Chapter 7
Quay Crane Scheduling

This chapter deals with the QCSP on the basis of container groups. It is studied as an isolated problem here and functionally integrated into the BACAP in the next chapter. Crane scheduling for container groups has been introduced by Kim and Park (2004). As noted by Moccia et al. (2006), the original QCSP model provided by Kim and Park shows an inaccuracy regarding the detection of crane interference. Unfortunately, even reworked problem formulations still tolerate certain cases of crane interference. A corrected problem formulation and a heuristic solution method have been provided by Bierwirth and Meisel (2009). The model and the heuristic are presented in Sects. 7.1 and 7.2, respectively. In Sect. 7.3 the QCSP is extended by incorporation of time windows for the cranes. Necessary modifications of the mathematical formulation and the solution method are described. Computational tests follow in Sect. 7.4. The study on the QCSP is concluded in Sect. 7.5.

7.1 Modeling the QCSP

7.1.1 Problem Description and Assumptions

In the QCSP for container groups a set of tasks $\Omega = \{1,2,\ldots,n\}$ and a set of QCs $Q = \{1,2,\ldots,q\}$ are given. Each task $i \in \Omega$ represents a loading or unloading operation of a certain container group. The tasks have individual processing times p_i and bay positions l_i. Additionally, dummy tasks 0 and $T = n+1$ with processing times $p_0 = p_T = 0$ are given to indicate the begin and the end of the service of the vessel. Further task sets are defined by $\Omega^0 = \Omega \cup \{0\}$, $\Omega^T = \Omega \cup \{T\}$, and $\overline{\Omega} = \Omega \cup \{0,T\}$. Precedence relations may exist between pairs of tasks that are located within the same bay. Let Φ denote the set of precedence constrained task pairs. Furthermore, let $\Psi \supseteq \Phi$ denote the set of all task pairs for which it is known in advance that

F. Meisel, *Seaside Operations Planning in Container Terminals*,
Contributions to Management Science,
DOI: 10.1007/978-3-7908-2191-8_7,

they cannot be processed simultaneously. For each crane $k \in Q$ a ready time r^k and an initial bay position l_0^k is given. Without loss of generality, it is assumed that the cranes are indexed sequentially according to their initial positioning alongside the vessel. All QCs can move between two adjacent bays in an identical travel time $\hat{t} > 0$. It is supposed that no two QCs can operate at the same bay at the same time. Moreover, cranes are not allowed to cross each other and have to keep a safety margin δ, measured in units of bays. The problem is to find completion times c_i for all tasks $i \in \overline{\Omega}$ on the cranes with respect to the constraints, such that the completion time c_T of the final task T (i.e., the makespan) is minimized.

Assumptions of the QCSP with container groups are:

1. Container groups are predefined from given stowage plans.
2. Processing of tasks is non-preemptive.
3. All QCs show an identical, deterministic transshipment productivity. For this reason fixed processing times of tasks are given. No consideration of individual container moves or crane cycle times is necessary.
4. The order of processing the tasks of a bay is completely determined by precedence constraints.
5. Crane operations cannot lead to an instable load configuration of the vessel, i.e., stability issues are not considered in the QCSP.
6. There is sufficient space beside the vessel to place idle QCs outside of the vessel area.

The described problem corresponds to a minimum makespan scheduling problem with parallel identical machines and precedence constraints. This problem is known to be $\mathcal{NP}$-hard in the strong sense, provided that more than two machines, non-preemption or non-uniform processing times are given, see Pinedo (2002).

7.1.2 Conventional Formulation of Interference Constraints

In the BACAP study, the productivity loss caused by crane interference has been modeled using an interference exponent α. Within the QCSP, interference effects are considered in more detail in order to generate feasible QC schedules.

In correspondence with models in machine scheduling, it is supposed that no two QCs can operate at the same bay at the same time. Moreover, since QCs are rail mounted, two types of interference constraints have to be respected:

- Non-crossing constraint: QCs cannot cross each other.
- Safety constraint: Adjacent QCs have to keep a safety margin at all times.

The *safety margin* δ signifies a certain number of in-between bays between adjacent QCs. If, for example, $\delta = 1$ this means that QCs can simultaneously operate at the vessel if they are separated by at least one bay. Safety margin must be respected not only while QCs are working but also during the movement operations.

Interference constraints were first included in the QCSP model of Kim and Park (2004). As noted by Moccia et al. (2006) this model does not detect interference in every case. Therefore, Moccia et al. (2006) have revised the model of Kim and Park by incorporating travel times for QCs that subsequently process tasks in the same bay. A compact mathematical formulation of the revised model can be found in Sammarra et al. (2007). Unfortunately, also the modification proposed in the revised model may yield solutions where QCs cross or violate the safety margin. To demonstrate the incorrectness of this model those constraints that are responsible for detecting crane interference, labeled (9)–(14) in the paper of Sammarra et al. (2007), are briefly revisited:

$$c_i + p_j - c_j \leq M(1 - z_{ij}) \qquad \forall i, j \in \Omega, \tag{7.1}$$

$$c_i + p_j - c_j + \sum_{k \in Q} \sum_{\substack{u \in \Omega^0, \\ l_u \neq l_i}} \hat{t} x_{uj}^k \leq M(1 - z_{ij}) \qquad \forall i, j \in \Omega, l_i = l_j, \tag{7.2}$$

$$c_j - p_j - c_i \leq M z_{ij} \qquad \forall i, j \in \Omega, \tag{7.3}$$

$$c_j - p_j - c_i - \sum_{k \in Q} \sum_{\substack{u \in \Omega^0, \\ l_u \neq l_i}} \hat{t} x_{uj}^k \leq M z_{ij} \qquad \forall i, j \in \Omega, l_i = l_j, \tag{7.4}$$

$$z_{ij} + z_{ji} = 1 \qquad \forall (i, j) \in \Psi, \tag{7.5}$$

$$\sum_{v=1}^{k} \sum_{u \in \Omega^0} x_{uj}^v - \sum_{v=1}^{k} \sum_{u \in \Omega^T} x_{ui}^v \leq M(z_{ij} + z_{ji}) \qquad \forall i, j \in \Omega, l_i < l_j, \forall k \in Q. \tag{7.6}$$

In this formulation x_{ij}^k and z_{ij} denote binary decision variables. x_{ij}^k is set to 1 if tasks i and j are processed consecutively by crane k, and z_{ij} is set to 1 if task j starts after the completion of task i. The variables z_{ij} are defined in Constraints (7.1) and (7.3). Constraints (7.2) and (7.4) ensure that the travel time $\hat{t}$ is kept between the completion of a task and the start of the next task in the same bay if both tasks are processed by different QCs. In order to express a safety margin of one bay between adjacent QCs, Sammarra et al. (2007) include those pairs of tasks in set Ψ that belong to adjacent bays. Constraints (7.5) ensure that these tasks are not processed simultaneously. Finally, a simultaneous processing of tasks that inevitably requires a crossing of the assigned cranes is forbidden by Constraints (7.6).

To demonstrate the incorrectness of the above interference constraints two small example problems are considered which are solved infeasible. The first problem consists of four tasks with processing times 10, 20, 40, and 30, positioned in bays 1, 3, 5, and 7, respectively, and two cranes. The safety margin is set to one bay and the travel time to $\hat{t} = 1$ time unit per bay. Figure 7.1a shows an optimal schedule derived from the above model. In this solution, tasks 1 and 3 are assigned to QC 1, and tasks 2 and 4 are assigned to QC 2. Obviously, the solution is infeasible because QC 1 crosses QC 2 in order to process task 3. Since no two tasks are positioned within the same bay or adjacent bays, $\Psi = \emptyset$ in this problem. Therefore, Constraints (7.2), (7.4), and (7.5) do not appear in the model instance. The start

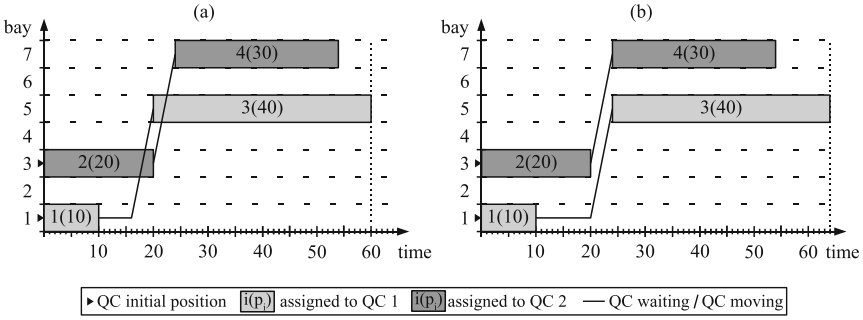

Fig. 7.1 Violation of the non-crossing requirement (**a**) and a feasible schedule (**b**)

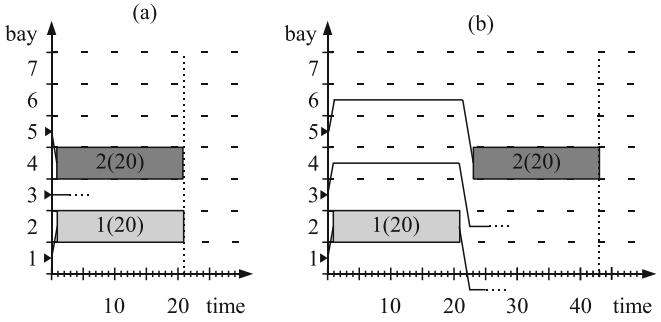

Fig. 7.2 Violation of the safety margin (**a**) and a feasible schedule (**b**)

time of task 4 is derived from the completion time of task 2 plus the time needed by QC 2 to travel from bay 3 to bay 7. Furthermore, the start time of task 3 is delayed by Constraint (7.1) because tasks 2 and 3 are not allowed to be processed simultaneously, as effected by setting $z_{23} = 1$ in Constraint (7.6). However, the inserted delay is insufficient to avoid a crossing of the cranes. The corrected feasible solution is shown in Fig. 7.1b. This solution is obtained by introducing a temporal distance of four time units between the completion of task 2 and the start of task 3. Through this correction the makespan increases from 60 to 64 time units.

The model formulation encounters a further weakness regarding QCs that stay idle during the service. The second problem instance consists of two tasks with identical processing time, positioned in bays 2 and 4. Three QCs, initially positioned at bays 1, 3, and 5, are available for the service. The safety margin and the travel time of cranes are as above. Figure 7.2a shows one optimal schedule where the QC positioned initially at bay 1 processes task 1 and the QC positioned at bay 5 processes task 2. Since there are no restrictions on idle cranes in the above model, the shown solution is not forbidden. However, it is infeasible because the idle QC is within the safety area of the active QCs. Without changing the task-to-QC assignment, a corrected feasible solution requires processing of the tasks consecutively. Moreover, as shown in Fig. 7.2b, the starting time of task 2 needs a further delay of two time units to enable a safe movement of the cranes. Now the safety margin is

always kept during the entire operation. As in the first problem, the inclusion of a suitable temporal distance between tasks resolves the crane conflict. Unfortunately, the repair does not preserve optimality of the solution.

7.1.3 Corrected Formulation of Interference Constraints

The above analysis discloses a serious weakness in the existing QCSP models. Temporal distances between tasks are only included if these tasks are positioned within the same bay. The key to a correct model formulation is the determination of a suitable temporal distance between any two tasks involved in a problem. For this purpose, the temporal distance is computed as a function of the bay positions of tasks, the safety margin, the QC travel time, and, last but not least, the realized task-to-QC assignment.

Let Δ_{ij}^{vw} denote the minimum time span to elapse between the processing of two tasks i and j, if processed by QCs v and w respectively. Due to the sequential indexing of cranes, one can say that v operates below (above) w if $v < w$ ($v > w$). Generally, a crossing must be avoided if the lower QC processes a task which is located at a bay above a task processed by the upper QC. Furthermore, compliance of the safety margin must be guaranteed between any two cranes v and w. Let δ_{vw} be the smallest allowed difference between the bay positions of cranes v and w. It is calculated as

$$\delta_{vw} = (\delta + 1)|v - w|. \tag{7.7}$$

For all combinations of tasks $i, j \in \Omega$ and QCs $v, w \in Q$ the minimum temporal distance is now defined as

$$\Delta_{ij}^{vw} = \begin{cases} (l_i - l_j + \delta_{vw})\hat{t}, & \text{if } v < w \text{ and } i \neq j \text{ and } l_i > l_j - \delta_{vw}, \\ (l_j - l_i + \delta_{vw})\hat{t}, & \text{if } v > w \text{ and } i \neq j \text{ and } l_i < l_j + \delta_{vw}, \\ 0, & \text{otherwise.} \end{cases} \tag{7.8}$$

The first case of (7.8), in which v operates below w, is illustrated in Fig. 7.3. Here, tasks i and j are processed by adjacent QCs v and $w = v + 1$. In the example, the safety margin is set to $\delta = 2$ and the QC travel time is set to $\hat{t} = 1$. Assuming that task j is completed at time c_j, QC v must not be positioned above $l_j - \delta_{vw}$ at this point in time because it operates below w. Since v processes its next task at bay l_i it has to traverse at least $l_i - (l_j - \delta_{vw})$ bays. The resulting minimum travel time of v is $\Delta_{ij}^{vw} = (l_i - l_j + \delta_{vw})\hat{t} = 6$ with $l_i - l_j = 3$. Note, that Δ_{ij}^{vw} yields the same value if task i is processed prior to j under ceteris paribus conditions. The reverse positioning of QCs is treated in the second case of (7.8). Without loss of generality, every instance of this case can be transformed into an identical instance of the former case by exchanging the roles of tasks i and j. In all other cases, cranes cannot come into conflict as indicated by setting the temporal distance to a value of 0.

The example in Fig. 7.3 considers interference of adjacent QCs. However, if v and w are not adjacent, (7.8) calculates a sufficiently large temporal distance between the

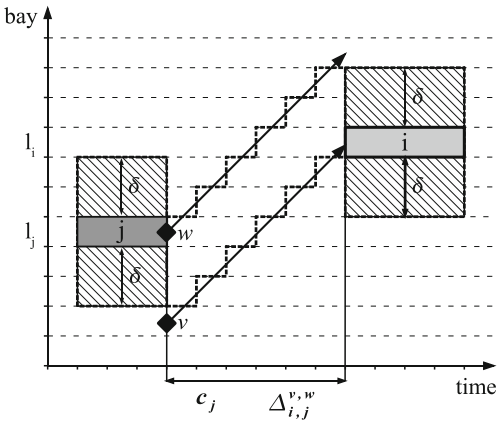

Fig. 7.3 Necessary time span between the execution of tasks by adjacent QCs

processing of any two tasks permitting a safe movement of the in-between cranes. This also applies if in-between cranes are idle, see Fig. 7.2b.

Let Θ denote the set of all combinations of tasks and QCs that potentially lead to crane interference. It can be defined as

$$\Theta = \{(i,j,v,w) \in \Omega^2 \times Q^2 \mid (i<j) \wedge (\Delta_{ij}^{vw} > 0)\}. \tag{7.9}$$

Due to the symmetry of the temporal distances Δ_{ij}^{vw} the consideration can be restricted to pairs of tasks with $i<j$. Actually, a certain combination in Θ will cause interference only if its task-to-QC assignment is selected in the QC schedule. Since this is unknown in advance, every element of Θ must be treated by a constraint in the QCSP model.

A correct formulation of the model of Sammarra et al. (2007) results, if Constraints (7.2), (7.4), and (7.6) are replaced by the following constraints:

$$\sum_{u\in\Omega^0} x_{ui}^v + \sum_{u\in\Omega^0} x_{uj}^w \le 1 + z_{ij} + z_{ji} \qquad \forall (i,j,v,w) \in \Theta, \tag{7.10}$$

$$c_i + \Delta_{ij}^{vw} + p_j - c_j \le M(3 - z_{ij} - \sum_{u\in\Omega^0} x_{ui}^v - \sum_{u\in\Omega^0} x_{uj}^w) \qquad \forall (i,j,v,w) \in \Theta, \tag{7.11}$$

$$c_j + \Delta_{ij}^{vw} + p_i - c_i \le M(3 - z_{ji} - \sum_{u\in\Omega^0} x_{ui}^v - \sum_{u\in\Omega^0} x_{uj}^w) \qquad \forall (i,j,v,w) \in \Theta. \tag{7.12}$$

In Constraints (7.10) those assignments of tasks to QCs are identified that are realized in the schedule. Here, $\sum_{u\in\Omega^0} x_{ui}^v = 1$ if and only if task i is processed by QC v and $\sum_{u\in\Omega^0} x_{uj}^w = 1$ if and only if task j is processed by QC w. If both assignments take place, the left side reveals a value of two and the tasks are not allowed to be processed simultaneously, i.e., either $z_{ij} = 1$ or $z_{ji} = 1$. In the case of $z_{ij} = 1$ Constraints (7.11) insert the minimum temporal distance calculated by (7.8) between the completion time of task i and the starting time of task j. The corresponding case of $z_{ji} = 1$ is handled in Constraints (7.12).

7.1.4 Optimization Model

In the original model of Kim and Park (2004) the minimization of the weighted sum of makespan and QC finishing times is pursued. Due to the predominant importance of short vessel handling times in current CT markets, most authors merely consider the minimization of makespan, as has been done in the computational studies of Kim and Park (2004), Moccia et al. (2006), and Sammarra et al. (2007) too. The following formulation takes up this lead and ignores QC finishing times in the objective function.

To model the movement of cranes, the travel time of QC k to traverse from its initial position l_0^k to l_j ($j \in \Omega$) is defined as $t_{0j}^k = \hat{t}|l_0^k - l_j|$. The travel time between bay positions l_i and l_j ($i, j \in \Omega$) is defined as $t_{ij} = \hat{t}|l_i - l_j|$. If i or j or both belong to $\{0, T\}$ the travel time is set to $t_{ij} = 0$. Since the minimization of QC finishing times is ignored here, final repositioning movements of QCs after completion of the vessel's service are not considered in this formulation.

The QCSP is formulated as follows:

$$\text{minimize } c_T \tag{7.13}$$

subject to

$$\sum_{j \in \Omega^T} x_{0j}^k = 1 \qquad \forall k \in Q, \tag{7.14}$$

$$\sum_{j \in \Omega^0} x_{jT}^k = 1 \qquad \forall k \in Q, \tag{7.15}$$

$$\sum_{k \in Q} \sum_{j \in \Omega^T} x_{ij}^k = 1 \qquad \forall i \in \Omega, \tag{7.16}$$

$$\sum_{j \in \Omega^0} x_{ji}^k - \sum_{j \in \Omega^T} x_{ij}^k = 0 \qquad \forall i \in \Omega, \forall k \in Q, \tag{7.17}$$

$$c_i + t_{ij} + p_j - c_j \leq M(1 - x_{ij}^k) \qquad \forall i, j \in \overline{\Omega}, \forall k \in Q, \tag{7.18}$$

$$c_i + p_j - c_j \leq 0 \qquad \forall (i, j) \in \Phi, \tag{7.19}$$

$$c_i + p_j - c_j \leq M(1 - z_{ij}) \qquad \forall i, j \in \Omega, \tag{7.20}$$

$$c_j - p_j - c_i \leq M z_{ij} \qquad \forall i, j \in \Omega, \tag{7.21}$$

$$z_{ij} + z_{ji} = 1 \qquad \forall (i, j) \in \Psi, \tag{7.22}$$

$$\sum_{u \in \Omega^0} x_{ui}^v + \sum_{u \in \Omega^0} x_{uj}^w \leq 1 + z_{ij} + z_{ji} \qquad \forall (i, j, v, w) \in \Theta, \tag{7.23}$$

$$c_i + \Delta_{ij}^{vw} + p_j - c_j \leq M(3 - z_{ij} - \sum_{u \in \Omega^0} x_{ui}^v - \sum_{u \in \Omega^0} x_{uj}^w) \qquad \forall (i, j, v, w) \in \Theta, \tag{7.24}$$

$$c_j + \Delta_{ij}^{vw} + p_i - c_i \leq M(3 - z_{ji} - \sum_{u \in \Omega^0} x_{ui}^v - \sum_{u \in \Omega^0} x_{uj}^w) \qquad \forall (i, j, v, w) \in \Theta, \tag{7.25}$$

$$r^k + t^k_{0j} + p_j - c_j \leq M(1 - x^k_{0j}) \qquad \forall j \in \Omega, \forall k \in Q, \quad (7.26)$$

$$c_i \geq 0 \qquad \forall i \in \overline{\Omega}, \quad (7.27)$$

$$x^k_{ij} \in \{0,1\} \qquad \forall i,j \in \overline{\Omega}, \forall k \in Q, \quad (7.28)$$

$$z_{ij} \in \{0,1\} \qquad \forall i,j \in \Omega. \quad (7.29)$$

The pursued objective given in (7.13) is to minimize the handling time of the vessel referred to as the makespan. The makespan is defined by the completion time of dummy task T because every crane is enforced to visit this task after processing its assigned non-dummy tasks. Constraints (7.14) and (7.15) ensure that every QC starts with the initial dummy task 0 and ends up with the final dummy task T. If both fall together, i.e., $x^k_{0T} = 1$, QC k remains idle during the entire service. Constraints (7.16) ensure that each non-dummy task is processed exactly once. Constraints (7.17) ensure that every non-dummy task has a preceding task and a succeeding task. The completion times of the tasks are computed in Constraints (7.18) where M is again a sufficiently large positive number. Note that for $j = T$ the makespan is computed by this constraint. The precedence relations are included in Constraints (7.19) with respect to the task completion times. Constraints (7.20) and (7.21) set the variables z_{ij}. On this basis the non-simultaneity condition of tasks is represented in Constraints (7.22). Constraints (7.23)–(7.25) are the new interference constraints formulated in the previous section. The ready times of QCs are handled in (7.26) and the feasible domains of the decision variables are defined in (7.27)–(7.29).

The number of variables used in this formulation grows in $O(n^2q)$, which is the same as in Sammarra et al. (2007). Due to the newly formulated interference handling, the number of constraints grows in $O(n^2q^2)$ instead of $O(n^2q)$.

The QCSP formulation is classified using the scheme of Sect. 4.2.1. Tasks are defined by container groups (*Group*) where precedence relations exist among pairs of tasks (*prec*). The model considers ready times, initial positions, and movement time of cranes, classified by the crane attribute values *ready*, *pos*, and *move*. The non-crossing requirement and safety margins are respected (*cross*, *save*). The makespan, i.e., the maximum completion time (*compl*) among tasks, is minimized. Hence, the model is classified by $Group, prec \mid ready, pos, move \mid cross, save$ $max(compl)$.

Example 7.1: QCSP instance: definition and optimal solution

Table 7.1 shows the data of a small QCSP instance, which is used in the following to illustrate the proposed solution procedure. The problem contains nine container groups placed in a vessel with eleven bays. Two QCs are assigned to this vessel. The minimum temporal distances for combinations of task pairs and QC pairs are preprocessed according to (7.8). E.g., for $(i,j,v,w) = (7,8,2,1)$ one obtains $\delta_{21} = 2$ and $\Delta^{21}_{78} = (9-7+2) = 4$. The complete matrix Δ^{vw}_{ij} is shown in Table 7.2. From this it can be seen that Δ is symmetric in i,j and v,w, i.e., $\Delta^{vw}_{ij} = \Delta^{wv}_{ji}$. Set Θ is composed

Table 7.1 Example QCSP instance

Task index i	1	2	3	4	5	6	7	8	9
Processing time p_i	22	46	8	70	10	38	40	16	12
Bay position l_i	1	1	2	3	5	5	7	9	11
Precedence-constrained tasks	$\Phi = \{(1,2),(5,6)\}$								
Non-simultaneous tasks	$\Psi = \{(1,2),(1,3),(2,3),(3,4),(5,6)\}$								
QC 1	$l_0^1 = 1, \quad r^1 = 0$								
QC 2	$l_0^2 = 4, \quad r^2 = 0$								
QC travel speed	$\hat{t} = 1$								
Safety margin	$\delta = 1$								

Table 7.2 Obtained temporal distances Δ_{ij}^{vw} for the QCSP instance

	i	1		2		3		4		5		6		7		8		9	
j	v \ w	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
1	1	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	2	0	3	0	4	0	6	0	6	0	8	0	10	0	12	0
2	1	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	2	2	0	0	0	3	0	4	0	6	0	6	0	8	0	10	0	12	0
3	1	0	3	0	3	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	2	1	0	1	0	0	0	3	0	5	0	5	0	7	0	9	0	11	0
4	1	0	4	0	4	0	3	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	1	0	0	0	4	0	4	0	6	0	8	0	10	0
5	1	0	6	0	6	0	5	0	4	0	0	0	2	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	2	0	4	0	6	0	8	0
6	1	0	6	0	6	0	5	0	4	0	2	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	2	0	0	0	4	0	6	0	8	0
7	1	0	8	0	8	0	7	0	6	0	4	0	4	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	6	0
8	1	0	10	0	10	0	9	0	8	0	6	0	6	0	4	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0
9	1	0	12	0	12	0	11	0	10	0	8	0	8	0	6	0	4	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

according to Definition (7.9). In the example it consists of 41 elements indicated by the positive entries below the main diagonal of matrix Δ.

The optimal solution to the problem is given by the task sequences 0-1-2-3-6-8-10 for QC 1 and 0-4-5-7-9-10 for QC 2. The associated task completion times are $c_i = (0, 22, 68, 80, 71, 83, 123, 125, 145, 141, 145)$. The corresponding schedule is depicted in Fig. 7.4. It respects the required temporal distances between task pairs $(3,4)$, $(5,6)$, and $(7,8)$ in order to avoid violations of constraints. Note that QC 1 becomes idle three times because it has to wait until QC 2 has finished a task and moves away.

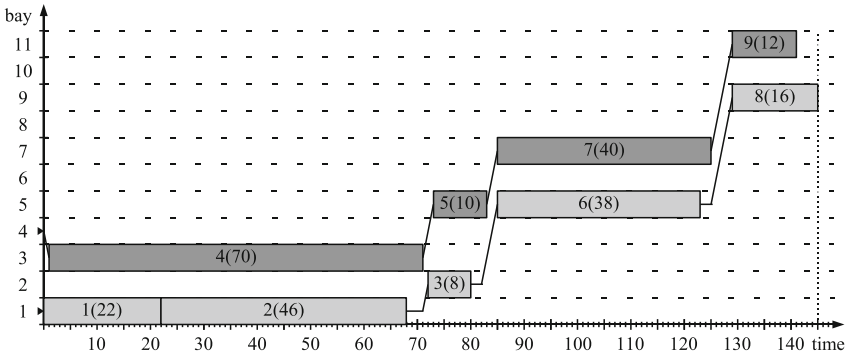

Fig. 7.4 Optimal QC schedule with makespan $c_T = 145$

7.2 Unidirectional Scheduling Heuristic

7.2.1 Idea and Outline

Following Bierwirth and Meisel (2009) a QC schedule is called a *unidirectional schedule* if the QCs do not change in moving direction after the initial repositioning and have identical directions of movement either from upper to lower bays or vice versa. The schedule depicted in Fig. 7.4 is a unidirectional schedule. As shown by Lim et al. (2007) for the QCSP with tasks defined by complete bays, there is at least one optimal schedule among the unidirectional ones. For this reason, anchoring unidirectionality in mathematical models of this QCSP variant, as has been done by Liu et al. (2006) and Lim et al. (2007), does not exclude the optimal schedule from the solution space.

In contrast, for the QCSP with container groups, one can easily construct instances with no unidirectional schedule existing among the optimal solutions. One such example is shown in Fig. 7.5 where (a) shows the optimal schedule and (b) and (c) show best unidirectional schedules. Note that the optimal schedule cannot be transformed into a unidirectional schedule without increasing the makespan or violating the precedence constraints of tasks in bay 3. Therefore, an optimization model for the QCSP with container groups must not restrict the structure to unidirectional schedules.

Although searching the space of unidirectional schedules can fail in finding the optimal solution, it might be a good strategy for solving the QCSP with container groups heuristically. The basic idea of the proposed heuristic is to search the space of unidirectional schedules exhaustively, ending up with an optimal schedule among the unidirectional ones. The *Unidirectional Scheduling (UDS) heuristic* respects all requirements of the QCSP including the issue of crane interference. It generates schedules by making decisions at three distinct levels:

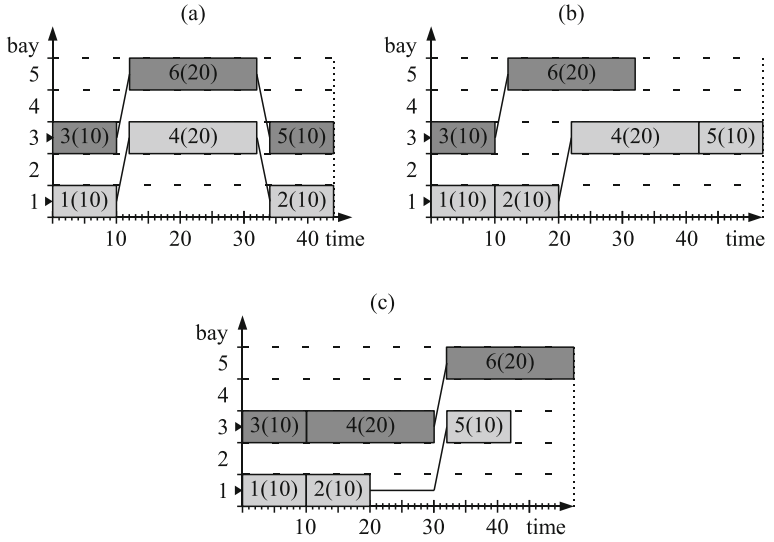

Fig. 7.5 A QCSP instance with a non-unidirectional optimal schedule (**a**) and best unidirectional schedules (**b**, **c**)

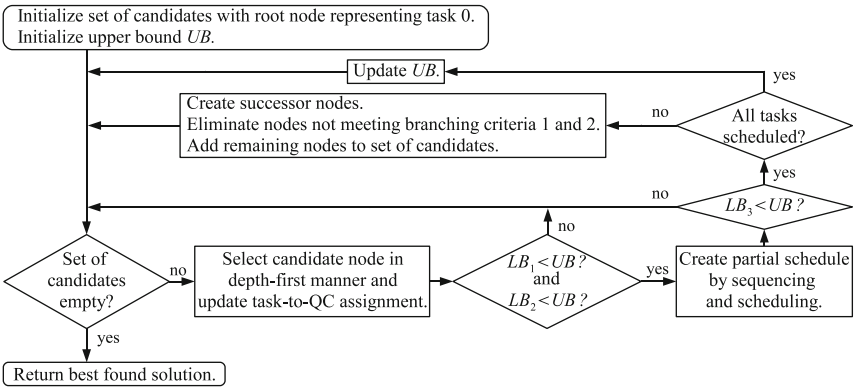

Fig. 7.6 Flowchart of the tree search

1. Task-to-QC Assignment: Using a tree search, the possible assignments of tasks to QCs are generated that allow for a unidirectional schedule.
2. Sequencing of tasks: For every considered task-to-QC assignment, sequences of tasks are determined that can be processed by QCs in a unidirectional movement.
3. Schedule building: Starting times for the tasks are iteratively determined with respect to the task sequences and the required temporal distances.

Task-to-QC assignments are the most complex decisions that need to be made if a unidirectional schedule is searched, see Lim et al. (2007). The UDS heuristic applies a tree search to generate these assignments as shown in Fig. 7.6. A detailed description of the procedure is given in Sect. 7.2.2. Decisions regarding the sequencing and

scheduling of tasks result from a transformation of task-to-QC assignments. The transformation is used in the UDS heuristic to evaluate assignments and appears as a single component in the tree search procedure. The sequencing and scheduling of tasks are described in Sects. 7.2.3 and 7.2.4, respectively.

Due to the initial positioning of the QCs, the optimal unidirectional schedule with a downward movement of cranes can differ from the optimal unidirectional schedule with an upward movement of cranes. Therefore, the above procedure must be applied twice to a problem instance. First, a unidirectional schedule is generated for an upward movement of the QCs. Afterwards, the bays are numbered in inverse order and the starting positions of QCs are adapted. Then a unidirectional schedule is generated for a downward movement of the QCs. The UDS heuristic delivers the better of the two schedules.

7.2.2 Assignment of Tasks to Cranes

Lexicographical Sorting: In the first step of the assignment procedure, the tasks involved in a problem are sorted in lexicographical order of increasing bay position and, within a bay, by precedence relations. Afterwards, the tasks are indexed according to the lexicographical sorting. An example is given by the tasks in Table 7.1. As stated in Sect. 7.1.1, it is assumed for this study that the precedence constraints completely determine the order for processing the tasks within every bay. Hence, the lexicographical sorting is unique and the procedure searches the entire space of unidirectional schedules. If no unique sorting is given, one may choose practical precedence constraints to avoid the evaluation of an exponentially growing number of different lexicographical sortings. Of course, in this case the UDS heuristic searches only a subset of the unidirectional schedules and does not necessarily end up with the best possible one.

Structure of the Tree Search: The Branch-and-Bound procedure builds up an enumeration tree starting with a root node representing task 0. At the first level of the tree, task 1 is assigned to QC $m_1 \in Q$. At tree level 2, task 2 is assigned to QC $m_2 \in Q$, and so on. Figure 7.7 shows a sketch of the search tree for the QCSP instance in Table 7.1. Since nine tasks are assigned to two QCs, each path from the root to a leaf in the tree corresponds to one of the 2^9 task-to-QC assignments. The bold printed path in the tree represents the task-to-QC assignment that underlies the schedule depicted in Fig. 7.4.

To calculate an initial upper bound UB for the tree search, two task-to-QC assignments are generated using the *S-TASKS* rule and the *S-LOAD* rule, as introduced by Sammarra et al. (2007). They divide the entire task set into q subsets such that the number of tasks (*S-TASKS*) or the workload (*S-LOAD*) is almost equally shared among the cranes. The delivered task-to-QC assignments are evaluated by the sequencing and scheduling procedures as described later in Sects. 7.2.3 and

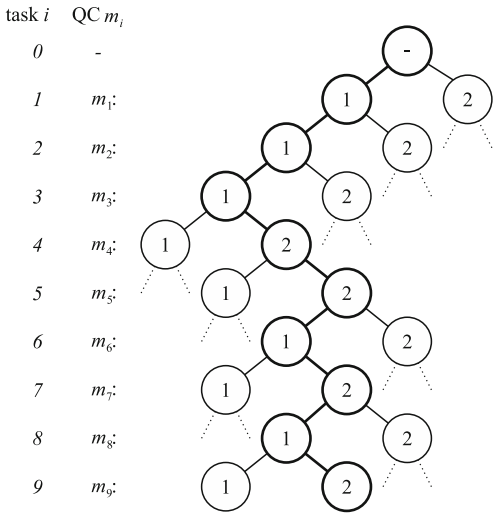

Fig. 7.7 Sketch of the search tree indicating the task-to-QC assignment

7.2.4. The shorter makespan computed for the two assignments serves as the initial upper bound.

The tree is searched in depth-first manner. A node is inspected by computing up to three lower bounds for the partial assignment. It is pruned if a lower bound overshoots UB. Otherwise, branching criteria are applied to decide on the successor nodes of the current node. Nodes that have passed the criteria enter the set of candidates that require further inspection. The upper bound UB is updated if a new best schedule has been generated. The Branch-and-Bound procedure terminates if the set of candidates is empty.

Bounding: The unique path leading from the root to a certain node in the search tree represents a partial task-to-QC assignment. Three lower bounds are applied to decide whether the current node at level i is pruned or not.

Lower bound 1: A lower bound on the makespan for the addressed partial task-to-QC assignment is computed by estimating the points in time when the cranes finish service. The finishing time c^k of QC $k \in Q$ is estimated by

$$c^k = r^k + w^k + d^k. \tag{7.30}$$

Here, r^k denotes the ready time of QC k, w^k denotes the workload of QC k under the current partial task-to-QC assignment, and d^k denotes the minimum travel time of QC k to traverse between the bays. The workload of QC k is computed by $w^k = \sum_{j \in \Omega^k} p_j$, where Ω^k denotes the set of all tasks prior to or equal to task i

in the lexicographical order and assigned to QC k. The travel time of the crane is determined by its initial bay position and the distance between the lowest bay l_{lo} and the upmost bay l_{up} it has to visit. For unidirectional schedules it is calculated as $d^k = (|l_0^k - l_{\text{lo}}^k| + l_{\text{up}}^k - l_{\text{lo}}^k)\hat{t}$. Given the case that a crane is completely idle, i.e., its workload is equal to zero, c^k is also set to zero in order to neglect the possible influence of a late ready time. The first lower bound is determined by the maximum finishing time of all QCs

$$LB_1 = \max_{k \in Q}\{c^k\}. \tag{7.31}$$

Lower bound 2: This bound takes advantage of the lexicographical sorting of tasks, which ensures that unassigned tasks at level i address bay l_i or bays above. Consider two QCs k and v where k operates above v. Let c^k and c^v denote the corresponding estimated finishing times with $c^k > c^v$. Since the remaining unassigned tasks belong to bays located above the position of QC k, QC v must remain idle as long as QC k has not finished its service. This means that QC v is blocked for a time period of length $c^k - c^v$. More precisely, the time QC v is blocked by QC k is given by $b_v^k = \max\{w^k + d^k - c^v, 0\}$, which allows for the fact that late ready times do not effect blocking. For a node at level i of the search tree, the expected total occupation time of the cranes including time periods in which they are blocked by QC k is

$$o^k = \sum_{v=1}^{q} c^v + \sum_{v=1}^{k-1} b_v^k + \sum_{j=i+1}^{n} p_j. \tag{7.32}$$

It is calculated as the sum of the QC finishing times at the current state of task assignment plus the total blocking time and the remaining workload of tasks unassigned so far. The shortest possible makespan results if the total occupation time is uniformly distributed to the cranes. This leads to the second lower bound

$$LB_2 = \max_{k \in Q}\left\{\frac{o^k}{q}\right\}. \tag{7.33}$$

Lower bound 3: To obtain a third lower bound, a partial schedule for the given partial task-to-QC assignment is computed and its makespan is determined as described in the subsequent Sects. 7.2.3 and 7.2.4. This makespan serves as LB_3. Obviously, LB_3 dominates LB_1 but the incurred computational cost is comparably high. Therefore, LB_3 is computed only if LB_1 and LB_2 did not effect a bounding. If a partial schedule is completed by the last task n and $LB_3 < UB$ holds, the schedule represents a new best solution. In this case UB is updated by LB_3.

Branching: To continue the partial task-to-QC assignment, nodes passing the bounding criteria are branched by adding successor nodes to the search tree at level $i+1$. The successor nodes represent possible assignments $m_{i+1} = k$ of task

$i+1$ to one of the QCs $k \in Q$. They must meet two branching criteria, which limit the tree search to the inspection of promising task-to-QC assignments for which a unidirectional schedule can be created.

Branching criterion 1: Unidirectional schedules show no change in the direction of QC movement after the initial crane repositionings. The first branching criterion prohibits task-to-QC assignments that lead to unavoidable changes in the direction of movement. Such a change is inevitable for precedence constrained tasks of the same bay if the QC of the succeeding task operates above the QC of the preceding task. Formally, for precedence constrained task pairs $(j,i+1) \in \Phi$ the following condition must hold

$$m_j \geq m_{i+1}. \tag{7.34}$$

The existence of precedence constraints leads to a strong reduction of branches in the tree search. Note that task $i+1$ is assigned to QC 1 if task j is assigned to QC 1. Task $i+1$ is assigned either to QC 1 or to QC 2 if task j is assigned to QC 2. Generally, task $i+1$ is assigned to one of the QCs $1,2,\ldots,m_j$. For this reason, at tree level i, at most m_j nodes are created for further inspection.

Branching criterion 2: With the only exception of the initial repositioning, QCs are not allowed to move downward in a unidirectional schedule. However, repositioning QC v downward does not make sense if another QC w can reach the bay position of the considered task $i+1$ earlier, because w will stay idle while v processes task $i+1$. Therefore, if adjacent QCs v and $w = v-1$ are not involved in the partial task-to-QC assignment, a successor node $m_{i+1} = v$ is only added to the search tree, if the following condition holds:

$$r^v + t^v_{0,i+1} \leq r^w + t^w_{0,i+1}, \tag{7.35}$$

As an example consider the instance in Fig. 7.4 but assume that task 1 is assigned to the upper QC ($v = 2$). This forces the lower QC ($w = 1$) to move downward and stay idle. Since there are no ready times given for both cranes and QC v has a longer travel time to reach bay 1, Condition (7.35) does not hold, which prevents a further investigation of this partial task-to-QC assignment.

7.2.3 Sequencing of Tasks

After the tasks have been assigned to QCs, a task sequence is determined for each crane. Although a feasible schedule can be derived from any combination of QC task sequences, only one certain combination of sequences leads to a unidirectional schedule. Due to the lexicographical sorting, the right sequences are already determined by the order in which the tasks have been treated in the assignment process. In other words, the first task assigned to a QC is the first in its sequence, the next assigned task is the second in its sequence, and so on. Each change in a

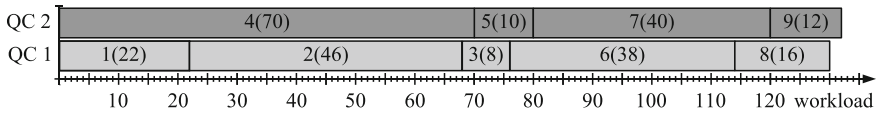

Fig. 7.8 Task sequences for the task-to-QC assignment of Fig. 7.7

sequence means that a unidirectional schedule can no longer be built. Precedence constrained tasks can obviously not be changed in sequence. Changing the sequence of unconstrained tasks belonging to different bays inevitably requires a change in the moving direction of the crane. Consequently the sequencing of tasks follows the lexicographical sorting which entails no computational effort for the UDS heuristic.

Example 7.2: Sequencing of tasks

From the task-to-QC assignment shown in Fig. 7.7 one obtains task sequences 1-2-3-6-8 and 4-5-7-9 for QCs 1 and 2, respectively. The sequences and the resulting distribution of the workload among the two cranes are shown in Fig. 7.8.

7.2.4 Scheduling of Tasks

Generally, one can build different schedules from a set of QC task sequences depending on the priority given to tasks which are forbidden to be processed simultaneously. Therefore, a schedule generation scheme is applied to assign priorities to tasks. In order to generate non-delay crane schedules, Kim and Park (2004) apply the list scheduling scheme. In a non-delay schedule, cranes do not remain idle while they could start processing a task. Since the set of non-delay schedules does not necessarily contain the optimal solutions, the approach conducts a heuristic reduction of the search space. Lim et al. (2007) generate unidirectional schedules by scheduling tasks in the order of increasing bay position, i.e., priority is given to the task with lower bay position. This schedule generation scheme can, however, not be applied to QCSP formulations that respect a safety margin between QCs. A safety margin can necessitate to give priority to a task with a higher bay position in order to generate an optimal unidirectional schedule (see tasks 3 and 4 in the schedule shown in Fig. 7.4 on page 94).

A schedule generation scheme capable of capturing the optimal QCSP schedule is based on the disjunctive graph model, which is well known in the field of machine scheduling. In the approach of Sammarra et al. (2007) the sequencing and the scheduling of QC tasks are commonly based on a problem representation using disjunctive graphs.

In the study at hand the disjunctive graph model is applied to build a unidirectional schedule for every task-to-QC assignment generated in the tree search. In face of a large number of possible task-to-QC assignments, the model turns out to be very valuable because it reveals an efficient way for incorporating crane interference issues into the schedule generation scheme.

In the disjunctive graph model, all tasks $i \in \overline{\Omega}$ are represented as nodes. The task sequence of crane $k \in Q$ is represented by a set of conjunctive arcs A_k which defines a path from node 0 to node T. The precedence constrained task pairs of set Φ are represented by a further set of conjunctive arcs A_Φ. The set of all conjunctive arcs is defined as $A = \bigcup_{k \in Q} A_k \cup A_\Phi$. The further task pairs that are forbidden to be processed simultaneously are represented by pairs of disjunctive arcs. A pair of disjunctive arcs between tasks expresses the two possible orders of processing them. The set of disjunctive arcs in the graph is denoted as D. It contains arcs for pairs of non-simultaneous tasks defined in Ψ. Additionally, D contains arcs for task pairs which are not precedence constrained but can cause crane interference under the given task-to-QC assignment. This latter task set is defined as $\Psi_\Theta = \{(i,j) \in \Omega^2 \backslash \Phi \mid (i,j,m_i,m_j) \in \Theta\}$. Now, the set of all disjunctive arcs is $D = \{(i,j) \in \Omega^2 \mid (i,j) \in \Psi \cup \Psi_\Theta \vee (j,i) \in \Psi \cup \Psi_\Theta\}$. Note that if tasks i and j cannot be processed simultaneously, both arcs, (i,j) and (j,i), must enter D.

Weights are defined for conjunctive and disjunctive arcs in different ways. Weights for the conjunctive arcs $(i,j) \in \bigcup_{k \in Q} A_k$ are given by

$$w_{ij} = \begin{cases} r^{m_j} + t_{0j}^{m_j}, & \text{if } i = 0 \text{ and } j \in \Omega, \\ p_i, & \text{if } i \in \Omega \text{ and } j = T, \\ p_i + t_{ij}, & \text{otherwise.} \end{cases} \tag{7.36}$$

These weights assess a processing time of a task or a ready time of a crane plus the travel time needed by a QC to move to the bay position of the next task. Weights for the conjunctive arcs $(i,j) \in A_\Phi \backslash \bigcup_{k \in Q} A_k$ which belong to precedence constraints not contained in the task sequences and weights for the disjunctive arcs $(i,j) \in D$ are defined slightly differently by

$$w_{ij} = p_i + \Delta_{ij}^{m_i m_j}. \tag{7.37}$$

Next to a task processing time these weights also reflect the required temporal distance for a safe crane movement, before the next task can be started. Summarizing, the disjunctive graph, which is obtained from a task-to-QC assignment, is denoted as $G = (\overline{\Omega}, A, D, W)$, where $W = [w_{ij}]$ represents the arc weights.

From the disjunctive graph G of a scheduling problem one can obtain a feasible schedule by selecting one arc of each pair of disjunctive arcs (and dropping the other) such that the resulting graph G' becomes acyclic. The unidirectional schedule is derived from G by always selecting those arcs from the pairs of disjunctive arcs that are directed from nodes of the upper QC toward nodes of the lower QC. This means that whenever two tasks cannot be processed simultaneously, the schedule generation scheme gives *priority to the upper QC*. Consequently, a cycle in G' can be effected only by arcs from A_Φ. However, due to the first branching criterion (7.34), arcs corresponding to precedence constraints are strictly downward oriented too and,

therefore, G' cannot become cyclic. The makespan of the resulting unidirectional schedule is computed as the length of the longest path in G'.

Example 7.3: Scheduling of QC tasks (continued Example 7.2)

The schedule generation is illustrated for the task-to-QC assignment shown in Fig. 7.7 and the corresponding task sequences shown in Fig. 7.8. Using the assignment and the task sequences, the node and arc sets shown in Table 7.3 are generated as described above. Here, A_1 and A_2 are the arc sets representing the task sequences of QCs 1 and 2, respectively. A_Φ represents the precedence constraint between tasks 1 and 2 and between tasks 5 and 6. The disjunctive arcs in D are used for handling crane interference.

The disjunctive graph G is shown in Fig. 7.9. The arc weights are calculated using (7.36) and (7.37). The composition of weights is shown in detail for arcs $(0,4),(4,5),\ (7,8)$, and $(9,T)$ in the figure.

Giving priority to the upper QC whenever two tasks cannot be processed simultaneously means selecting the downward oriented arc from each of the five disjunctive arc pairs of graph G. Figure 7.10 shows the directed graph G' that is obtained from G. The corresponding longest path $(0,4,5,7,8,T)$ is of length 145. This value measures the makespan of the schedule shown in Fig. 7.4. While a two-crane problem has been considered for illustrating the proposed heuristic, the procedure can be applied without restrictions to generate unidirectional schedules for larger problems.

Table 7.3 Objects for the construction of the disjunctive graph

$\overline{\Omega} = \{0,1,2,3,4,5,6,7,8,9,T\}$
$A_1 = \{(0,1),(1,2),(2,3),(3,6),(6,8),(8,T)\}$
$A_2 = \{(0,4),(4,5),(5,7),(7,9),(9,T)\}$
$A_\Phi = \{(1,2),(5,6)\}$
$D = \{(3,4),(4,3),(4,6),(6,4),(4,8),(8,4),(5,8),(8,5),(7,8),(8,7)\}$

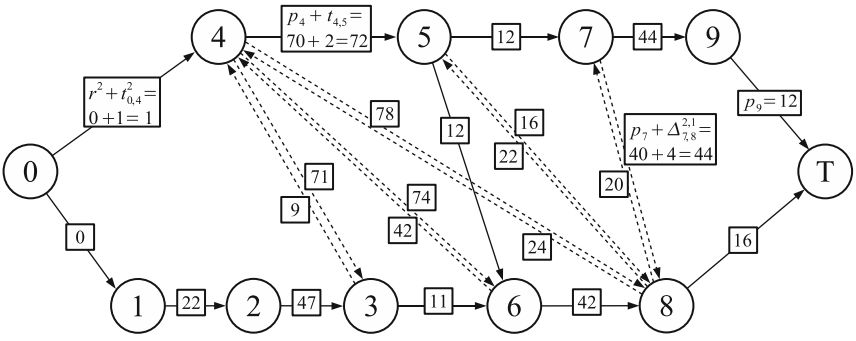

Fig. 7.9 Graph G obtained for the task-to-QC assignment of Fig. 7.7

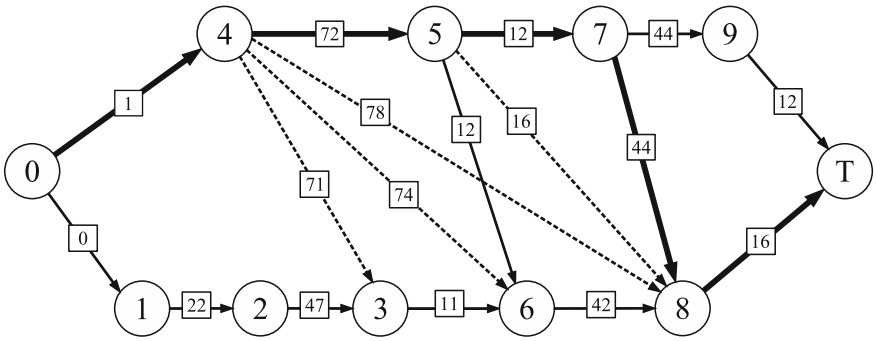

Fig. 7.10 Graph G' corresponding to Fig. 7.9

7.3 The QCSP with Time Windows

The computational study of the BACAP has revealed that the quality of berth plans is improved if variable-in-time QC-to-Vessel assignments are considered. While the above QCSP formulation can handle non-zero ready times of cranes, it cannot be applied if QCs are (temporarily) removed from a vessel during the service. Hence, the incorporation of *time windows for the cranes* becomes necessary. A time window defines a time span at which a crane is available at the vessel. The resulting problem is called the *Quay Crane Scheduling Problem with Time Windows* (QCSPTW). To access the QCSPTW a consistent declaration of time windows for cranes is discussed, followed by a mathematical formulation of the problem. Afterwards, an adaptation of the UDS heuristic is presented to solve the QCSPTW.

7.3.1 Declaration of Time Windows for Cranes

Since cranes can be temporarily removed from a vessel during the service, a crane can possess multiple time windows. The following notation is used to state time windows for a crane $k \in Q$. Let $TW_k = \{1, \ldots, \tau_k\}$ denote the set of time windows of k where τ_k is the number of time windows.

Each time window $u \in TW_k$ is defined by the quadruple $(r^{ku}, d^{ku}, l_0^{ku}, l_T^{ku})$ with:

- The ready time r^{ku}, i.e., the begin of the time window
- The withdraw time d^{ku}, i.e., the end of the time window
- The initial crane position l_0^{ku} at the begin of the time window
- The final crane position l_T^{ku} at the end of the time window

Without loss of generality it is assumed that the time windows of a crane are indexed according to increasing ready times, i.e., $r^{k,1} < r^{k,2} < \cdots < r^{k,\tau_k}$.

In order to obtain feasible QCSPTW solutions, time windows for cranes must be consistently declared. For example, if overlapping time windows are declared for

two non-adjacent QCs, a corresponding time window needs to be declared for every in-between crane because in-between cranes cannot be removed from the vessel during the respective time span. Furthermore, the initial (final) positions of cranes which approach (are removed from) the vessel at the same time have to respect the safety margin and the non-crossing condition.

A specific QC-to-Vessel assignment as generated within the berth planning phase can serve as a basis for the declaration of consistent time windows. Furthermore, the following assumptions are made to simplify the declaration of time windows:

1. Ready times and withdraw times of QCs refer to full hours only.
2. There is sufficient clearance between vessels for positioning cranes.
3. The time needed to move a QC from one vessel to another vessel is neglected.
4. QCs which serve the considered vessel in its last handling hour (according to the given QC-to-Vessel assignment) stay until the service is completed.

The first assumption is justified because the BACAP assigns cranes to vessels on an hourly basis. The second assumption ensures that approaching cranes and removing cranes can be positioned outside of the vessel area without getting into conflict with cranes serving other vessels. The third assumption has already been stated for the BACAP. This simplification allows to focus on the considered vessel without taking into account the origin of an approaching QC or the destination of a removed QC. The fourth assumption ensures that there is always a feasible solution for a QCSPTW instance existing under any QC-to-Vessel assignment. For this purpose the latest-starting time window τ_k of those cranes $k \in Q$ which are assigned to the vessel in its last handling hour receives an infinite withdraw time. Such time windows are called *open-ended* time windows. An open-ended time window enables a QC to remain at the vessel and complete handling operations. It is mathematically described by $d^{k,\tau_k} = M$.

Following these assumptions *ready times* and *withdraw times* of cranes can be derived straightforward from a specific QC-to-Vessel assignment of a BACAP solution. Merely a time transformation is required. The berthing time of the considered vessel in the BACAP is transformed into time 0 in the QCSPTW while the time unit is changed from hours to minutes. This transformation is illustrated in Fig. 7.11 where a single vessel berths at time $s_1 = 3$ and departs at time $e_1 = 9$. Consider QC 1, which approaches the vessel 1 h after the berthing time. The crane is removed from the vessel 2 h later. Hence, the corresponding time window $u = 1$ of QC $k = 1$ in the QCSPTW shows a crane ready time $r^{1,1} = 60$ and a crane withdraw time $d^{1,1} = 180$.

The *initial crane positions* of time windows are determined differently depending on the crane ready time. Cranes which are assigned to the vessel at its berthing time are lined up alongside the vessel with inter-crane clearance as required from the safety margin. Cranes which approach the vessel at a later point in time are initially positioned outside of the vessel area. For a vessel with b bays, these outside positions can be addressed by virtual bays $0, -1, \ldots$ and $b+1, b+2, \ldots$, respectively. Assume that a QC v approaches a vessel while the service is running. Let w_{lo} and w_{up} denote

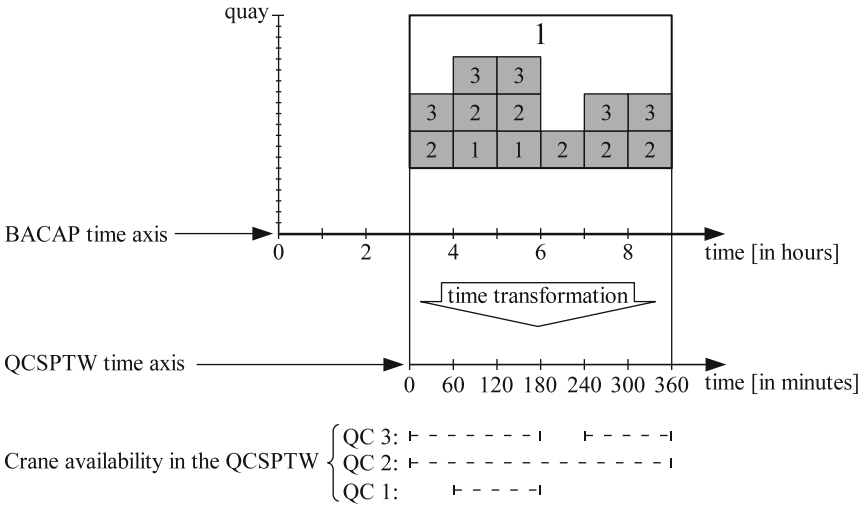

Fig. 7.11 Time transformation between BACAP and QCSPTW

the downmost crane and the upmost crane among the already assigned QCs. The initial position of crane v at the begin of time window $u \in TW_v$ is given by

$$l_0^{v,u} = \begin{cases} 1 - \delta_{v,w_{\text{lo}}}, & \text{if } v < w_{\text{lo}}, \\ b + \delta_{v,w_{\text{up}}}, & \text{if } v > w_{\text{up}}. \end{cases} \tag{7.38}$$

Here $\delta_{v,w_{\text{lo}}}$ ($\delta_{v,w_{\text{up}}}$) denotes the smallest allowed difference between the bay positions of QCs v and w_{lo} (w_{up}) as calculated by (7.7) on page 89. Eq. (7.38) positions a crane at the nearest feasible position outside of the vessel area. The first case of (7.38) applies if v is a crane positioned below the already assigned cranes at the quay. It can be seen that v is initially positioned below bay 1. For example, if the safety margin is set to $\delta = 1$ bay, and v is the crane adjacent to w_{lo} ($v = w_{\text{lo}} - 1$), crane v is initially positioned at bay $l_0^{v,u} = 1 - \delta_{v,w_{\text{lo}}} = 1 - 2 = -1$. If the next crane $v' = w_{\text{lo}} - 2$ approaches the vessel at the same time, the initial position of v' is set to bay $l_0^{v',u'} = 1 - \delta_{v',w_{\text{lo}}} = 1 - 4 = -3$, and so on. The second case of (7.38) applies if v is a crane positioned above the already assigned cranes at the quay. In this case the initial position is set to a bay position above bay b.

For calculating *final crane positions* of time windows, (7.38) is modified by replacing $l_0^{v,u}$ by $l_T^{v,u}$. Furthermore, v refers to the QC that is removed from the vessel, and w_{lo} (w_{up}) refers to the downmost (upmost) QC among the cranes that remain at the vessel. From these modifications (7.38) calculates the final position of a removed crane at the end of its time window. Note that the equation cannot be applied to open-ended time windows because w_{lo} and w_{up} are not defined at the end of a vessel's service. Since the final crane positions of open-ended time windows have no impact on the makespan of a schedule, they are of no particular interest within the QCSPTW. For reasons of completeness the final positions of cranes with

an open-ended time window are lined up alongside the vessel as described for initial positions above.

Crane positions as determined above ensure that the non-crossing requirement and the safety margin are respected among approaching and removing cranes at any time. Conflicts with operating QCs are avoided as well. However, one potential conflict remains. According to the assumptions made for the QCSP, a crane can move out of the vessel area in order to let another crane process a task. However, in the QCSPTW, this crane may occupy the initial position of approaching cranes. This conflict is avoided by the interference constraints introduced for the QCSP. They insert a sufficient temporal distance between the processing of consecutive tasks to allow a safe movement of cranes, which is also sufficient for temporarily idle cranes. The example in Fig. 7.2b on page 88 gives an idea of this effect.

Example 7.4: Declaration of time windows

The declaration of time windows for cranes is demonstrated using the QC-to-Vessel assignment shown in Fig. 7.11. The vessel is assumed to have $b = 10$ bays and the safety margin is set to $\delta = 1$ bay. The following time windows are derived from the depicted QC-to-Vessel assignment:

QC 1 is assigned to the vessel from the beginning of the second service period to the end of the third service period. According to the time transformation, the corresponding time window in the QCSPTW shows a ready time of $r^{1,1} = 60$ and a crane withdraw time of $d^{1,1} = 180$. The initial position of the crane at the begin of the time window is calculated using (7.38) as $l_0^{1,1} = 1 - \delta_{1,2} = -1$, i.e., the crane is placed outside of the vessel area. The calculation takes into account that QC 2 is already assigned to the vessel at the ready time. Also the final position is set to $l_T^{1,1} = 1 - \delta_{1,2} = -1$ because QC 2 remains at the vessel while QC 1 is removed. Hence, one time window $(r^{1,1}, d^{1,1}, l_0^{1,1}, l_T^{1,1}) = (60, 180, -1, -1)$ is declared for QC 1.

QC 2 is assigned to the vessel throughout its entire handling time. A time window $(r^{2,1}, d^{2,1}, l_0^{2,1}, l_T^{2,1}) = (0, M, 1, 1)$ is declared for a consistent treatment. Note that this time window is open-ended as stated above by the fourth assumption. The initial position of the crane is taken from the lining up of those cranes that are assigned to the vessel at its berthing time. The final position is taken from lining up all cranes with an open-ended time window alongside the vessel.

The assignment of QC 3 to the vessel requires two time windows. The first time window is defined by $(r^{3,1}, d^{3,1}, l_0^{3,1}, l_T^{3,1}) = (0, 180, 3, 12)$. Here, the initial position follows from lining next to QC 2 with respect to the safety margin. The final position is calculated by $b + \delta_{3,2} = 12$ which places the QC outside of the vessel area and, thereby, respects that QC 2 remains at the vessel. The second time window of QC 3 is $(r^{3,2}, d^{3,2}, l_0^{3,2}, l_T^{3,2}) = (240, M, 12, 3)$.

7.3.2 Optimization Model

A mathematical formulation of the QCSPTW is derived by extending the QCSP formulation given in Sect. 7.1. The formulation uses the additional terms $t_{0i}^{ku} = \hat{t}|l_0^{ku} - l_i|$ and $t_{iT}^{ku} = \hat{t}|l_T^{ku} - l_i|$. The former denotes the travel time of QC k between the initial crane position of time window $u \in TW_k$ and the bay position of task $i \in \Omega$. The latter denotes the travel time between the position of task i and the final crane position at the end of the time window u. Moreover, binary decision variables y_i^{ku} are introduced, set to 1 if and only if task i is processed by QC k in its time window u.

The QCSPTW is formulated as follows:

$$\text{minimize } c_T \tag{7.39}$$

subject to

$$\sum_{u \in TW_k} y_i^{ku} = \sum_{j \in \Omega^0} x_{ji}^k \qquad \forall i \in \Omega, \forall k \in Q, \tag{7.40}$$

$$c_i - p_i \geq y_i^{ku}(r^{ku} + t_{0i}^{ku}) \qquad \forall i \in \Omega, \forall k \in Q, \forall u \in TW_k, \tag{7.41}$$

$$c_i \leq M(1 - y_i^{ku}) + d^{ku} - t_{iT}^{ku} \qquad \forall i \in \Omega, \forall k \in Q, \forall u \in TW_k, \tag{7.42}$$

$$y_i^{ku} \in \{0,1\} \qquad \forall i \in \Omega, \forall k \in Q, \forall u \in TW_k, \tag{7.43}$$

and (7.14)–(7.25), (7.27) –(7.29).

As in the QCSP, the objective pursues the minimization of the makespan of the schedule. Constraints (7.40) ensure that every task is processed within one time window of the assigned crane. Constraints (7.41) ensure that a task is not started earlier than the crane ready time of the addressed time window plus the time needed by the crane to move from its initial position to the task. Constraints (7.42) ensure that each crane is able to reach its final position at the withdraw time of a time window. The model is completed by the Constraints (7.14)–(7.25) and (7.27)–(7.29). The QCSPTW is classified by $Group, prec \mid TW, pos, move \mid cross, save \mid max(compl)$.

7.3.3 Adaptation of the UDS Heuristic

The UDS heuristic is adapted for the QCSPTW, referred to as the *UDSTW heuristic*. The following adaptations are necessary:

1. The task-to-QC assignments generated by the *S-TASKS* and the *S-LOAD* rule may not allow for the generation of feasible schedules in the presence of time windows. In this case both rules are repeated, but this time only cranes with an open-ended time window are considered. The derived task-to-QC assignments lead to feasible schedules. The obtained schedules yield the initial upper bound for the UDSTW heuristic.

2. The lower bounds that have been introduced for the UDS heuristic already respect ready times r^k and initial positions l_0^k of QCs. The bounds are applicable in the UDSTW heuristic by replacing these values with the corresponding values of the earliest time windows of the cranes, namely $r^{k,1}$ and $l_0^{k,1}$.
3. The second branching criterion enforces that a task is assigned to the QC that can reach it in the fastest possible way. However, if the chosen crane shows no time window sufficiently large to process the task, the QCSPTW cannot be solved feasible. For this reason, the second branching criterion is disabled in the UDSTW heuristic.
4. The scheduling rule derived from the disjunctive graph representation of the QCSP is applicable within the UDSTW heuristic. However, it may happen that (i) the derived earliest starting time of a task does not fall within a time window of the assigned QC, or (ii) the task cannot be completed before the addressed time window ends. In both cases the task is postponed by shifting it to the next time window of the assigned crane that is sufficiently large to allow processing the task and the required crane movement. Successor tasks and tasks assigned to QCs with lower priority are postponed accordingly. If no appropriate time window can be found for a task, the task-to-QC assignment does not lead to a feasible unidirectional schedule. The assignment is not further investigated by the UDSTW heuristic.

The UDSTW heuristic searches the space of unidirectional schedules of a QCSPTW instance. However, in the presence of time windows for the cranes, this search space reduction can exclude the optimal schedule even if no precedence relations exists among tasks. Figure 7.12 shows an optimal solution for a QCSPTW instance with two cranes assigned to the vessel. In this example, the first assumption stated in Sect. 7.3.1 is dropped, i.e., ready times and withdraw times of QCs do not refer to full hours in this example. QC 1 is available within two time windows $(r^{1,1}, d^{1,1}, l_0^{1,1}, l_T^{1,1}) = (0, 15, 1, -1)$ and $(r^{1,2}, d^{1,2}, l_0^{1,2}, l_T^{1,2}) = (25, M, -1, 1)$. QC 2 is assigned to the vessel during the entire service interval, which is represented by the single open-ended time window $(r^{2,1}, d^{2,1}, l_0^{2,1}, l_T^{2,1}) = (0, M, 3, 3)$. In order to capture the optimal solution with a makespan of 69 time units, QC 1 needs to process task 2 within its first time window and tasks 1 and 3 within its second time

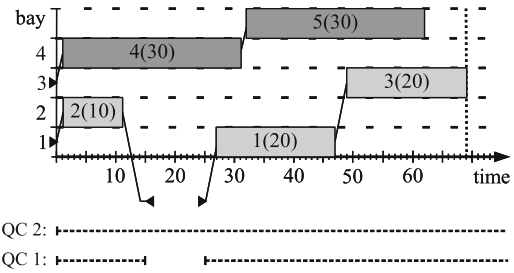

Fig. 7.12 A non-unidirectional optimal solution for a QCSPTW instance

window. However, this task sequence contradicts the lexicographical task indexing, i.e., the resulting schedule is not a unidirectional one. It can therefore not be generated by the UDSTW heuristic. The example illustrates that the reduction of the solution space to unidirectional schedules might have a stronger impact on the quality of solutions for the QCSPTW than for the QCSP. A quantitative investigation on the performance of the heuristic is left to the computational study in the subsequent section.

7.4 Computational Study

The following tests assess the competitiveness of the UDS heuristic against methods published in the literature, the sensitivity of the solutions on the problem parameter settings, and the performance of the UDSTW in solving the problem variant with time windows for the cranes:

- Performance of the UDS heuristic:

  Test 7.1: Performance on small QCSP instances
  Test 7.2: Performance on large QCSP instances
  Test 7.3: Solution quality and runtime demand

- Sensitivity on problem parameter settings:

  Test 7.4: Sensitivity on the task definition
  Test 7.5: Sensitivity on the safety margin

- Performance of the UDSTW heuristic:

  Test 7.6: Solution quality and runtime demand of the UDSTW heuristic

The heuristics have been implemented in JAVA. A PC P4 2.8 GHz is used for the tests.

Benchmark Instances

A suite of benchmark problems is used that has been introduced by Kim and Park (2004). It contains nine instance sets of different problem size with ten instances each, see Table 7.4. For each instance the order of processing the tasks of a bay is completely determined by precedence constraints, i.e., a unique lexicographical

Table 7.4 QCSP benchmarks of Kim and Park (2004)

	Set								
	A	B	C	D	E	F	G	H	I
# Tasks n	10	15	20	25	30	35	40	45	50
# QCs q	2	2	3	3	4	4	5	5	6

sorting of tasks is always possible. The QC ready times r^k are zero in all instances. The safety margin δ is set to one bay. The travel time $\hat{t}$ of QCs is set to one time unit per bay. Variations in these settings are stated in the respective tests.

Test 7.1: Performance on small QCSP instances

This test assesses the performance of the UDS heuristic and compares it with the QCSP solution methods published in the literature, see Table 7.5.

Since the studies of Kim and Park (2004), Moccia et al. (2006), and Sammarra et al. (2007) tackled only the first 37 instances, named *k*13 to *k*49, from the benchmark suite, the comparison is carried out on this subset of benchmarks. Moccia et al. (2006) report optimal solutions for 28 of these instances and tight lower bounds for the remaining nine instances. While it is unknown whether the solutions are feasible with respect to the corrected interference constraints, they can still serve as lower bounds. This holds because optimal solutions generated for a model with incomplete interference constraints will show a makespan that is not larger than the minimum makespan of the corrected model's solutions.

Table 7.6 shows the computational results of this first experiment. The optimal makespan or best known lower bound to each of the instances appears in column f_{opt}. Note that previous studies weighted the obtained makespan of a schedule by a fixed factor of 3, as is also done here to ease comparison. The performance of the methods is reported on the basis of the relative error *RE* in percent of the best found solution f_{best} against f_{opt}, i.e., $RE = (f_{\text{best}} - f_{\text{opt}})/f_{\text{opt}} \times 100$. For the UDS heuristic the objective function value of the best found solution is reported in cases where it fails to reach f_{opt}.

With the exception of the algorithms of Kim and Park (2004) all compared methods solve instance set A to optimality. The observed deviation between f_{best} and f_{opt} for the UDS solution of *k*22 results from the interference constraints (7.10)–(7.12) in the corrected model. Replacing these constraints by (7.2), (7.4), and (7.6), which are used by Sammarra et al. (2007), the UDS heuristic always reaches f_{opt}. For this reason it is assumed that the corrected solution for *k*22 is optimal with respect to the revised QCSP model. For instance sets B and C, the relative error of the Tabu Search clearly increases against the Branch-and-Cut algorithm and the UDS heuristic. The Branch-and-Cut algorithm and UDS generate schedules of identical quality. The relative error of 2.26% for the UDS solution of *k*42 stems again from the strict

Table 7.5 QCSP solution methods published in the literature

Abbr.	Method	Reference
B&B	Branch-and-bound	Kim and Park (2004)
GRASP	Greedy randomized adaptive search procedure	Kim and Park (2004)
B&C	Branch-and-cut	Moccia et al. (2006)
TS	Tabu search	Sammarra et al. (2007)

Table 7.6 Performance comparison of QCSP solution methods (*RE* in percent)

No.	Set	f_{opt}	B&B	GRASP	B&C	TS	UDS	f_{best}
*k*13	A	453	0.00	0.00	0.00	0.00	0.00	
*k*14	A	546	0.00	0.00	0.00	0.00	0.00	
*k*15	A	513	0.00	0.58	0.00	0.00	0.00	
*k*16	A	312	2.88	2.88	0.00	0.00	0.00	
*k*17	A	453	0.66	0.66	0.00	0.00	0.00	
*k*18	A	375	0.00	0.00	0.00	0.00	0.00	
*k*19	A	543	1.66	1.66	0.00	0.00	0.00	
*k*20	A	399	20.30	20.30	0.00	0.00	0.00	
*k*21	A	465	0.00	0.00	0.00	0.00	0.00	
*k*22	A	537	34.08	34.08	0.00	0.00	0.56	540[b]
*k*23	B	576	0.00	2.60	0.00	1.04	0.00	
*k*24	B	666	0.45	1.35	0.00	0.45	0.00	
*k*25	B	738	0.00	0.41	0.00	0.41	0.00	
*k*26	B	639	0.00	1.88	0.00	0.00	0.00	
*k*27	B	657	0.00	4.57	0.00	0.46	0.00	
*k*28	B	531	1.13	3.39	0.00	0.00	0.00	
*k*29	B	807	0.00	1.49	0.00	0.37	0.00	
*k*30	B	891	0.00	1.68	0.00	0.00	0.00	
*k*31	B	570	0.00	0.00	0.00	0.00	0.00	
*k*32	B	591	0.00	1.02	0.00	0.00	0.00	
*k*33	C	603	0.00	10.45	0.00	0.00	0.00	
*k*34	C	717	0.00	6.28	0.00	2.51	0.00	
*k*35	C	684	0.88	2.19	0.00	0.88	0.00	
*k*36	C	678	6.19	4.42	0.00	0.44	0.00	
*k*37	C	510	1.18	5.88	0.00	1.76	0.00	
*k*38	C	613.67[a]	3.15	7.55	0.71	0.71	0.71	618
*k*39	C	508.38[a]	8.58	13.89	0.91	2.09	0.91	513
*k*40	C	564	2.13	5.85	0.00	0.53	0.00	
*k*41	C	585.06[a]	11.78	9.73	0.50	1.53	0.50	588
*k*42	C	560.31[a]	4.94	18.86	1.73	2.80	2.26	573[b]
*k*43	D	859.32[a]	10.67	9.62	4.38	2.29	1.94	876[c]
*k*44	D	820.35[a]	7.15	4.59	0.20	1.66	0.20	822
*k*45	D	824.88[a]	4.38	5.83	1.83	3.29	1.11	834[c]
*k*46	D	690	2.61	6.52	0.00	0.00	0.00	
*k*47	D	792	15.15	1.89	0.00	0.00	0.00	
*k*48	D	628.87[a]	6.38	6.38	2.56	5.43	1.61	639[c]
*k*49	D	879.22[a]	4.07	10.55	5.43	3.73	1.68	894[c]
ARE (%)			4.06	5.65	0.49	0.87	0.31	

[a]Lower bound; [b]Corrected solution; [c]New best solution

interference handling. The UDS heuristic returns the Branch-and-Cut solution if the new interference constraints are not applied. For the larger instances of set D, the Branch-and-Cut algorithm often fails to reach the optimum within the allowed runtime of 2 h. Here, the UDS heuristic is clearly superior to all other methods. For instances *k*43, *k*45, *k*48, and *k*49 it delivers new best solutions. This is also reflected by a comparison of the average relative error (*ARE*) observed for the heuristics. The

Table 7.7 Runtime comparison of QCSP solution methods (average-in-set in minutes)

Set	B&B	GRASP	B&C	TS	UDS
A	0.44	0.35	1.01	1.52	1.12×10^{-5}
B	17.53	1.46	8.91	5.86	3.68×10^{-5}
C	564.47	3.16	72.19	21.75	6.26×10^{-4}
D	809.73	7.56	102.49	48.68	3.43×10^{-3}

achieved excellent solution quality implies that at least the smaller instances of Kim and Park (2004) have optimal solutions which are unidirectional, too. In total the UDS heuristic is capable of solving all instances to optimality or to the best solution quality known so far.

The average computation time demand of the various algorithms (as reported in the literature) is presented for each of the four instance sets in Table 7.7. The machines used were a PC P2 466 MHz for the Branch-and-Bound method and the GRASP heuristic of Kim and Park, a PC P4 2.5 GHz for the Branch-and-Cut method of Moccia et al., a PC P4 2.66 GHz for the Tabu Search of Sammarra et al., and a PC P4 2.8 GHz for the UDS heuristic. Although within milliseconds, the computation times of the UDS procedure are specified in minutes for the purpose of comparability. Despite the fact that it has been tested on the fastest machine, it can be seen that the UDS heuristic tremendously cuts down the computation times.

Test 7.2: Performance on large QCSP instances

The performance of the UDS heuristic on the instance sets A to D encourages one to tackle the larger instances as provided in sets E to I of the benchmark suite, see Table 7.4. They contain problems with up to six QCs and 50 tasks as observed for large container vessels. It is supposed that previous studies had not tackled these problems because the proposed methods ran into their boundaries. Computational results obtained from the UDS heuristic for instances $k50$ to $k102$ are shown in Table 7.8. The UDS heuristic is given a runtime limit of 1 h. Recall from Sect. 7.2.1 that the procedure searches consecutively for unidirectional schedules with respect to upward and downward movements of the QCs. To ensure a fair allocation of computation time, both search processes are performed concurrently. In the event that one of the processes terminates within 30 min, the remaining computation time is made available to the other process. The reported runtime is the sum of the runtimes spent on searching in the two directions. If the limit of 60 min is exceeded, no runtime is reported. In these cases value f_{best} does not necessarily represent the optimal unidirectional schedule. To assess the quality of UDS solutions a lower bound is required. The lower bounds presented in Sect. 7.2.2 do not serve this purpose because they evaluate given (partial) task-to-QC assignments. Therefore, a lower bound on the makespan of the instances is calculated by solving a relaxed QCSP model given in Appendix C using CPLEX.

Table 7.8 Results of the UDS heuristic for the remaining instances in sets D to I

No.	*LB*	f_{best}	*RE*	*Time*	No.	*LB*	f_{best}	*RE*	*Time*
		D					E		
		(see Table 7.6 for *k*43–*k*49)			*k*53	657	717	9.13	–
					*k*54	753	774	2.79	0.02
					*k*55	663	684	3.17	0.01
					*k*56	666	690	3.60	0.22
					*k*57	681	705	3.52	0.24
					*k*58	765	786	2.75	0.17
					*k*59	666	687	3.15	0.01
*k*50	723	741	2.49	<0.01	*k*60	765	783	2.35	0.19
*k*51	777	798	2.70	<0.01	*k*61	618	639	3.40	0.04
*k*52	939	960	2.24	<0.01	*k*62	828	837	1.09	0.01
ARE (%)			2.48		*ARE* (%)			3.50	
		F					G		
*k*63	927	948	2.27	1.51	*k*73	837	870	3.94	31.71
*k*64	714	741	3.78	1.06	*k*74	822	843	2.55	4.71
*k*65	816	837	2.57	1.61	*k*75	657	675	2.74	0.37
*k*66	903	924	2.33	0.63	*k*76	825	852	3.27	0.90
*k*67	858	882	2.80	0.24	*k*77	672	699	4.02	1.27
*k*68	945	963	1.90	0.03	*k*78	621	642	3.38	8.96
*k*69	783	807	3.07	1.40	*k*79	717	744	3.77	1.52
*k*70	936	957	2.24	0.61	*k*80	720	750	4.17	1.28
*k*71	807	834	3.35	3.77	*k*81	705	738	4.68	1.28
*k*72	720	744	3.33	0.35	*k*82	696	717	3.02	1.03
ARE (%)			2.76		*ARE* (%)			3.55	
		H					I		
*k*83	921	948	2.93	6.37	*k*93	786	816	3.82	–
*k*84	876	897	2.40	3.29	*k*94	765	786	2.75	–
*k*85	945	972	2.86	5.82	*k*95	801	834	4.12	–
*k*86	786	816	3.82	–	*k*96	780	819	5.00	–
*k*87	840	867	3.21	–	*k*97	690	720	4.35	–
*k*88	744	768	3.23	43.73	*k*98	711	735	3.38	23.79
*k*89	822	843	2.55	10.96	*k*99	819	852	4.03	–
*k*90	1,023	1,053	2.93	24.95	*k*100	852	900	5.63	–
*k*91	810	837	3.33	10.74	*k*101	765	813	6.27	–
*k*92	873	897	2.75	34.61	*k*102	870	903	3.79	–
ARE (%)			3.00		*ARE* (%)			4.31	

The gained results of this test show that the UDS heuristic is capable of solving the majority of these instances within the runtime limit. Regarding instance sets E to H, merely three instances are not solved. Set I is the only set for which the heuristic fails in solving the majority of the instances. However, the observed deviation between *LB* and f_{best} ranges moderately within a few percent for all considered instances. This indicates that the UDS heuristic delivers schedules of good quality also for large instances.

Table 7.9 Solution quality at selected runtimes (ARE-in-set in percent)

	Set				
Time	E	F	G	H	I
0	16.00	12.73	19.67	14.77	20.34
1	3.50	2.76	3.75	3.35	4.95
10	3.50	2.76	3.55	3.07	4.60
60	3.50	2.76	3.55	3.00	4.31

Test 7.3: Solution quality and runtime demand

As shown by Test 7.2, the UDS heuristic may not terminate within an acceptable runtime if applied to large-sized instances. For this reason, the relation between runtime and solution quality is investigated here in order to determine a reasonable runtime limit for the heuristic. The test considers the large-sized instances of sets E to I. Table 7.9 reports the *ARE* for each set after running the UDS heuristic for 0, 1, 10, and 60 min. The values shown in the row of time 0 are those of the initial solutions.

As can be seen in the table, the solution quality is drastically improved within the first minute of computation for each of the five instance sets. For sets E and F, no further improvement is observed after that time. This can hardly surprise because most of these instances are solved within less than a minute, but it shows that also for instances with longer runtimes no further improvement takes place. For sets G to I, further improvements are observed. The most improvement, i.e., the largest reduction in the *ARE*, is observed for set I. However, even this *ARE* reduction is only 0.64%. It can be concluded that the UDS heuristic converges very quickly for all instance sets. A runtime limit of 1 min per instance is sufficient to ensure that finding solutions of acceptable quality is at a level of high likelihood.

Test 7.4: Sensitivity on the task definition

According to the QCSP classification scheme of Sect. 4.2.1, defining tasks on the basis of container groups is only one possibility. Two alternatives are to define tasks on the basis of complete bays, e.g., Lim et al. (2007), or on the basis of bay areas, e.g., Winter (1999). The advantage of the container group approach is that a more uniform distribution of workload among QCs can be achieved. However, the QCSP becomes more difficult to solve because a larger number of tasks and precedence relations between pairs of tasks come into the play. This test assesses the dependency of solution quality and computational effort on the different task definitions.

For the test, the QCSP instances and the UDS heuristic are slightly modified. First, within each instance, all tasks belonging to the same bay are combined into a single task. Applying the UDS heuristic to such an instance solves a QCSP with tasks defined on the basis of complete bays. Second, to solve the QCSP with tasks

Table 7.10 Results for different tasks definitions

	Container groups								
Set	A	B	C	D	E	F	G	H	I
ARE (%)	0.06	0.00	0.44	1.40	3.50	2.76	3.55	3.00	4.31
Max. RE (%)	0.56	0.00	2.26	2.70	9.13	3.78	4.68	3.82	6.27
Avg. time	<0.01	<0.01	<0.01	<0.01	6.09	1.12	5.30	26.05	56.38
	Complete bays								
Set	A	B	C	D	E	F	G	H	I
ARE (%)	3.00	2.89	1.77	1.88	4.70	3.73	5.25	3.90	4.78
Max. RE (%)	13.41	15.79	7.08	3.32	14.16	7.06	7.95	6.45	10.98
Avg. time	<0.01	<0.01	<0.01	<0.01	0.86	0.06	0.98	0.88	19.51
	Bay areas								
Set	A	B	C	D	E	F	G	H	I
ARE (%)	5.62	7.06	8.11	7.53	9.79	9.60	14.48	10.53	11.67
Max. RE (%)	19.55	36.32	15.07	12.86	15.53	17.10	23.85	17.04	18.70
Avg. time	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

defined on the basis of bay areas, a branching criterion is added to the UDS heuristic that forbids task-to-QC assignments which lead to overlapping operation areas of QCs at a vessel.

Table 7.10 shows comprehensive results for the benchmark sets if tasks are defined on the basis of container groups (taken from Tests 7.1 and 7.2), on the basis of complete bays, and on the basis of bay areas. It reports the *ARE* per set, the maximum *RE* observed for an instance within a set, and the required average runtime for solving instances of a set. The relative errors are calculated with respect to the optimal solutions and the lower bounds reported by Moccia et al. (2006) for instances up to $k49$ and with respect to the CPLEX lower bounds presented in Table 7.8 for the instances $k50$ to $k102$.

Comparing the results of tasks defined by container groups with tasks defined by complete bays, one can see that the *ARE*s differ by at minimum 0.47% (set I) and at maximum 2.94% (set A). From these findings the solution quality seems to deteriorate only little if tasks are defined on the basis of complete bays. However, the observed maximum *RE*s differ at a much higher rate. For sets A, B, E, and I the maximum *RE* is even above 10% if tasks are defined by complete bays. Defining tasks by container groups is clearly advantageous for the corresponding instances. The required runtimes decrease significantly if tasks are defined by complete bays. However, since the UDS heuristic can be prematurely terminated after 1 min if tasks are defined by container groups (see Test 7.3), the saving of computation time does not increase the attractiveness of a task definition on the basis of complete bays.

If tasks are defined by bay areas, runtimes are negligible even for the largest instances. However, the solution quality deteriorates drastically compared to

Table 7.11 Relative increase in makespan for different safety margins (average-in-set in percent)

δ	Set									Avg.
	A	B	C	D	E	F	G	H	I	
2	5.75	0.16	0.76	0.20	2.27	0.14	2.56	0.06	0.81	1.41
3	15.04	3.02	8.15	2.16	6.23	4.61	9.16	3.49	3.80	6.18
4	26.98	7.72	19.28	10.63	15.97	15.01	18.29	10.39	9.72	14.89

solutions with tasks defined by container groups. *ARE*s are above 10% for the large instances in sets G, H, and I. The maximum *RE* in set B shows that the makespan of an instance may increase by more than one-third compared to the solution with tasks defined by container groups. Since such an increase in the handling time of a vessel is unacceptable from a vessel operator's point of view, defining tasks by bay areas has to be rejected.

Test 7.5: Sensitivity on the safety margin

This test studies the impact of a safety margin on a schedule. Table 7.11 shows the relative change of the makespan observed in the instance sets for different safety margins against $\delta = 1$. Note, that $\delta = 0$ is not investigated because a QC's uprights occupy the bays adjacent to the crane's location and, therefore, positioning QCs at adjacent bays is technically forbidden as a matter of fact. The derived results confirm that the larger the safety margin is, the more the handling times of vessels increase. While the average increase over all sets is only 1.41% for $\delta = 2$, this value rises to 6.18% and even 14.89% for $\delta = 3$ and $\delta = 4$, respectively. Not surprisingly, the small-sized vessels in instance set A suffer most from a large safety margin, but also large-sized instances show a considerable increase in the makespan. The fluctuation which is observed for a certain value of δ stems from the varying number of bays and QCs involved in the different instance sets, see Table 7.4. The test results indicate that incorporating safety margins in the QCSP model is by no means marginal, it is an increasing need, the more safety requirements grow.

Test 7.6: Solution quality and runtime demand of the UDSTW heuristic

A final test assesses the performance of the UDSTW heuristic in solving crane scheduling instances with time windows for the cranes. For the test, the benchmark instances are modified by declaring time windows for the two upmost QCs as follows. The cranes are assigned to a vessel for the first 2 h of service, removed from the vessel for the following 2 h, reassigned for two more hours, and then finally removed. For instances in sets A and B, removing two QCs from a vessel means to interrupt the service process, which, however, can be dealt with by UDSTW. The UDSTW heuristic is given a runtime limit of 1 h per instance. Table 7.12 shows for

Table 7.12 Results of the UDSTW heuristic

Set	A	B	C	D	E	F	G	H	I
ARE (%)	9.10	8.64	7.04	11.75	7.70	9.44	7.74	9.19	10.35
Max. RE (%)	13.29	12.50	9.96	20.25	11.46	12.87	14.20	12.91	19.46
Avg. time	<0.01	<0.01	0.08	0.83	50.41	60.00	60.00	60.00	60.00

Table 7.13 Solution quality for the QCSPTW at selected runtimes (ARE-in-set in percent)

time	set				
	E	F	G	H	I
0	62.25	64.66	52.10	52.90	46.75
1	7.95	9.80	7.89	9.32	10.68
10	7.76	9.47	7.80	9.27	10.41
60	7.70	9.44	7.74	9.19	10.35

every instance set A to I the *ARE* over the contained instances, the maximum *RE* observed for the instances, and the required average runtime. For calculation of the relative errors, a lower bound on the makespan of QCSPTW instances is derived from solving the mathematical model in Appendix D by CPLEX.

As can be seen from this table, the solutions show considerable relative errors. Although the heuristic terminates within the runtime limit for instances in sets A to D, which means that the best unidirectional schedule has been found, *ARE*s of about 10% and a maximum *RE* of more than 20% (set D) are observed. However, also for the larger instances, where UDSTW systematically fails to terminate within the runtime limit, similar *ARE*s and maximum *RE*s are observed. This means that the solution quality does hardly deteriorate in the problem size.

The fact that the UDSTW heuristic does not terminate within the runtime limit does not necessarily mean that it is unable to find good solutions quickly. For this reason, similar to Test 7.3, the *ARE*s observed for instance sets E to I after running the heuristic for 0, 1, 10, and 60 min are reported in Table 7.13.

It can be seen that the UDSTW heuristic drastically improves the initial solutions in the first minute of computation. Different to the QCSP, further improvements are observed for all instance sets even after 10 min of runtime. However, these improvements are only marginal. Although the comparably high *ARE*s indicate further improvement potential, it can be concluded that the UDSTW heuristic delivers solutions of acceptable quality for the QCSPTW even if terminated after 1 min of runtime.

7.5 Summary

Within this chapter the problem of QC scheduling on the basis of container groups has been studied, as pioneered by Kim and Park (2004). The problem formulation considers crane scheduling in detail by incorporating crane interference constraints

and by respecting travel time for crane movement. However, the QCSP model and also revised versions presented in later papers do not detect crane interference in every case. To derive a correct QCSP model, a set of new interference constraints has been formulated. A so-called UDS heuristic is used to solve the problem. It works on a reduced search space of unidirectional schedules. Computational tests demonstrate the power of the heuristic. It clearly outperforms all existing approaches to the QCSP with container groups, in terms of solution quality as well as in terms of computation times. It confirms that defining tasks by container groups leads to better solutions than task definitions on the basis of complete bays or bay areas. Furthermore, safety margins have a strong impact on the makespan of QC schedules and, therefore, need to be incorporated into practical QCSP formulations.

Moreover, a variant of the QCSP that respects time windows for cranes, referred to as the QCSPTW, has been formulated. The UDS heuristic has been adapted to solve this problem. The resulting UDSTW heuristic solves with difficulty medium-sized and large-sized QCSPTW instance within a runtime limit of 60 min. Nevertheless, the heuristic delivers solutions of good quality after a runtime of 1 min.

Summarizing this study, rich formulations for QC scheduling problems have been derived, which can be solved to good or even optimal solution quality within short runtimes by the proposed heuristics. These properties enable a functional integration of crane scheduling into the BACAP.