$ because if task j is assigned to QC k and $z_{ij} + z_{ji} = 0$ (relevant case of simultaneity between two tasks), then task i belongs to a QC h with $h \neq k$. Since $\sum_{v=1}^{k} \sum_{u\in\Omega^0} x_{ui}^v \geq \sum_{v=1}^{k-1} \sum_{u\in\Omega^0} x_{ui}^v$ this also constitutes a strengthening. The non-crossing constraints (11), since $\sum_{u\in\Omega^0} \sum_{v=1}^{k-1} x_{ui}^v = 1 - \sum_{u\in\Omega^0} \sum_{v=k}^{q} x_{ui}^v$, can therefore be rewritten as follows:

$$\sum_{v=1}^{k} \sum_{u\in\Omega^0} x_{uj}^v + \sum_{v=k}^{q} \sum_{u\in\Omega^0} x_{ui}^v \leq 1 + z_{ij} + z_{ji} \quad \forall i, j \in \Omega, l_i < l_j, (i, j) \notin \Psi, \forall k \in K. \tag{23}$$

3.3.3 A lower bound for the task starting time

A trivial lower bound for the starting time of a task i is $d_{0i} = \min_k\{r_k + t_{0i}^k\}$. Taking into account the precedence relationships inside a ship bay leads to some refinements. The lower bound $a(i)$ on the starting time of a task i can be computed recursively as:

$$a(i) = \max\{d_{0i} + \zeta(0, i), \max_{j\in\pi(i)\setminus\pi'(i)} (a(j) + p_j)\}. \tag{24}$$

The value $a(i)$ is always defined since the precedence graph is acyclic, for otherwise the problem would not be feasible. Using $a(i)$ we can replace M in constraints (13) with $r_k + t_{0j}^k - a(j)$.

3.3.4 An upper bound for the task completion time

An upper bound on task completion time can be used instead of M in equations (7), (9) and (12). The upper bound for the task completion time corresponds to the upper bound for the makespan, minus the travelling time t^k_{iT}. We note that computing an upper bound on task completion time with a constructive heuristic will not be valid since the objective function is the weighted sum of the makespan and of the sum of the QC completion times. The sum of the QC completion times corresponds to a constant part, $\sum_{i \in \Omega} p_i$, plus the QC travelling times and the QC idle times caused by the interferences between cranes. In any practical case the idle times are much larger than the travelling times, and therefore we can conclude that the sum of the QC completion times decreases when interferences between cranes are reduced. A larger makespan may yield a reduced QC interference, and therefore the two elements of our objective function are conflicting. Anyhow, when only one QC is allowed on a vessel, the interference waiting times are eliminated by definition, and the corresponding makespan U is easy to compute. Using this value as an upper bound on task completion time does not cut off any feasible solution. The makespan upper bound is set equal to $\min_k \{r_k + \sum_{i \in \Omega} p_i + T^k\}$, where T^k is a feasible solution value for the TSP associated with QC k over all tasks. Since the quay crane moves along a line this TSP instance is immediately solved.

3.4 Formulation $F2$

We now introduce a new formulation for the QCSP which integrates the observations made in Section 3.3. To define branching priorities and simplify the equations we introduce new variables $y_{ik} \in \{0, 1\}, \forall i \in \Omega, k \in K$ and $y_{ik} = 1$ if and only if task i is assigned to QC k, and $y_{ik} = 0$ otherwise. Formulation $F2$ has the same objective function (1) and is subject to constraints (2), which define the makespan, plus the following:

$$\sum_{j\in\Omega^T} x^k_{0j} = 1 \qquad \forall k \in K, \tag{25}$$

$$\sum_{i\in\Omega^0} x^k_{iT} = 1 \qquad \forall k \in K, \tag{26}$$

$$y_{ik} = \sum_{j\in\Omega^T} x^k_{ij} \qquad \forall i \in \Omega, \forall k \in K, \tag{27}$$

$$y_{ik} = \sum_{j\in\Omega^0} x^k_{ji} \qquad \forall i \in \Omega, \forall k \in K, \tag{28}$$

$$\sum_{k\in K} y_{ik} = 1 \qquad \forall i \in \Omega, \tag{29}$$

$$D_i + t_{ij} + p_j - D_j \le M^1_{ij}(1 - \sum_{k\in K} x^k_{ij}) \qquad \forall i, j \in \Omega, \tag{30}$$

$$D_i + p_j - D_j \le M^2_j(1 - z_{ij}) \qquad \forall i, j \in \Omega, l_i \neq l_j, \tag{31}$$

$$D_j - p_j - D_i \le M^3_i z_{ij} \qquad \forall i, j \in \Omega, l_i \neq l_j \tag{32}$$

$$\begin{aligned} &D_i + p_j + \zeta(i,j) - D_j \\ &+ \sum_{k\in K}\sum_{u\in\Omega_i}\sum_{l\in\tilde{\Upsilon}(i,j)} t x^k_{ul} \le M^4_{ij}(1 - z_{ij}) \qquad \forall i, j \in \Omega, l_i = l_j, \end{aligned} \tag{33}$$

$$D_j - p_j - D_i - \sum_{k\in K}\sum_{u\in\Omega_i} t x^k_{uj} \le M^5_{ij} z_{ij} \qquad \forall i, j \in \Omega, l_i = l_j, \tag{34}$$

$$z_{ij} = 1, z_{ji} = 0 \qquad \forall (i,j) \in \Phi, \tag{35}$$

$$z_{ij} + z_{ji} = 1 \qquad \forall (i,j) \in \Psi \setminus \Phi, \tag{36}$$

$$\sum_{v=1}^{k}\sum_{u\in\Omega^0} x^v_{uj} + \sum_{v=k}^{q}\sum_{u\in\Omega^0} x^v_{ui} \le 1 + z_{ij} + z_{ji} \qquad \forall i, j \in \Omega, l_i < l_j, (i,j) \notin \Psi, k \in K, \tag{37}$$

$$D_j + t^k_{jT} - C^k \le M^6_{jk}(1 - x^k_{jT}) \qquad \forall j \in \Omega, \forall k \in K, \tag{38}$$

$$r_k - D_j + t^k_{0j} + p_j \le M^7_{jk}(1 - x^k_{0j}) \qquad \forall j \in \Omega, \forall k \in K, \tag{39}$$

$$x^k_{ij} \in \{0,1\} \qquad \forall k \in K, \forall (i,j) \in A, \tag{40}$$

$$y_{ik}, z_{ij} \in \{0,1\} \qquad \forall k \in K, \forall i, j \in \Omega, \tag{41}$$

$$a(i) + p_i \le D_i \le U \qquad \forall i \in \Omega, \tag{42}$$

$$C^k \ge r_k \qquad \forall i \in \Omega, \forall k \in K, \tag{43}$$

where $M_{ij}^1 = U + t_{ij} - a(j)$, $M_j^2 = U - a(j)$, $M_i^3 = U - p_j - (a(i) + p_i)$, $M_{ij}^4 = U + \zeta(i,j) - a(j) + t \times |\tilde{\Upsilon}(i,j)|$, $M_{ij}^5 = U - p_j - (a(i) + p_i)$, $M_{jk}^6 = U + t_{jT}^k - r_k$, $M_{jk}^7 = r_k - a(j) + t_{0j}^k$.

Constraints (25) and (26) ensure, respectively, that each crane k leaves its initial state 0 and ends at its final state T, but unlike what is stated in constraints (3) and (4) of $F1$, a QC k is allowed to go from its initial state directly to its final state. This modification enlarges the solution space but allows the detection of instances in which a reduction of the number of QCs is beneficial. Constraints (27) and (28) define variables y_{ik}. Only one group of these constraints would have been necessary, but together with constraints (29) they also model the assignment of each task to one and only one crane and the flow balance for each task. Constraints (30) determine the completion time for each task and eliminate subtours. Constraints (31) - (35) have already been discussed in Section 3.3.1 The non-crossing requirement among QCs is modelled by constraints (37) as discussed in Section 3.3.2. The completion time and the earliest starting time of each QC are defined by constraints (38) and (39), respectively.

4 Valid Inequalities

We now describe several families of valid inequalities for the QCSP. All of these inequalities are redundant for model $F2$ but can strengthen its LP relaxation.

The following additional notation will be used to describe the valid inequalities. Given a vertex set $S \subseteq V$, define $\bar{S} = \{i \in V | i \notin S\}$ and $\delta(S) = \delta^+(S) \cup \delta^-(S)$ where $\delta^+(S) = \{(i,j) \in A | i \in S, j \notin S\}$ and $\delta^-(S) = \{(i,j) \in A | i \notin S, j \in S\}$. For notational convenience, let $x_{ij} = \sum_{k \in K} x_{ij}^k$ denote the total flow on arc (i,j) and define $x(S) = \sum_{i,j \in S} x_{ij}$. Similarly, let $x(A') = \sum_{(i,j) \in A'} x_{ij}$ for any arc set $A' \subseteq A$.

4.1 Lower bounds on the QC completion time

To strengthen the formulation $F2$ we introduce lower bounding inequalities on the QC completion time. New variables are required:

$w_0^k, \forall k \in K,$ a lower bound on the idle time of the QC k at the starting time,

$\rho_k, \forall k \in K,$ a lower bound on the travel time for the QC k on its right side,

$\lambda_k, \forall k \in K,$ a lower bound on the travel time for the QC k on its left side,

$\upsilon_k, \forall k \in K,$ the maximum value between ρ_k and υ_k.

The defining constraints are then:

$$w_0^k \geq \sum_{j \in \Omega, \pi'(j) \neq \emptyset} \zeta(0,j) x_{0j}^k \qquad \forall k \in K, \tag{44}$$

$$\rho_k \geq t_{0i}^k y_{ik} \qquad \forall i \in \Omega, l_i > l_k^0, \forall k \in K, \tag{45}$$

$$\lambda_k \geq t_{0i}^k y_{ik} \qquad \forall i \in \Omega, l_i < l_k^0, \forall k \in K, \tag{46}$$

$$\upsilon_k \geq \rho_k \qquad \forall k \in K, \tag{47}$$

$$\upsilon_k \geq \lambda_k \qquad \forall k \in K, \tag{48}$$

$$C^k \geq w_0^k + r_k + 2\lambda_k + 2\rho_k - \upsilon_k + \sum_{i \in \Omega} p_i y_{ik} \qquad \forall k \in K, \tag{49}$$

$$\rho_k, \lambda_k, \upsilon_k \geq 0 \qquad \forall k \in K. \tag{50}$$

Constraints (44) define a straightforward lower bound w_0^k on the idle time of the QC k at the starting time when the first task performed by the QC k requires preceding tasks to be completed. Note that we are assuming that all the crane earliest available times r_k are equal. This assumption holds for our instance set but is not true in general. However, a slightly different constraint, not reported here, can work for the general case. Constraints (45) compute ρ_k as a lower bound on the travel time of the QC k with respect to the starting position l_k^0. Analogously, constraints (46) define a lower bound on the travel time λ_k of QC k. Constraints (47) and (48) define υ_k as the maximum value between ρ_k and υ_k. Using ρ_k, λ_k and υ_k we can compute a lower bound on the QC travel time. Since a crane moves along a line, this quantity cannot be less than $2\lambda_k + 2\rho_k - \upsilon_k$. Therefore constraints (49) define the QC completion time as bounded by the sum of the starting idle time, travel time and processing times of its assigned tasks.

Another way of defining a lower bound on the QC completion time is:

$$C^k \geq r_k + \sum_{j \in \Omega}(t_{0j}^k + \zeta(0,j))x_{0j}^k + \sum_{i \in \Omega}\sum_{j \in \Omega^T}(p_i + t_{ij})x_{ij}^k \qquad \forall k \in K. \tag{51}$$

In the following we consider inequalities (44) - (50) and (51) as part of formulation $F2$.

4.2 Subtour elimination constraints

In the case of the QCSP, the subtour elimination constraint $x(S) \leq |S| - 1$ for $S \subseteq \Omega$ can be lifted in many different ways by taking into account the precedence relationships. Balas et al. (1995) have proposed two families of inequalities for the asymmetric PC-TSP that also apply to the QCSP. The following inequality, called a successor inequality (or σ-inequality) is valid for the QCSP:

$$x(S) + \sum_{i \in \bar{S} \cap \sigma(S)}\sum_{j \in S} x_{ij} + \sum_{i \in \bar{S} \setminus \sigma(S)}\sum_{j \in S \cap \sigma(S)} x_{ij} \leq |S| - 1. \tag{52}$$

Example. Consider the set $S = \{i, j, l\} \subseteq \Omega$ for which a non-transitively derived precedence relationship between i and l exists, i.e. $l \in \sigma(i)$ and $\sigma(i) \cap \pi(l) = \emptyset$. Suppose also that task j does not have precedence relationships with the other tasks in S. We will use this set S to illustrate lifted subtour elimination constraints later in this section. We note that $\sigma(S) = \{l\} \cup \sigma(l) \cup \sigma(j)$. We depict in Figure 1 the successor inequality associated with S, where the lifted arcs are indicated with dotted lines.

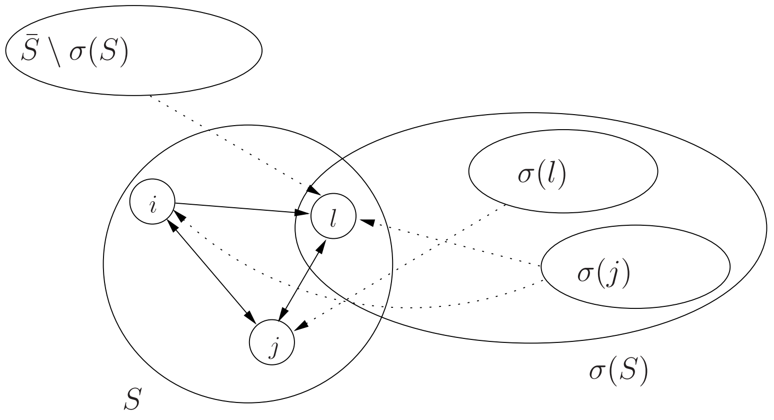

Figure 1: Successor inequality for $S = \{i, j, l\} \subseteq \Omega$

Similarly, for any set $S \subseteq \Omega$, the following predecessor inequality (or π-inequality) is valid for the QCSP:

$$x(S) + \sum_{i \in S} \sum_{j \in \bar{S} \cap \pi(S)} x_{ij} + \sum_{i \in S \cap \pi(S)} \sum_{j \in \bar{S} \setminus \pi(S)} x_{ij} \leq |S| - 1. \tag{53}$$

Example. The set $\pi(S)$ is equal to $\{i\} \cup \pi(i) \cup \pi(j)$. We illustrate in Figure 2 the corresponding predecessor inequality where the lifted arcs are shown by dotted lines.

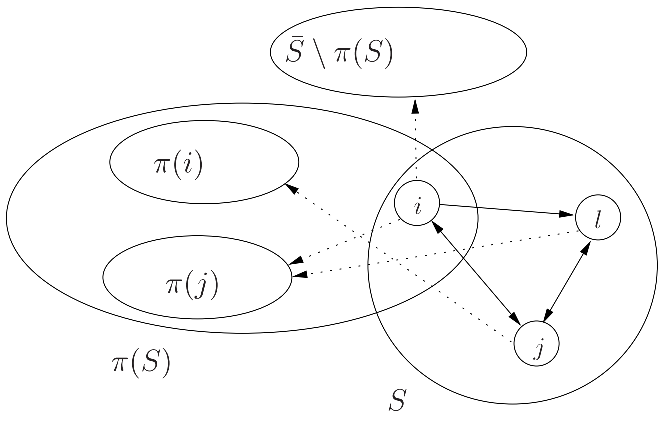

Figure 2: Predecessor inequality for $S = \{i, j, l\} \subseteq \Omega$

Because we have a directed formulation, we can also lift subtour elimination constraints by taking into account the orientation of the arcs. For an ordered set $S = \{i_1, i_2, \ldots, i_h\} \subseteq$

Ω with $h \geq 3$, Grötschel and Padberg (1985) proposed the following inequalities for the asymmetric TSP:

$$\sum_{j=1}^{h-1} x_{i_j,i_{j+1}} + x_{i_h,i_1} + 2\sum_{j=2}^{h-1} x_{i_j,i_1} + \sum_{j=3}^{h-1}\sum_{l=2}^{j-1} x_{i_j,i_l} \leq h-1 \tag{54}$$

$$\sum_{j=1}^{h-1} x_{i_j,i_{j+1}} + x_{i_h,i_1} + 2\sum_{j=3}^{h} x_{i_1,i_j} + \sum_{j=4}^{h}\sum_{l=3}^{j-1} x_{i_j,i_l} \leq h-1. \tag{55}$$

Considering different orderings of the nodes in S we can obtain different liftings. As in Cordeau (2005) these inequalities can be further lifted by taking precedence relationships into account. Two families of valid inequalities for the DARP that exploit this idea are introduced in Cordeau (2005) and are also valid for the QCSP. We report these inequalities in Propositions 1 and 3 along with their proofs because we extend them in Propositions 2 and 4. These new propositions take into account situations occurring with the QCSP structure, i.e. tasks can have multiple predecessors and successors, which is different from the DARP structure.

Proposition 1. Let $S = \{i_1, i_2, \ldots, i_h\} \subseteq \Omega$. The following inequality is valid for the QCSP:

$$\sum_{j=1}^{h-1} x_{i_j,i_{j+1}} + x_{i_h,i_1} + 2\sum_{j=2}^{h-1} x_{i_j,i_1} + \sum_{j=3}^{h-1}\sum_{l=2}^{j-1} x_{i_j,i_l} + \sum_{l\in\bar{S}\cap\sigma(S)} x_{l,i_1} \leq h-1. \tag{56}$$

Proof. Suppose that $\bar{S} \cap \sigma(S) \neq \emptyset$ and one arc of the form (l, i_1) with $l \in \bar{S} \cap \sigma(S)$ is part of the solution. Then no arc (i_j, i_1) with $2 \leq j \leq h$ can belong to the solution. As a result, if the left-hand side of (56) is larger than $h-1$, then there exists a subpath linking the h elements of S. But because S contains at least one task i_l which must precede l, i.e. $l \in \sigma(i_l)$, this subpath together with the arc (l, i_1) would violate the precedence constraint for i_l. $\square$

Example. One possible lifted directed subtour elimination constraint (obtained with $i_1 =$

$j, i_2 = i, i_3 = l$) is illustrated in Figure 3.

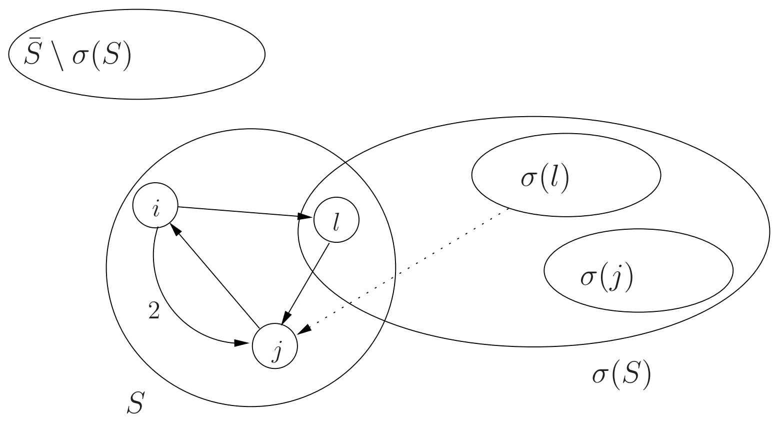

Figure 3: Lifted directed subtour elimination constraint of Proposition 1

Proposition 2. Let $S = \{i_1, i_2, \ldots, i_h\} \subseteq \Omega$ and $i_1 \in \sigma(S)$. The following inequality is valid for the QCSP:

$$\sum_{j=1}^{h-1} x_{i_j,i_{j+1}} + x_{i_h,i_1} + 2\sum_{j=2}^{h-1} x_{i_j,i_1} + \sum_{j=3}^{h-1}\sum_{l=2}^{j-1} x_{i_j,i_l} + \sum_{l \in \bar{S}} x_{l,i_1} \quad \leq \quad h-1. \tag{57}$$

Proof. Suppose that one arc (l, i_1) with $l \in \bar{S}$ is part of the solution. If $l \in \bar{S} \cap \sigma(S)$ Proposition 1 still holds. Hence assume that $l \in \bar{S} \setminus \sigma(S)$. Then no other arc entering i_1 can belong to the solution. But, if the left-hand side of (57) is larger than $h-1$, then there exists a subpath linking the h elements of S, starting from i_1. But because S contains at least one task i_d that must precede i_1, i.e. $i_1 \in \sigma(i_d)$, this subpath would violate the precedence constraint for i_d. □

Example. We illustrate in Figure 4 the case with $i_1 = l, i_2 = j, i_3 = i$.

Proposition 3. Let $S = \{i_1, i_2, \ldots, i_h\} \subseteq \Omega$. The following inequality is valid for the QCSP:

$$\sum_{j=1}^{h-1} x_{i_j,i_{j+1}} + x_{i_h,i_1} + 2\sum_{j=3}^{h} x_{i_1,i_j} + \sum_{j=4}^{h}\sum_{l=3}^{j-1} x_{i_j,i_l} + \sum_{l \in \bar{S} \cap \pi(S)} x_{i_1,l} \quad \leq \quad h-1. \tag{58}$$

Proof. The proof is similar to that of Proposition 1 by observing that if one arc (i_1, l) with

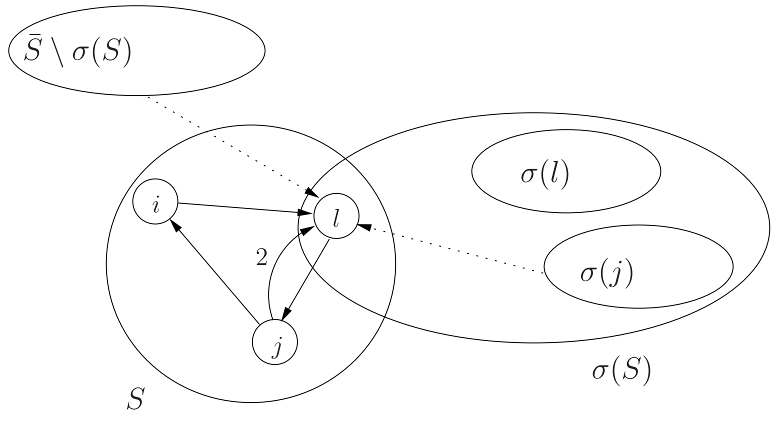

Figure 4: Lifted directed subtour elimination constraint of Proposition 2

$l \in \bar{S} \cap \pi(S)$ is part of the solution, then no arc (i_1, i_j) with $2 \leq j \leq h$ can belong to the solution. □

Example. In Figure 5 we report a lifted directed subtour elimination constraint obtained with $i_1 = j, i_2 = i, i_3 = l$.

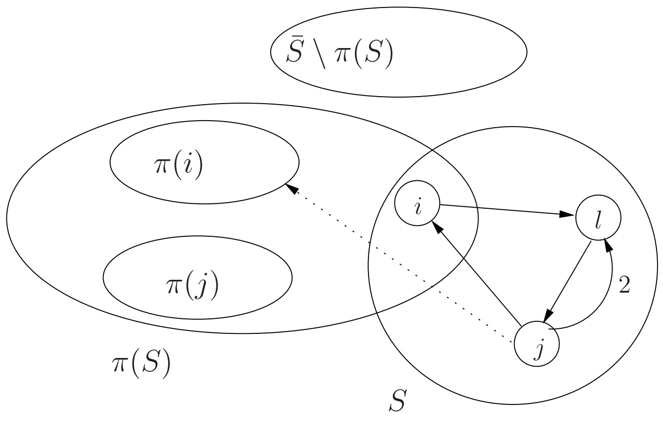

Figure 5: Lifted directed subtour elimination constraint of Proposition 3

Proposition 4. Let $S = \{i_1, i_2, \dots, i_h\} \subseteq \Omega$ and $i_1 \in \pi(S)$. The following inequality is valid for the QCSP:

$$\sum_{j=1}^{h-1} x_{i_j,i_{j+1}} + x_{i_h,i_1} + 2\sum_{j=3}^{h} x_{i_1,i_j} + \sum_{j=4}^{h}\sum_{l=3}^{j-1} x_{i_j,i_l} + \sum_{l\in\bar{S}} x_{i_1,l} \quad \leq \quad h-1. \tag{59}$$

Proof. The proof is similar to that of Proposition 2. □

Example. In Figure 6 we depict a lifted directed subtour elimination constraint obtained with $i_1 = i, i_2 = j, i_3 = l$.

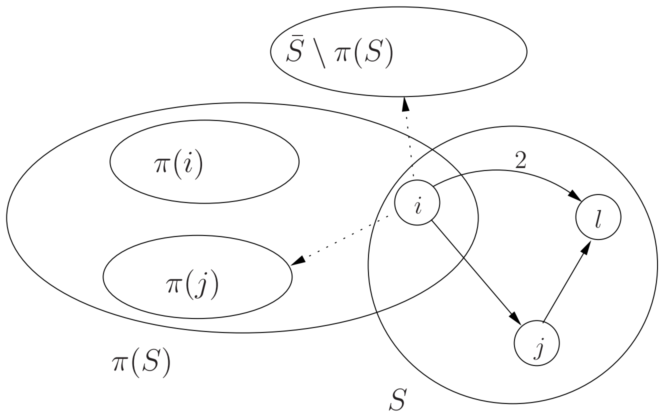

Figure 6: Lifted directed subtour elimination constraint of Proposition 4

5 Branch-and-Cut Algorithm

Branch-and-cut has been successfully applied to several routing problems (see, e.g., Ascheuer et al. 2001, Naddef and Rinaldi 2002, Laporte et al. 2003). This section describes our branch-and-cut implementation for the QCSP. It uses preprocessing techniques to reduce instance size and separation heuristics to identify violated inequalities.

After applying the preprocessing steps presented in Section 5.1 and generating an initial pool of inequalities, the algorithm first solves the LP relaxation of the problem. If the solution to the LP relaxation is integer, an optimal solution has been identified. Otherwise, an enumeration tree is constructed and violated valid inequalities are identified by means of the separation heuristics described in Section 5.4, and incorporated into subproblems. Because all inequalities described in Section 4 are valid for the original formulation, the inequalities added at any node of the tree are also valid for all other nodes. Hence, whenever the LP bound is evaluated at a given node of the tree, the linear program incorporates all cuts generated thus far.

To control the branch-and-cut process, branching is first performed on the variables y_i^k which reflect the assignment of tasks to cranes. At a given node of the search tree, if the y_i^k variables are all integer but there exists at least one fractional x_{ij}^k variable, the separation heuristics are executed in the hope of identifying violated valid inequalities. If at least one of the heuristics succeeds in finding one or more violated inequalities, the relaxation is solved with all identified cuts and the heuristics are executed again. The cut generation process at a node terminates when all heuristics fail to find any violated inequality. If the solution to the relaxation is still fractional after the generation of cuts, branching is performed on a fractional y_i^k variable, if there is any, or on a fractional x_{ij}^k variable, otherwise. The priorities for branching are reported in Section 5.2. In addition to the application of the separation heuristics at some nodes of the branch-and-bound tree, a pool of inequalities is checked exhaustively for violations at each node of the tree, including those where not all y variables take integer values. The inequalities that belong to this pool are described in Section 5.3.

5.1 Variable fixing

During the preprocessing phase we use the following variable fixing rules which result from the precedence relationships:

$$x_{ji}^k = 0 \qquad \forall (i,j) \in \Phi, \forall k \in K,$$

$$x_{ij}^k = 0 \qquad \forall (i,j) \in \Phi, \forall k \in K : \exists\, l \in \Omega, (i,l) \in \Phi, (l,j) \in \Phi.$$

5.2 Branching priorities

We set the branching priorities, expressed as $b()$, as follows:

$b(y_{ik}) = 100, \forall i \in \Omega, k \in K,$

$b(x_{0j}^k) = 10, \forall j \in \Omega^T, k \in K,$

$b(x_{ij}^k) = 1, \forall i \in \Omega, j \in \Omega^T, k \in K,$

where higher values correspond to higher priorities.

5.3 Initial pool of inequalities

The initial pool of inequalities comprises all constraints that are enumerated exhaustively and whose violations are checked individually at every node of the branch-and-bound tree. These inequalities are chosen to ensure that the size of the pool will remain reasonable so as to avoid slowing down the processing of a node. We have included in the initial pool of inequalities subtour elimination constraints (52) and (53) with $|S| = 2$.

5.4 Separation Heuristics

The identification of violated inequalities of the form $x(S) \leq |S| - 1$ can be achieved by solving a series of maximum flow problems between any task i and all other tasks $j \in V \setminus \{i\}$. However, in addition to being computationally expensive, this approach does not take the possible liftings into account. For these reasons, we resort to a simple tabu search heuristic inspired from the procedures proposed by Augerat et al. (1999) and Cordeau (2005).

When all y_i^k variables are integer but there is at least one fractional x_{ij}^k variable, two heuristics are executed sequentially. The first one checks for violations of (52), (56) and (57). Using the fact that $2x(S) + x(\delta(S)) = 2|S|$ in a feasible integer solution, violations of (52) can be identified by finding sets S such that

$$x(\delta(S)) - 2 \sum_{i \in \bar{S} \cap \sigma(S)} \sum_{j \in S} x_{ij} - 2 \sum_{i \in \bar{S} \setminus \sigma(S)} \sum_{j \in S \cap \sigma(S)} x_{ij} < 2. \tag{60}$$

We select the set of tasks $i \in \Omega$ assigned to a QC $k \in K$. Define this set as S_k. We search for violated inequalities only if $|S_k| \geq 3$, since we have already inserted in the initial pool of inequalities all subtour elimination constraints with $|S| = 2$. The heuristic starts with an empty set S. At each iteration, it either adds or removes a task from S so as to minimize the left-hand side of (60). Whenever a task is removed from S, its reinsertion is declared tabu for θ iterations. The heuristic runs for a number of iterations equal to $\tau \times |S_k|$, where τ is a user-defined parameter. At each iteration, the current set S is also checked for possible violations of inequalities (56) and (57). Given a task $i \in S$ we define $\chi_\sigma(i) = 2\sum_{j \in S \setminus \{i\}} x_{j,i} + \sum_{l \in \bar{S}} x_{l,i}$, if $i \in \sigma(S)$, or $\chi_\sigma(i) = 2\sum_{j \in S \setminus \{i\}} x_{j,i} + \sum_{l \in \bar{S} \cap \sigma(S)} x_{l,i}$, otherwise. The task with the largest value of χ_σ is labelled as i_1 and the other tasks are labelled at random. A similar heuristic is then used to identify violations of inequalities (53), (58) and (59). In the latter case, the task with the largest value of $\chi_\pi(i)$ is labelled as i_1, where $\chi_\pi(i) = 2\sum_{j \in S \setminus \{i\}} x_{i,j} + \sum_{l \in \bar{S}} x_{i,l}$, if $i \in \pi(S)$, or $\chi_\pi(i) = 2\sum_{j \in S \setminus \{i\}} x_{i,j} + \sum_{l \in \bar{S} \cap \pi(S)} x_{i,l}$, otherwise.

6 Computational Experiments

We first describe our test instances (Section 6.1), and then provide results obtained with formulations $F1$ and $F2$ in Section 6.3, and with the branch-and-cut algorithm in Section 6.4. We also present some implementation details and sensitivity analyses of our algorithms in Section 6.2.

6.1 Test instances

We have conducted tests on the 37 instances introduced by Kim and Park (2004). These instances are numbered from 13 to 49 and in Table 1 we report their size in terms of number of QCs and tasks. It turns out that the instances belonging to the sets $\{13, \ldots, 22\}$

Instance set	Number of QCs	Number of tasks
$\{13, \ldots, 22\}$	2	10
$\{23, \ldots, 32\}$	2	15
$\{33, \ldots, 42\}$	3	20
$\{43, \ldots, 49\}$	3	25

Table 1: Description of the instances

and $\{23, \ldots, 32\}$ are easily solved with formulation $F2$, while the remaining instance sets are more challenging. Therefore, we break up our computational experiments in two parts and, for convenience, we define the instances ranging from 13 to 32 as instance set $I1$ and the others as instance set $I2$.

A task is a collection of containers and an average of 30 containers per task is common in practice. Therefore an instance of 25 tasks corresponds to 750 moves. Using data from the Medcenter Container Terminal located in the port of Gioia Tauro we found that 75 % of the vessels requested less than 750 moves. However, the terminal traffic originating from vessels requiring more than 750 moves accounts for about half of the total traffic. Furthermore, these larger vessels have higher priorities.

6.2 Implementation details

Our algorithms were implemented in C++ by using ILOG Concert 1.3 and CPLEX 8.1. They were run on a 2.5 GHz Pentium IV computer with 512MB of memory. The CPLEX parameters affect all algorithms. We let only two parameters vary: the MIP emphasis and the heuristic frequency. The CPLEX MIP emphasis parameter, which in the following is denoted μ, controls the exploration of the branch-and-bound tree. The four options are:

- $\mu = 0$, default, balance optimality and feasibility;
- $\mu = 1$, emphasis on feasibility;
- $\mu = 2$, emphasis on optimality;
- $\mu = 3$, emphasis on improving the best bound.

The heuristic frequency parameter indicates how often the CPLEX heuristic is applied in the hope of identifying feasible integer solutions. The default value is zero, and the heuristic is activated at an interval chosen automatically by CPLEX. Setting the value to a positive integer f_h applies the heuristic at the node interval 0, f_h, $2f_h$, $3f_h$, etc.

A sensitivity analysis of our algorithms to the CPLEX parameters was carried out. The performance of the algorithm on formulations $F1$ and $F2$ is not very sensitive to μ on instance set I1, except for the fact that setting $\mu = 3$ produces rather poor results. Some more noticeable differences occur when formulation $F2$ is tested on the instance set $I2$. Here the $\mu = 1$ yields the best results. Therefore the μ parameter is set equal to one for formulation $F2$ and, for consistency, the same choice is applied to $F1$. The CPLEX heuristic frequency f_h is set to the default zero value for formulations $F1$ and $F2$ since other choices produce worse results. For these two algorithms setting $\mu = 1$ (with an emphasis on feasibility) already provides good quality solutions. The branch-and-cut algorithm, instead, uses different settings: $\mu = 2$ and $f_h = 1$. This parameter combination works better since the cutting phase improves the lower bounds but takes additional computation time. The MIP emphasis on optimality exploits the better lower bounds and the CPLEX heuristic finds good solutions by exploring fewer nodes of the branch-and-cut tree. The parameters used in the tabu search procedures are as follows:

- θ : tabu duration, equal to 5;
- τ : parameter that controls the number of tabu search iterations, equal to 4.

We set a two hours time limit for all algorithms.

6.3 Results with the formulations $F1$ and $F2$

We first compare the two formulations $F1$ and $F2$ on the instance set $I1$. Here formulation $F2$ is always capable of yielding an optimal solution within a short computation time, while formulation $F1$ is much slower. The results are reported in Table 2 and are consistent with those of Kim and Park (2004) who showed that formulation $F1$ could only solve within a reasonable time instances involving up to two QCs and six tasks, smaller than those considered in our experiments. We note that instance 13 is solved to optimality by formulation $F2$ with a solution value larger than the upper bound obtained with formulation $F1$. This apparently strange result is due to the slight variation in the solution space between the two formulations: formulation $F2$ incorporates constraints (33) and (34) that avoid QC collisions as indicated in Section 3.3.1. However, if we remove these constraints from $F2$ the optimal solution of the instance 13 is equal to that found by $F1$, but the solution contains a collision. We note that in Kim and Park (2004) the scheduling problem is solved exactly by a dedicated branch-and-bound scheme on a reduced solution space which avoids such cases. In fact the optimal solution on this reduced solution space found on instance 13 by Kim and Park has the same solution value, 453, as ours. However, comparing the optimal solutions found on instance set $I1$ by our algorithm and that of Kim and Park, we note that our approach (which does not assume solution space reduction) generates some benefits because, for example, on instances 20 and 22 we find better solutions with a 20 and 34% makespan reduction (Table 3).

We have run additional tests in order to asses the contribution of inequalities (44) - (50) and (51) to the performance of formulation $F2$. Denote by $F2_a$ formulation $F2$ without constraints (44) - (50) and (51), by $F2_b$ formulation $F2_a$ plus constraints (44) - (50), and by $F2_c$ formulation $F2_a$ plus constraints (51). The results of the corresponding three algorithms over the instance set $I1$ are reported in Table 4. We note that $F2_a$ improves over $F1$ (see also Table 2). However more significant enhancements are achieved adding constraints (44) - (50) or (51). Constraints (51) yield the most important contribution. However, by only combining (44) - (50) and (51) we achieve the best results since $F2$ dominates $F2_c$.

6.4 Results with the branch-and-cut algorithm

We have tested our branch-and-cut algorithm on instance set $I2$ where the formulation $F2$ does not always converge within the assigned two hours time limit. As reported in Table 5, the branch-and-cut algorithm proves the optimality of three more instances with respect to $F2$ and also improves the average computation time and solution quality. We have run some tests to evaluate the merit of two parts of our algorithm, the subtour elimination constraints with $|S| = 2$ added to the root node, and the tabu search heuristic which seeks violated subtour elimination constraints. Running the algorithm without one of these two components yields slightly worse results. In both cases the average gap on the instance set $I2$ becomes equal to 1.9% instead of 1.0% with the full branch-and-cut algorithm.

7 Conclusions and future research

We have formulated and solved the quay crane scheduling problem encountered in the daily operation of maritime container terminals. We have developed an improved formulation of the QCSP capable of solving small and medium size instances. A branch-and-cut algorithm, which exploits the precedence relationships of the problem, further improves these results.

Ideally, algorithms for the QCSP should handle larger instances than those solved in this study, which would require the use of heuristics. Our branch-and-cut algorithm can already handle the scheduling of the majority of the vessels arriving to a transhipment terminal container. It should also prove valuable to benchmark solution quality of heuristics.

Acknowledgment

This work was supported by the Canada Research Chair in Distribution Management, by the HEC Centre for International Business Studies, and by the CN Chair in Transport Economics and Intermodality. The authors thank K. H. Kim and Y. M. Park who kindly provided the instance set. Thanks are also due to the referees for their valuable comments.

References

Ascheuer, N.: 1996, *Hamiltonian Path Problems in the On-line Optimization of Flexible Manufacturing Systems*, PhD thesis, Konrad Zuse Zentrum für Informationstechnik Berlin.

Ascheuer, N., Fischetti, M. and Grötschel, M.: 2001, Solving the asymmetric travelling salesman problem with time windows by branch-and-cut, *Mathematical Programming* **90**, 475–506.

Ascheuer, N., Jünger, M. and Reinelt, G.: 2000, A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints, *Computational Optimization and Applications* **17**, 61–84.

Augerat, P., Belenguer, J., Benavent, E., Corberán, A. and Naddef, D.: 1999, Separating capacity inequalities in the CVRP using tabu search, *European Journal of Operational Research* **106**, 546–557.

Balas, E., Fischetti, M. and Pulleyblank, W.: 1995, The precedence-constrained asymmetric traveling salesman polytope, *Mathematical Programming* **68**, 241–265.

Christiansen, M., Fagerholt, K. and Ronen, D.: 2004, Ship routing and scheduling: Status and perspectives, *Transportation Science* **38**, 1–18.

Cordeau, J.-F.: 2005, A branch-and-cut algorithm for the dial-a-ride problem, *Operations Research*. Forthcoming.

Daganzo, C. F.: 1990, The productivity of multipurpose seaport terminals, *Transportation Science* **24**, 205–216.

Grötschel, M. and Padberg, M.: 1985, Polyhedral theory, *in* E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys (eds), *The Traveling Salesman Problem*, Wiley, New York, pp. 251–305.

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

Laporte, G., Riera Ledesma, J. and Salazar González, J. J.: 2003, A branch-and-cut algorithm for the undirected traveling purchaser problem, *Operations Research* **51**, 940–951.

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

Naddef, D. and Rinaldi, G.: 2002, Branch-and-cut algorithms for the capacitated VRP, *in* P. Toth and D. Vigo (eds), *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, pp. 53–84.

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

Steenken, D., Voss, S. and Stahlbock, R.: 2004, Container terminal operation and operations research - a classification and literature review, *OR Spectrum* **26**, 3–49.

UNCTAD: 2004, Review of maritime transport, *Technical report*, United Nations, New York and Geneva.

Vis, I. F. A. and Koster, R. D.: 2003, Transshipment of containers at a container terminal: An overview, *European Journal of Operational Research* **147**, 1–16.

	$F1$			$F2$		
Instance code	Solution	Time (min.)	Gap (%)	Solution	Time (min.)	Gap (%)
13	450	120.00	41.33	453	7.75	0.00
14	555	120.00	56.76	546	0.01	0.00
15	513	120.00	58.48	513	0.02	0.00
16	321	120.00	48.60	312	0.09	0.00
17	459	120.00	47.06	453	0.04	0.00
18	399	120.00	55.64	375	0.01	0.00
19	552	120.00	50.54	543	1.96	0.00
20	402	120.00	53.73	399	0.04	0.00
21	471	120.00	63.69	465	0.01	0.00
22	537	120.00	43.58	537	0.18	0.00
23	708	120.00	51.69	576	0.06	0.00
24	900	120.00	63.00	666	0.96	0.00
25	822	120.00	64.60	738	0.59	0.00
26	792	120.00	72.35	639	0.14	0.00
27	708	120.00	62.29	657	0.05	0.00
28	693	120.00	70.13	531	0.21	0.00
29	1026	120.00	68.71	807	0.15	0.00
30	999	120.00	65.77	891	0.22	0.00
31	711	120.00	32.91	570	85.56	0.00
32	684	120.00	74.12	591	1.12	0.00
Average	635	120.00	57.25	563	4.96	0.00

Table 2: Comparison between $F1$ and $F2$ formulations. The % gap is computed with respect to the value of the linear relaxation as $100 \times$ (upper bound $-$ lower bound)/upper bound.

Instance code	Solution value KP	$F2$	Gap (%)
13	453	453	0.0
14	546	546	0.0
15	513	513	0.0
16	321	312	2.9
17	456	453	0.7
18	375	375	0.0
19	552	543	1.7
20	480	399	20.3
21	465	465	0.0
22	720	537	34.1
23	576	576	0.0
24	669	666	0.5
25	738	738	0.0
26	639	639	0.0
27	657	657	0.0
28	537	531	1.1
29	807	807	0.0
30	891	891	0.0
31	570	570	0.0
32	591	591	0.0

Table 3: Comparison between solutions on instance set $I1$ reported in Kim and Park (2004) and with the formulation $F2$. The % gap is computed as $100 \times (KP - F2)/F2$

	$F2_a$			$F2_b$			$F2_c$		
Instance code	Solution	Time (min.)	Gap (%)	Solution	Time (min.)	Gap (%)	Solution	Time (min.)	Gap (%)
13	453	120.00	41.72	453	24.96	0.00	453	7.33	0.00
14	558	120.00	56.99	546	1.89	0.00	546	0.03	0.00
15	513	120.00	58.48	513	0.18	0.00	513	0.02	0.00
16	312	120.00	47.12	312	4.64	0.00	312	0.15	0.00
17	456	120.00	46.71	453	1.41	0.00	453	0.04	0.00
18	375	120.00	52.80	375	0.50	0.00	375	0.04	0.00
19	543	120.00	49.72	543	20.34	0.00	543	1.19	0.00
20	399	120.00	53.38	399	1.18	0.00	399	0.39	0.00
21	468	120.00	63.46	465	0.36	0.00	465	0.01	0.00
22	537	120.00	43.58	537	1.99	0.00	537	0.20	0.00
23	618	120.00	44.66	585	120.00	3.08	576	2.11	0.00
24	732	120.00	54.51	675	120.00	3.11	666	26.11	0.00
25	819	120.00	64.47	741	120.00	1.62	738	6.31	0.00
26	759	120.00	71.15	657	120.00	4.11	639	3.25	0.00
27	699	120.00	61.80	678	120.00	4.87	657	0.14	0.00
28	588	120.00	64.80	537	120.00	3.35	531	0.78	0.00
29	870	120.00	63.10	810	120.00	2.59	807	2.08	0.00
30	948	120.00	63.92	897	120.00	0.67	891	3.02	0.00
31	621	120.00	23.19	585	120.00	6.67	570	120.00	0.53
32	654	120.00	72.94	600	120.00	3.00	591	0.26	0.00
Average	596	120.00	54.92	568	62.87	1.65	563	8.67	0.03

Table 4: Comparison between three variants of formulation $F2$ on instance set $I1$. The % gap is computed with respect to the value of the linear relaxation as 100 $\times$ (upper bound $-$ lower bound)/upper bound.

	$F2$			B&C		
Instance code	Solution	Time (min.)	Gap (%)	Solution	Time (min.)	Gap (%)
33	603	32.8	0.0	603	10.6	0.0
34	717	31.1	0.0	717	14.9	0.0
35	684	53.0	0.0	684	42.0	0.0
36	678	120.0	0.6	678	86.1	0.0
37	510	112.7	0.0	510	21.2	0.0
38	618	120.0	2.0	618	120.0	0.7
39	546	120.0	8.2	513	120.0	0.9
40	564	120.0	1.2	564	67.1	0.0
41	588	120.0	1.2	588	120.0	0.5
42	570	120.0	2.1	570	120.0	1.7
43	939	120.0	8.9	897	120.0	4.2
44	858	120.0	5.7	822	120.0	0.2
45	846	120.0	3.3	840	120.0	1.8
46	870	120.0	21.9	690	90.4	0.0
47	792	56.0	0.0	792	27.0	0.0
48	762	120.0	18.3	645	120.0	2.5
49	1101	120.0	20.4	927	120.0	5.1
Average	720	101.5	5.5	686	84.7	1.0

Table 5: Results of the branch-and-cut algorithm on instance set $I2$ compared to formulation $F2$. The % gap is computed with respect to the value of the linear relaxation as 100 $\times$ (upper bound $-$ lower bound)/upper bound.