Constraint Programming Approach to Quay Crane Scheduling Problem

by

Celal Özgür Ünsal

A Thesis Submitted to the

Graduate School of Engineering

in Partial Fulfillment of the Requirements for

the Degree of

Master of Science

in

Industrial Engineering

Koç University

February, 2013

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Celal Özgür Ünsal

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Committee Members:

Ceyda Oğuz, Ph.D. (Advisor)

Serpil Sayın, Ph.D.

Deniz Aksen, Ph.D.

Date: 08.02.2013

ABSTRACT

Minimizing the average vessel berthing time is one of the challenges for container terminals. Since containers are deployed from vessels by a quay crane, operations of this huge equipment may cause a bottleneck for the overall performance of a terminal. This study examines the quay crane scheduling problem (QCSP) at the seaside of container terminals. The QCSP requires completion of all loading and unloading operations of a berthed vessel. A constraint programming (CP) model, which consists of global constraints and propositional logic, is constructed by taking numerous properties of the problem such as safety margins, travel times and precedence relations into account. The performance of the proposed CP model is compared with algorithms presented in recent QCSP literature. The result from the computational experiments indicates that the proposed CP model is able to produce good results for the QCSP while reducing the computational time. Lastly, to show the robustness and the flexibility of the proposed model, extensions of the problem with ready times and time windows are also discussed.

Keywords: Quay Crane Scheduling, Constraint Programming, Container Terminals, Ready Times, Time Windows

ÖZET

Gemilerin rıhtıma bağlı kalma sürelerinin enazlanması konteyner terminallerinin önemli sorunlarındandır. Bu çalışmada konteyner terminallerindeki rıhtım vinci çizelgeleme problemi incelenmiştir. Rıhtım vinci çizelgeleme (RVÇ) probleminde rıhtıma yerleşmiş bir gemideki tüm konteynerlerin boşaltılması ve yerine yeni konteynerlerin yüklenmesi işlemlerinin tamamı gerçekleştirilir. Bu problem için vinçlerin hareket süreleri, güvenlik mesafeleri, işlerin öncelik ilişkileri, vinçlerin birbirini geçmemesi gibi birçok kısıt içeren zengin bir kısıt programlama (KP) modeli geliştirilmiştir. Bu KP modeli, literatürde daha önce yapılmış düzeltmeler dikkate alınarak, tamamen evrensel kısıtlar ve basit mantıksal kısıtlarlarla oluşturulmuştur. Geliştirilen KP modelinin performansı, literatürde sunulan en güçlü çözüm yöntemleriyle karşılaştırılmıştır. Hesaplamalı deneyler, RVÇ problemi için geliştirilen KP modelinin istikrarlı bir biçimde en iyiye çok yakın sonuçlara daha önceki çözüm sürelerini önemli miktarda azaltarak ulaştığını göstermiştir. Yöntemin esnekliğini göstermek için RVÇ probleminin vinçlerin hazır olma sürelerinin ve zaman pencerelerinin olduğu çeşitleri de çalışılmıştır.

Anahtar Kelimeler: Kısıt Programlama, Konteyner Terminalleri, Rıhtım Vinci Çizelgeleme , Hazır Olma Zamanları, Zaman Pencereleri

to Rococo…

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

As the distance between manufacturing and consumption locations increases, the container traffic and the competition between the container terminals grow accordingly. To exist in this competitive market, the container terminal operations must be efficient. In any container terminal, all of the inbound and the outbound containers pass through the seaside. Some important seaside operations in container terminals and their relationships are shown in Figure 1.1. Therefore, a container terminal management should allocate berth positions, assign (allocate) and schedule quay cranes for each vessel which is going to arrive to the terminal.

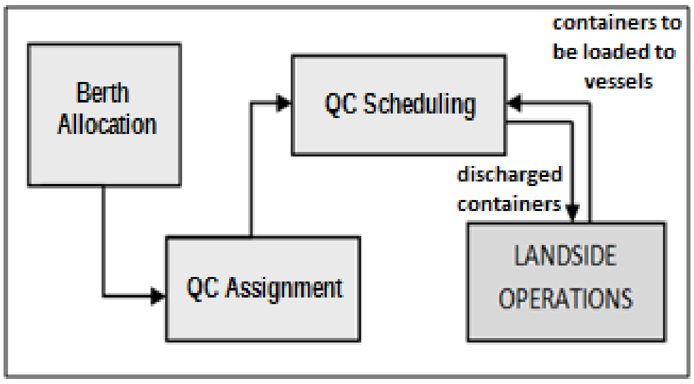


Figure 1.1: Seaside operations in container terminals.

Comprehensive surveys for the problems related to the seaside operations in container terminals are presented by Vis and de Koster (2003), Steenken *et al* (2004), and Stahlbock and Voß (2008).

In this thesis, scheduling of quay cranes is studied. A quay crane (QC) is huge equipment which is used to load and unload container vessels in container terminals (see Figure 1.2). The speed and the reliability of QC operations strongly depend on work schedules which are projected for the QCs. In the real-world, the quay crane scheduling significantly affects the waiting time of a container vessel since quay cranes are the interface between the land and the water side in any port container terminal.

At the same time, a QC is the most expensive equipment in any container terminal and this restricts the number of QCs that a terminal equips. Therefore, it is a prerequisite to carry out QC operations efficiently to provide a high quality service to vessels with a limited number of QCs.

Also the nature of the problem will entail a large number of side constraints. Accordingly there is a need for an alternative solution technique which can respond to wide variety of changes (additions) in the problem. Therefore the motivation in this thesis is to propose a solution technique to QCSP which is not only able to generate high quality results, but also is flexible to the changes in the problem.

The thesis is organized as follows. In Chapter 1, some key elements of container terminals are described. Then container terminal operations are classified and briefly introduced. Chapter 2 describes the problem and then gives a comprehensive literature review. The methodologies used in this study are explained in Chapter 3; that is, a general introduction to constraint programming is presented. Firstly in Chapter 4, a mixed integer programming (MIP) model for QCSP is introduced. Then a new constraint programming model which is functionally analogous to MIP model is proposed for QCSP and a bin packing based mixed-integer programming model is developed for a relaxed version of the QCSP to find a lower bound to the original one. Later in Chapter 4, two different extensions of the problem, QCSP with ready times and QCSP with time windows are studied. At the end of Chapter 4, computational experiments and their results are also explained and discussed in details. The thesis concludes in Chapter 5 with a discussion of future research possibilities.

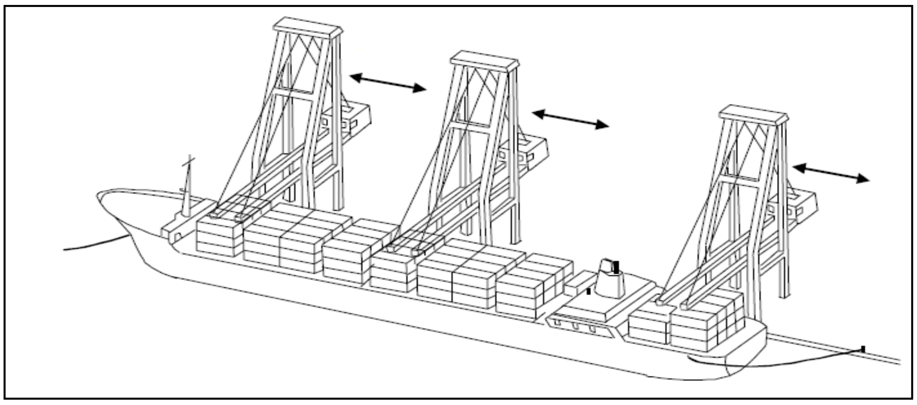


Figure 1.2: A representation of QCs working on a vessel (Kim and Park (2004)).

1.1 Container Terminals

In this section, some important concepts related with container terminal operations are introduced. Then a brief introduction to operational problems that exist in container terminals is made. Comprehensive surveys for these problems can be found in Vis and de Koster (2003), Steenken *et al* (2004), and Stahlbock and Voß (2008).

1.1.1 Container

A container is a box-shaped equipment which is used to transport goods from one destination to another. Standard sizes for containers are 20-foot (6.1 m) and 40-foot (12.2 m) and the standard measure of a container is called TEU, which is equivalent to 20-foot container. Total workload and capacity of the container vessels are often represented in TEUs. Most common container types of this size are 1-TEU dry (standard), 2-TEU dry, open top, refrigerated and flat rack. All of these containers are built by considering some international standards to make them interchangeable between different companies and carriers. This standardization helps to carry and handle containers easily by all means of transportation. Also containers are manufactured from very strong materials. For this reason, they can be stacked easily. A proper type of a container is the most secure alternative to transport goods without being damaged.

To sum up containers are widely used, because a container is;

- safe and enduring,
- can meet different kind of requirements,
- easy to handle,
- and easy to storage in stacks.

1.1.2 Container Terminal

Since first container vessel docked in 1956, containerization of cargoes is becoming ever more popular worldwide and almost all type of goods are now transported by containers. Containers supported a vast increase in overseas trading by allowing the effective transportation of goods over long distances. This intercontinental container traffic passes through a large number of container terminals all around the world.

Container terminals are huge facilities where containers are transshipped between different vehicles, for onward transportation. There are two types of container terminals; maritime container terminals and inland container terminals. Maritime container terminals are located at a seaside and provide connection between sea-freight and means of land transportation. Usually they are located around major harbors. On the other hand, inland container terminals are the connection points for different means of land transportation. They are usually located in territories with good rail connections to maritime container terminals.

In this thesis, from now on, a container terminal (CT) will refer to maritime container terminals. The most general representation of a CT can be found in Figure 1.4 below.

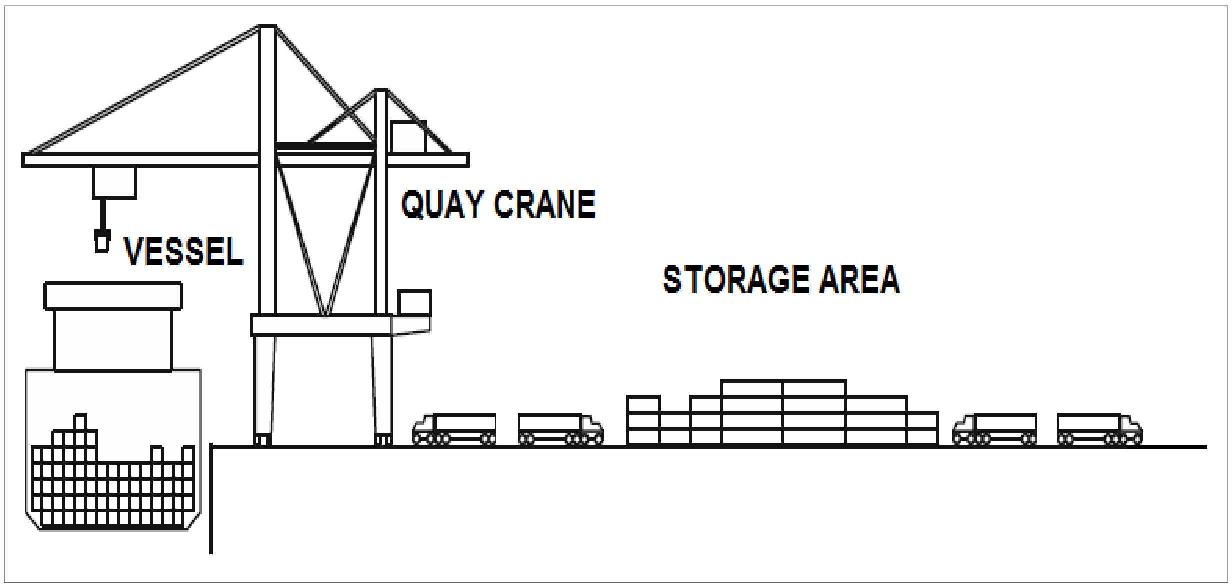


Figure 1.4: A general representation of a maritime container terminal.

As previously denoted, a CT connects sea-freight to land transportation. Then, any CT can be divided into two major areas based on operations performed; seaside and landside. At the seaside, container vessels dock to berths and their loading and unloading operations are completed by a set of quay cranes. At the landside, there is a storage area in which containers are temporarily stored in stacks. Trucks and/or automated guided vehicles (AGVs) transfer containers between the seaside and the storage area continuously. Moreover the containers which are coming from (or going to) land transportation are also transferred between storage area and gates of CT by trucks and/or AGVs.

1.1.3 Quay Crane

A quay crane (QC) is a huge equipment which is used to load and unload container vessels in container terminals (see Figure 1.5). There must be a number of QCs at the seaside of any maritime container terminal because it is the one and the only equipment to load (unload) containers to (from) vessels.

Therefore their workload is very intense and they often restrict the performance of whole container terminal. Having and operating dozens of QCs would be better for the performance of any container terminals, however, at the same time it is very costly to purchase and operate even a single QC. The purchasing cost of a QC is more than 5-million dollars. Therefore, there is an important trade-off for QCs: *"service speed vs. cost"*.

As a result, it is very crucial to operate QCs effectively in any container terminal.

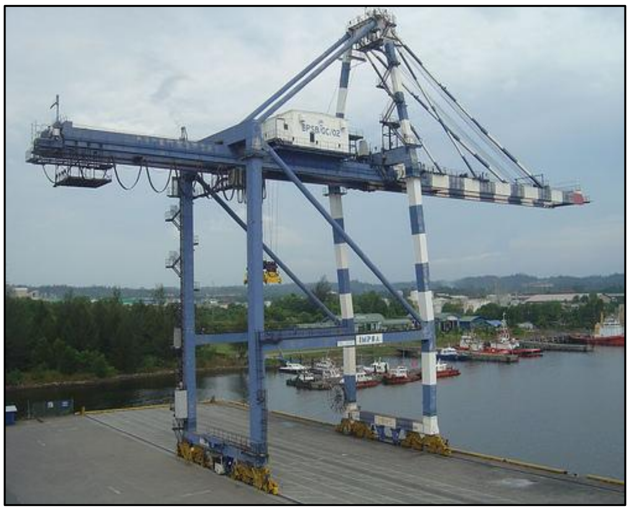


Figure 1.5: A quay crane.

1.2 Significance of CTs

Today, most of the cargoes are transported by sea-freight around the world (Ebeling (2009)). According to this, most of the CTs are facing with very intensive levels of workload. This intensity mainly centered at far-east countries because of the cheap labor. Even western companies tend to manufacture their products in areas where labor costs are relatively low, and then transport products to all around the world by container vessels because sea-freight is at least five-times cheaper than rail, truck and air freight. The workload of some major CTs in 2005 and 2010 are listed in Table 1.1 below.

Table 1.1: Workload of some CTs.

$Rank.$	$Terminal$	$2005\ (x1000\ TEUs)$	$2010\ (x1000\ TEUs)$	$Increase\%$
1	Shanghai	18,084	29,069	60.8
2	Singapore	23,192	28,431	22.6
3	Hong Kong	22,427	23,699	5.7
48	Ambarlı-İstanbul	1,446	2,540	75.7

These numbers alone do not indicate much; however, if these numbers are converted to daily workload in terms of approximate number of vessels, the intensity of the workload can be understood clearly. For example, based on 2010 data, the daily average of 333 container vessels were arrived to Hong Kong container terminal, and approximately 65,000 containers were loaded and unloaded every day. This huge amount of workload makes sense here for quay crane scheduling operations since every single container must be loaded and unloaded by QCs. Operations of QCs are a key factor for the productivity of a container terminal because it is the operation with the highest workload and is often causing a bottleneck for the performance of not only seaside operations, but also the whole container terminal.

Recently, the advantages of using sea-freight are started to be recognized in Turkey. Therefore, with an increasing demand to this transportation method, the need for new CTs and additional capacities for existing ones is also becoming a current issue in Turkey. As can be seen in Table 1.1, between 2005 and 2010 the largest increase of workloads among these four CTs is observed in Ambarli, İstanbul container terminal and this trend towards sea-freight is expected to continue.

1.3 Operations in Container Terminals

As previously mentioned, operations in container terminals can be divided into two as seaside and landside operations.

1.3.1 Seaside Operations

The major seaside operations consist of berth allocation, quay crane allocation (assignment) and quay crane scheduling. These three activities must be performed sequentially. Seaside operations for each vessel start with approaching of a vessel to CT and finish with the departure of this vessel. Also stowage planning is required to define the layout of containers on a vessel.

1.3.1.1 Berth Allocation

A berth position must be allocated for each vessel which is scheduled to arrive to a CT. Schedules of large ocean vessels can be known couple of months in advance, however CT managements made exact decision for their berth allocations closer to their arrival because lots of smaller-sized vessels also arrives with more immediate notifications of their schedules. Often berth allocation for a large ocean vessel begins two or three weeks before its arrival. The decision for berth allocation also requires a careful investigation of some parameters, for example, availability of different depths of berth positions, length and workload of vessels, priorities, etc. Also there can be different objectives for this problem; however, most of the time minimizing the average waiting times for arriving vessels and minimizing the total distance between all vessels on the berth is considered. The former is mainly for the benefit of vessels, while the latter is desired by CT practitioners for performing more effective QC operations.

Berth allocation problem (BAP) is often modeled as a two-dimensional bin-packing problem (Lim (1998)), while a bin is considered as a space-time graph. In this representation, each rectangle in the bin represents the berthed vessel where vertical and horizontal axes stand for its berth position and handling time, respectively. Of course, all rectangles in a bin must be non-overlapping. Lim (1998) also proves that BAP is NP-complete, and develops an effective problem specific heuristic. Nishimura *et al* (2001) study dynamic berth allocation problem and develop a genetic algorithm to solve this problem. Hansen *et al* (2008) solve the problem effectively with a variable neighborhood search. Comprehensive literature review and a classification scheme for BAP can be found in Bierwirth and Meisel (2010).

Since berth allocation is the first decision made by a CT for each arriving vessel, it affects the performance of other CT operations. Hence one can claim that the decision of berth allocation is strongly related with allocation of QCs (Bierwirth and Meisel (2010)).

1.3.1.2 Quay Crane Allocation

Based on the berthing plan of arriving vessels, quay cranes must be allocated to these vessels by considering that QCs operate on a same rail-track. The number of QCs to be allocated to a vessel mainly depends on the amount of total workload and the priority of each vessel. We have to choose between two planning alternatives while allocating QCs:

- QCs are allocated to a vessel to complete all of jobs on this vessel; that is, the number of QCs assigned to a vessel is the same during whole makespan. It would be easier for planning and operating QCs; however at the same time it causes QCs to have lower productivities and a decrease in overall performance of seaside operations (Bierwirth and Meisel (2010)).
- The number of QCs can change during the operation, which allows a terminal to reach higher overall performance. On the other hand, it is very complex for planning and operating QCs. It will lead CT management to solve more complex QC scheduling problems with ready times or time windows. Detailed explanations of these problems can be found in Sections 4.5 and 4.6.

On average two to six QCs operate at a single ocean vessel. Studies for the problem of allocating QCs (QCAP) are often integrated with BAP and the problem of scheduling of QCs (QCSP) because considering QCAP itself often gives impractical and inefficient output for overall performance of CTs. Integration of BAP and QCAP is studied by

Blazewicz *et al* (2011) and Park and Kim (2003), and of QCAP and QCSP is studied by Tavakkoli-Moghaddam *et al* (2009).

1.3.1.3 Quay Crane Scheduling

This operation consists of scheduling QCs to load and unload berthed vessels. The input for this operation is the plan of the workload of a vessel and the complete allocation information of QCs to this vessel. Detailed problem definition and literature review for this operation will be given in the next chapter.

1.3.1.4 Stowage Planning

A stowage plan of a vessel defines the location of containers over the vessel. Each location is defined as a triplet of <bay-row-tier>. Each container vessel is split into longitudinal parts which are termed as bays. Row is the position in which the container is placed across the width of the vessel. Tier indicates that at which level a container is placed, namely how high the container is stacked. Figure 1.6 depicts a representation of a stacking area of a vessel.

This plan is updated at each port the vessel visits. Both shipping lines and CT management involve in this planning process at different levels. Most of the input parameters to schedule QCs –precedence relationships, processing times, bay locations- are determined based on the stowage plan of the vessel.

1.3.2 Landside Operations

Some crucial operations are run continuously at the landside, as in the seaside. Landside of a CT consists of storage area, intermodal transportation vehicles, stacking equipment and gates which are providing the only connection of a CT with means of land transportation (trains and/or trucks). Some important operations in storage area and the means of intermodal transportation are described below.

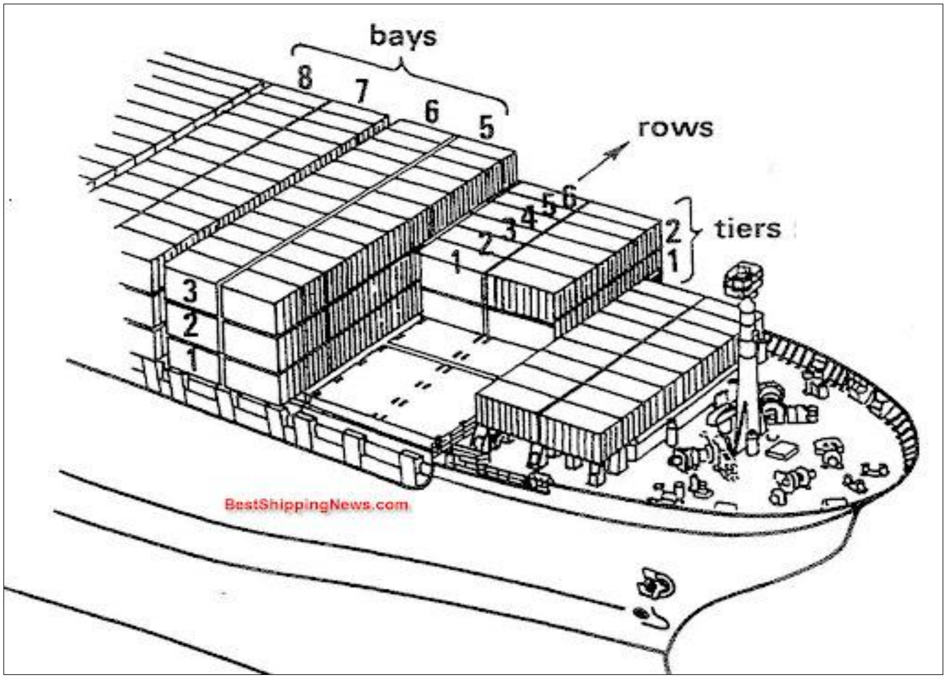


Figure 1.6: Representation of container positions on a vessel.

More information about stowage planning can be found in Wilson and Roach (2000).

1.3.2.1 Operations in Storage Area

Number of containers to be stacked in the storage area has increased accordingly with the volume of sea freight traffic. Almost every container needs to be stored in stacks for some amount of time because it is very hard to match arrival and departure times of different means of transportation to which a container is assigned. Accordingly, a storage area can be considered as a scarce resource and this fact makes the optimization of a storage area more important. For each container which needs to be temporarily stored in the CT, the terminal management has to select a storage position in terms of block, bay, row and tier with the same pattern as represented in Figure 1.6 above. Often there are different storage areas available for different types of containers. The distance of the storage position to the arrival and the departure points of a container is one of the key measures while allocating a storage position for a container. However, consideration of remarshalling and reshuffling of containers makes the problem harder.

Firstly, a number of reshufflings in the whole storage area need to be minimized. A reshuffling in a stack occurs when a specific container which is not directly accessible, needs to leave the stack. That is, to reach this specific container, all other containers which are placed on its top must be removed first and restacked again after removal.

The other important measure is the number of total moves, which is remarshalling. A remarshalling refers to a more general concept; every move of a container (removal, shifting, transfer and etc.) in a storage area called remarshalling. For an example move, a container can be transferred to other stacks to prevent potential reshufflings. Therefore it is also important to minimize the total number of moves of containers in a whole storage area. Even a single move of a container took some significant time; therefore, effective operations of storage area can reduce the overall handling time significantly.

Different storage allocation strategies are presented and discussed by Vis and de Koster (2004). Storage allocation can be considered as a dynamic operation, since also arrive after the start of the planning period. Accordingly, dynamic approaches are proposed by Kim *et al* (2000) and Kim and Park (2003).

1.3.2.2 Transfer operations

There are many different transfer and handling equipments used at the landside. Most common ones are AGVs, trucks and yard cranes.

Containers are transferred between QCs, storage area and gates of the terminal by AGVs and trucks. Automated systems are hard to operate and requires some considerable investment but offer more effective transfer operations. More information about operating AGV systems can be found in Qui *et al* (2002). Note that, the management of complex AGV systems can be considered as one of the trending subjects in container terminal optimization.

In storage area, stacking operations are carried out by yard cranes. Yard cranes for the same block operate on the same rail-track, therefore their scheduling operations requires a non-crossing constraint similar to scheduling of QCs.

1.4 Container Flows

Containers can also be classified based on their routes. First of all, an export container arrives to CT by land transportation and waits in the storage area until loaded to a vessel. On the other hand, import containers arrive by vessels and then depart the terminal by land transportation. Transit containers arrive by vessels, stacked in a storage area temporarily until loaded to another vessel. Each category of containers follows different operations flow in the terminal. General operations flow of different types of containers in container terminals represented in Figure 1.7.

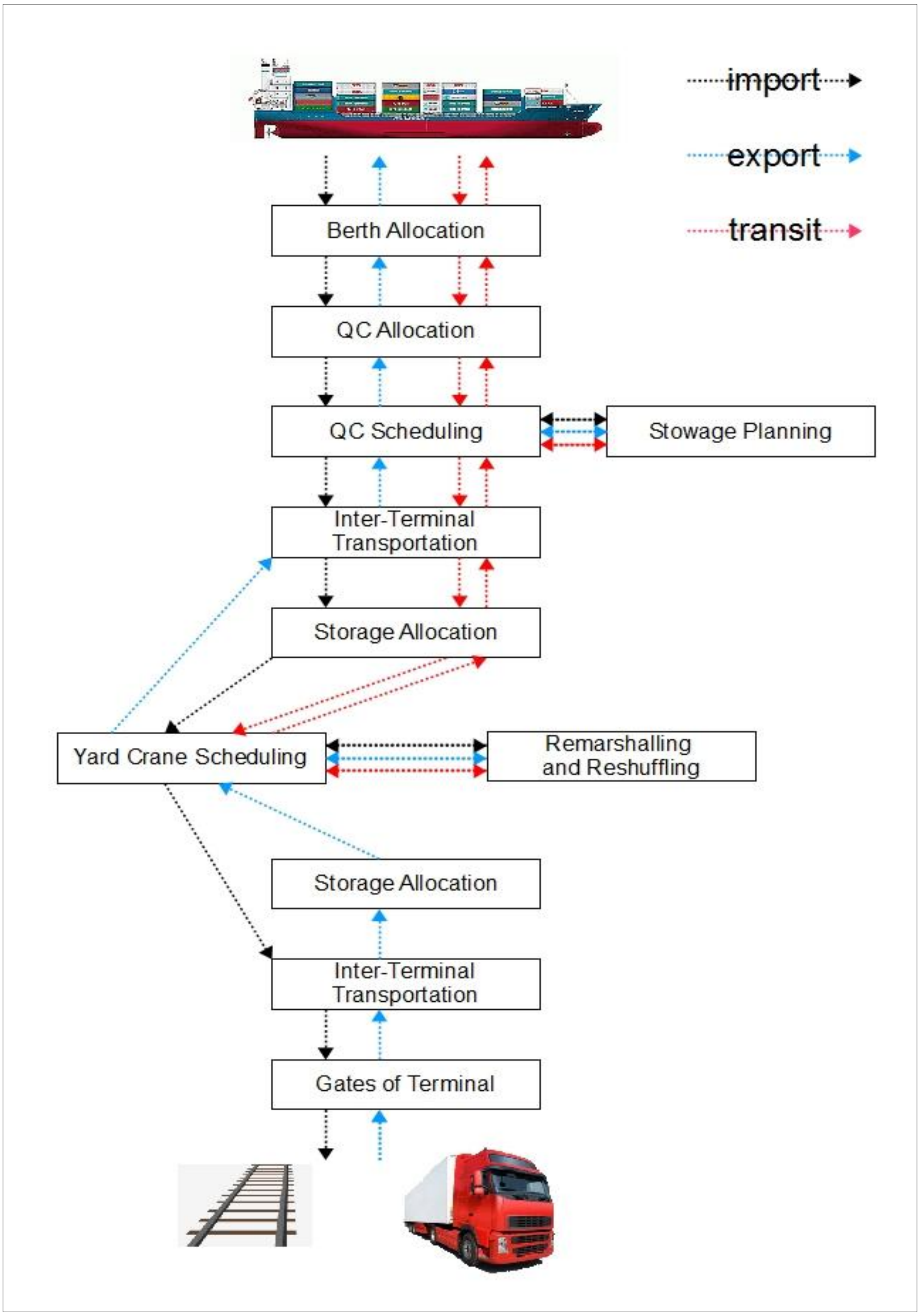


Figure 1.7: Container flows.

CHAPTER 2

THE QUAY CRANE SCHEDULING PROBLEM

2.1 Problem Definition

The quay crane scheduling problem (QCSP) is to find a schedule for the loading and the unloading of tasks of a single vessel by using a set of quay cranes with the aim of optimizing some objective function. In more detail, we are given a set of tasks $T = \{1,2,\ldots,nbT\}$, which are on a set of bays $B = \{1,2,\ldots,nbB\}$, and a set of assigned quay cranes $C = \{1,2,\ldots,nbC\}$, where nbT, nbB and nbC stand for number of tasks, bays and quay cranes, respectively. A very general representation of the QCSP of a single vessel with these parameters is shown below, in Figure 2.1.

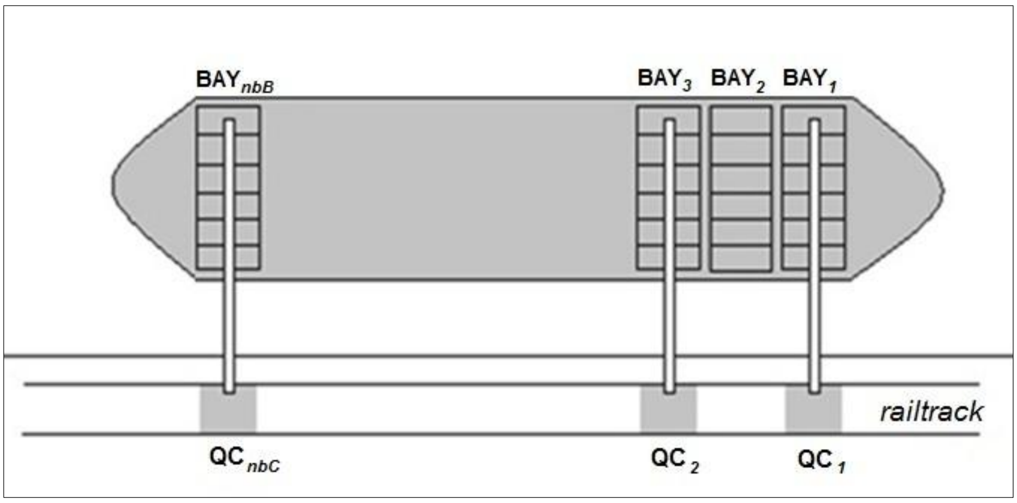


Figure 2.1: Positioning of QCs and bays.

Each task $i \in T$ must be performed by a single QC without preemption. QCs are operated on the same track; consequently, they cannot cross each other. Each task has a processing time p_i which represents the time required to complete task $i \in T$ by any crane. The problem is to find time-intervals in which the tasks are processed by the cranes with respect to a wide variety of problem constraints. Most of the time, the objective is to minimize the completion time of the latest completed job (makespan).

This problem with makespan minimization is NP-hard (Lim *et al* (2007), Lee *et al* (2008)).

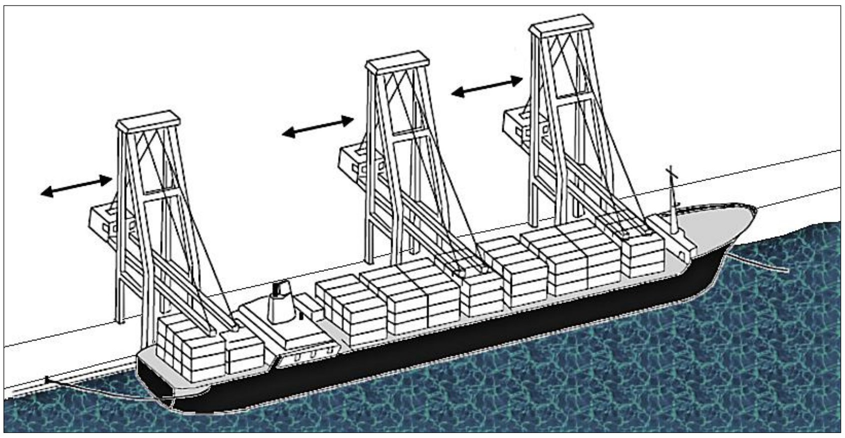


Figure 2.2: A drawing of QCs working on a vessel.

The more realistic representation of a general instance of the problem can be found in Figure 2.2. In this instance, nbB and nbC are defined as 10 and 3, respectively. However the number of tasks would be varied based on the definition of task.

The definition of a task divides the general QCSP into two major classes:

- QCSP with complete bays, in which a vessel is divided into parts longitudinally into bays. A single task consists of all unloading and loading tasks of a bay.
- QCSP with container groups, in which a task represents a group of containers that are stored in a bay and usually have a common destination.

To compare two major classes in the simplest way, in QCSP with complete bays, the maximum number of tasks equals the number of bays. However, in QCSP with container groups, there can be more than one container group in a single bay; hence, the number of tasks is not restricted and can be more than the number of bays. In the literature, these two classes are treated separately according to a survey conducted by Bierwirth and Meisel (2010).

In this thesis a task is defined as a container group and accordingly number of tasks per bay is more than one, more precisely, varies between 3 and 5. From now on, QCSP will refer to QCSP with container groups in rest the of the thesis.

Two important decisions must be made to solve this problem. First, each task must be assigned to one and only one quay crane (QC). The latter is to schedule these assigned tasks for each QC. From this perspective, the problem can be considered as a fundamental planning and scheduling problem which consists of assignment part as planning and then generating schedules based on this plan.

Furthermore, the roots of the QCSP come from the parallel machine scheduling problem (Guinet (1993)) where a QC is a machine and a container group (or a complete bay, based on the definition) is a task. Then each task must be processed with one of the machines which are identical and have capacity of one. However, QCSP is harder than the general parallel machine scheduling problem because of five additional constraints.

1. Non-crossing constraint: QCs are moving on the same rail-track, hence QCs cannot cross each other.
2. Non-interference: QCs must not interfere with each other, namely, at any time only one QC can work on a single bay.
3. Safety Distances: Other than non-interference constraint, at any time there should be a predefined space between two adjacent QCs to avoid some potential collusions on a rail track.
4. Precedence relationships among tasks based on a stowage plan (also may exist in parallel machine scheduling).
5. Travel Times: It takes some time for a QC to travel horizontally on a rail-track between bays. This constraint of the problem corresponds to sequence dependent setup times in parallel machine scheduling.

As a result, it can be easily stated that an instance of a QCSP with m machines and n tasks is harder than the same instance of a parallel machine scheduling problem with precedence relationships and sequence dependent setup times. Note that, solving the latter problem itself is even very hard (Lenstra *et al* (1977)).

2.2 Literature Review

As previously stated, the definition of task divides the general QCSP into two major classes:

a) QCSP with complete bays, in which a vessel is divided into parts longitudinally into bays. A single task consists of all unloading and loading tasks of a bay.
b) QCSP with container groups, in which a task represents a group of containers that are stored in a bay and usually have a common destination.

In this study, the latter is considered. In the existing literature these two classes are treated separately according to the survey of Bierwirth and Meisel (2010). In this survey, the authors investigate berth allocation problem and QCSP literatures comprehensively and they develop classification schemes for both problems.

The QCSP with complete bays is introduced by Daganzo (1989) and it is also the first QCSP article in the literature. In this study tasks can be preemptive; therefore one bay can be served by more than one crane. A mixed-integer programming model which considers more than one vessel is proposed. The objective is to minimize the weighted sum of the completion times of tasks. However they do not take even non-crossing of QCs into account and solve a problem which is very similar to parallel machine scheduling of preemptive tasks with identical machines. Peterkofsky and Daganzo (1990) solve the same mixed-integer programming model provided by Daganzo (1989) by developing a branch and bound method.

Lim *et al* (2004) state that non-crossing of QCs can be easily established for QCSP with complete bays by considering unidirectional schedules. In unidirectional schedules, all QCs have only one and the same moving direction. They develop an approximation algorithm to find feasible solutions. The objective is to minimize the makespan of the schedule.

Zhu and Lim (2005) prove that the problem with non-preemptive tasks is NP-hard. They formulate a mixed-integer programming model for QCSP with the objective of makespan minimization. They develop a branch and bound method to obtain optimal results. Also they apply a simulated annealing metaheuristic to QCSP to cope with larger instances. However, the only constraint that they take into account is non-crossing of QCs and this makes the problem much easier to solve.

Liu *et al* (2006) propose a mixed-integer-programming model for QCSP with complete bays by considering unidirectional schedules. The objective is to minimize the makespan of the schedule. The model considers moving speed and travel times of QCs, initial QC positions and safety margins between QCs; therefore their problem is much more complex than any other QCSP with complete bays study. However the last positions of the cranes are always greater than their initial positions because of the unidirectional schedules, hence all QCs must return to their initial position before any new operation. It is not practical since it requires a non-negligible amount of travel time for such operation.

In another study that deals with complete bays, Lim *et al* (2007) show that the optimal solution can be found by searching all unidirectional schedules. The authors also introduce that all unidirectional schedules can be obtained from QC-to-bay assignments. Based on this premise, they develop a constraint propagation method, simple approximation heuristics and a simulated annealing metaheuristic. Again, the only constraint that they take into account is non-crossing of QCs.

Lee *et al* (2008) prove that the QCSP with complete bays is NP-hard. They formulate a mixed-integer programming model which is derived from the model of Kim and Park (2004). The objective is to minimize the makespan of the schedule. They develop an efficient genetic algorithm to find near optimal solutions. In this study travel times of the QCs are assumed to be zero. Also there is no interference between QCs since only one QC serves a single bay. This situation holds for every QCSP with complete bays study with non-preemptive tasks.

Lee and Chen (2010) identify some important deficiencies that are mainly found in the models which are constructed based on the model of Kim and Park (2004). They develop a couple of approximation algorithms by using dynamic programming after resolving such deficiencies. Again, in that QCSP with complete bays study, they ignore travel times of QCs. Similar to studies which are listed above they do not consider precedence relations among tasks, safety margins (only exists in Liu *et al* (2006)) and QC interferences.

The QCSP with container groups, which is the most complex QCSP class among other classes, is formulated by Kim and Park (2004). In this problem, there can be more than one crane assigned to a single bay. Therefore there may be some interference among QCs and such interference must be avoided to obtain a feasible schedule. The authors develop a detailed mathematical model that covers a wide variety of problem constraints. The objective is to minimize the weighted sum of makespan of the schedule and the completion times of QCs. They propose a branch and bound method to solve small instances and a greedy randomized adaptive search procedure (GRASP) heuristic for larger instances.

The mathematical model of Kim and Park (2004) is improved by the stronger formulation of Moccia *et al* (2006). The authors also develop a branch and cut method to solve the problem. Sammarra *et al* (2007) identify some interference among QCs in Moccia *et al* (2006) and present a modified formulation. Then they solve the modified QCSP model by using tabu search metaheuristic. Recently, Lee and Chen (2010) identify some important deficiencies in container groups studies..

Bierwirth and Meisel (2009) investigate and fix the QC interference constraints and then develop a branch and bound based heuristic solution procedure for the problem. The objective is to minimize makespan of the schedule. The authors also show that the optimal schedules of QCSP with container groups do not have to be unidirectional schedules.

Meisel and Bierwirth (2011) introduce a unified approach to compare different quay crane scheduling problems. Moreover they provide a scheme for generating benchmark instances with certain characteristics.

Meisel (2011) considers that QCs have definite time windows. In this problem, QCs are only available in some certain time windows. The problem has a practical relevance because in container terminals QCs are frequently redeployed between vessels to provide faster service. For this version of QCSP with container groups, he formulates a mixed-integer programming model by using previously fixed QC interference constraints. The objective of the problem is makespan minimization of the schedule. He also constructs a tree-search based heuristic to solve large size QCSP with time windows instances.

Legato *et al* (2012) present a rich QCSP with numerous different properties of the problem as ready times, time windows, crane dependent processing times (with non-uniform QCs). They develop an independent-unidirectional heuristic called Timed-Petri-Net to solve these different types of problems.

Recently, Chung and Choy (2012) develop a genetic algorithm based on the model of Moccia *et al* (2006). The objective is to minimize makespan of the schedule. Although it is the latest QCSP paper published in the literature, neither QC interference deficiencies caused by travel times (Bierwirth and Meisel (2009)) nor other deficiencies identified by Lee and Chen (2010) are considered.

In Table 2.1 below, overview of qualities for all of these studies are listed. Note that, columns D1 and D2 shows the existence of corrections of modeling deficiencies identified by Bierwirth and Meisel (2009) and Lee and Chen (2010), respectively.

Table 2.1: Comparison of QCSP literature.

Study	*Container Groups*	*Complete Bays*	*Preemptive Tasks*	*Precedence*	*Travel times*	*Non - crossing*	*Safety margin*	*Ready times*	*Time Windows*	*D1*	*D2*
Daganzo(1989)		x	x								
Peterkovsky and Daganzo(1990)		x	x								
Kim and Park(2004)	x			x	x	x	x	x			
Lim et al (2004)		x			x	x					
Moccia et al(2006)	x			x	x	x	x	x			
Ng and Mak(2006)	x			x	x	x					
Liu et al (2006)		x			x	x	x	x			
Zhu and Lim(2006)		x				x					
Lim et al(2007)		x				x					
Samarra et al(2007)	x			x	x	x	x				
Lee et al(2008)		x				x					
Bierwirth and Meisel (2009)	x			x	x	x	x	x		x	
Lee and Chen(2010)		x				x					x
Meisel(2011)	x			x	x	x	x	x	x	x	
Meisel and Bierwirth(2011)				x	x	x	x			x	
Chung and Choy(2012)	x			x	x	x	x				
Legato et al (2012)	x			x	x	x	x	x	x	x	
This study	x			x	x	x	x	x	x	x	x

The last column of Table 2.1 should be treated carefully. Some of these methods, except that of Lee and Chen (2010) and our study, may be generating schedules which do not allow assigned QCs to pass the boundaries of the vessel, while the corresponding mathematical model that they represented as a base QCSP model in their studies are contrarily allowing such movements. Each of the methods represented in these studies should be carefully investigated to find out if they possess this deficiency.

2.3 Assumptions

The following assumptions are made for the QCSP:

- Every task must be completed by a single crane.
- Tasks are non-preemptive.
- QCs are identical and are operated on the same track.
- In a single bay, only one QC can work because of its size.
- The physical presence of idle QCs, if any, is not ignored. (Bierwirth and Meisel (2009) and Lee and Chen (2010) both identified that, in most of the previous QCSP models, ignoring idle cranes caused some deficiencies.) Idleness is a state of a QC which is waiting in some bay without operating on any task. Based on findings of the researchers, while a QC is idle, active QCs behave as if idle cranes do not exist on the rail track in most of the literature.
- Two QCs cannot be operated simultaneously in adjacent bays; therefore there must be one bay of safety margin between two adjacent QCs at any time.
- Travel time of a QC between two adjacent bays is one time unit.
- Allocated rail-track width must be equal to vessel size. This means that a QC assigned to a vessel cannot travel out of the boundary of this vessel. Boundaries can be defined as the rail-track area between leftmost (bay 1) and rightmost (bay nbB) bays of this vessel, including both. This assumption limits some task-to-QC assignments; for example, QC $nbC - 1$ cannot be assigned to a task on bay $nbB - 1$ to keep QC nbC within boundaries. If such an assignment is made, then QC nbC have to pass left side of the boundary because of the safety margin and non-crossing requirements.

- Initial positions of QCs are ignored in this study. Initial positions of QCs may be essential to generate unidirectional schedules, but in any non-unidirectional method, some good solutions may be restricted by using pre-definite initial positions. Since an optimal solution for an instance is directly dependent to initial positions of QCs, in this study, a starting position of a QC is set to be the bay in which it starts processing tasks in the schedule. That is, we are generating a schedule without defining initial positions, and then QCs are starting from the bay of the first scheduled task. In the literature, the QC schedules are evaluated somewhat non-realistically because travel times of QCs from last positions to pre-definite initial positions are completely ignored in a QC scheduling cycle (Figure 2.3).

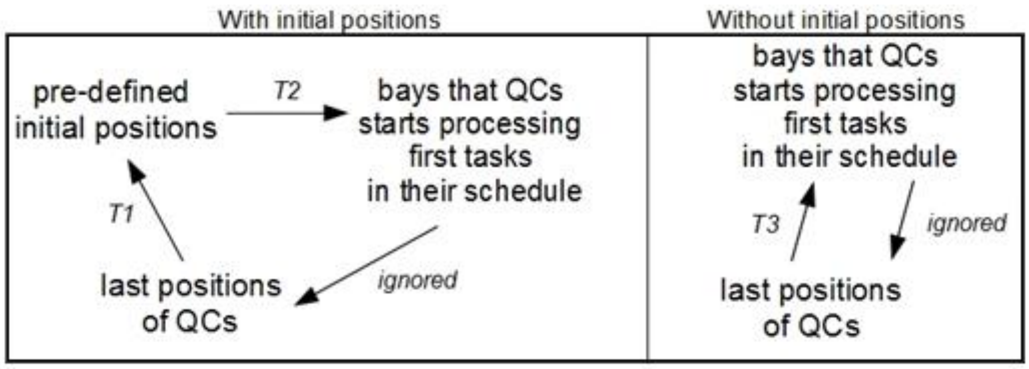

With initial positions
pre-defined initial positions
T2
bays that QCs starts processing first tasks in their schedule
T1
last positions of QCs
ignored
Without initial positions
bays that QCs starts processing first tasks in their schedule
T3
ignored
last positions of QCs

Figure 2.3: Initial positions.

What is more, if we use pre-definite initial positions, then we may have a different optimal solution for each permutation of these positions. However these optimal results are always greater than or equal to the optimal value of the same instance without pre-definite initial positions. Also there is no statistical or logical evidence which shows that $T3$ time units are longer than $T1 + T2$ or even $T2$ time units. As a result, we prefer $T3$ time units with optimal value z_1 rather than $T1 + T2$ with optimal value z_2, while $z_1 \leq z_2$ for each instance. As a result, $T3 + z_1 \leq T1 + T2 + z_2$ in a strong sense.

CHAPTER 3

METHODOLOGY

In this chapter the methodologies that we used in this thesis are briefly introduced. Constraint programming offers rich modeling language that supports using logical constraints directly in the model and this type of constraints is widely used in the constraint programming context for representing all kinds of problems. Accordingly, we have used propositional logic multiple times in our CP model.

3.1 Propositional Logic

Propositional logic (or simply, logic) is a system in which formulas of a formal language can be interpreted as propositions. The language of a propositional calculus consists of logical operators. The operators are used in this thesis are AND (∧), OR (∨), exclusive OR (⊻) and THEN (→). A proposition p can be "the weather will be rainy tomorrow", or "$x - 2y < 5$". A proposition in logic context can only take values of true or false.

Assume that p and q are different propositions. Then,

$p \vee q$ is true if at least one of the propositions is true,

$p \veebar q$ is true if exactly one of the propositions is true,

$p \wedge q$ is true if both of them are true,

$p \rightarrow q$ is true if p is true regardless of q's value or

is true if both of them are false.

3.2 Constraint Programming

Constraint programming (CP) is a technique that is used for representing and solving combinatorial optimization problems which are hard to solve. Similar to mathematical programming, CP is a combination of defining constraints about the problem via decision variables and finding a solution that satisfies all of the constraints. However, in CP, constraints are used actively to infer new constraints from the existing ones and to the reduce domains of variables by removing inconsistent values which are violating the constraints.

Constraint programming can be stated as a combination of three important components: modeling, filtering and search.

A CP model can be represented as an instance of a Constraint Satisfaction Problem (CSP) with the addition of concept of global constraints. In more details, a CSP is defined by a set of n decision variables X_i , $i \in \{1, .., n\}$ and corresponding set of domains D_i, $i \in \{1, .., n\}$, which defines the allowable values for these variables. There are also m constraints C_j $(X_1, \dots, X_n)$ $(j = 1, \dots, m)$ over these variables. Then the problem is to assign values for all variables from corresponding domains, while satisfying all constraints. The whole solution space for this problem (both feasible and infeasible) can be represented as $D_1 x \dots x D_n$. Most of the time this problem also involves an objective function $H(X_1, \dots, X_n)$ in CP context. A general CSP can be represented as below:

$$minimize \quad H(X_1, \dots, X_n)$$

$$subject\ to :$$

$$\begin{aligned} & C_j\ (X_1, \dots, X_n) && \forall j\ \in \{1, \dots, m\} \\ & X_i \in D_i && \forall i\ \in \{1, \dots, n\}. \end{aligned}$$

Then, in CP, this problem is solved by using the combination of filtering and search; simply, the domains of variables (or the search space) are reduced via constraint filtering and good solutions are searched within these reduced search space.

3.2.1 Modeling

The most important property of constraint programming paradigm is its modeling. In CP context, there are rich tools available for modeling. Therefore it allows representing the real-world concepts more directly (Lustig and Puget (2001)). A CP model is often considered independently from its solution strategy (propagation + search); that is, a specific model can be solved by very different approaches.

First of all, CP supports the same domain types (Discrete, Continuous and Boolean) with mathematical programming. Similarly the main scope is intensified over discrete and boolean variables. Differently from mathematical programming, CP offers two different types of decision variables; interval and sequence. Although these can be considered as compound decision variables which are constructed from discrete and boolean variables. They are unique and are allowing CP to have a very powerful and compact language. These new type of decision variables is introduced in Section 3.2.1.1.

CP supports most of the constraint types; linear, non-linear, logical, global and etc. By using well-known and simple constraint filtering algorithms, linear ($3x + y \leq 10$) and non-linear ($3x^2 + y^4 > 15$) constraints can be straightforwardly handled for reducing domains of variables. Ability to use logical constraints makes CP to be able to represent real-world constraints in a more declarative way. Assume that p and q are two binary (Boolean) decision variables. A constraint $p \rightarrow q$ ensures that, q can take value of "$True$" (or 1), if and only if p is also "$True$". As a result, the big-M technique is no more required. This fact simply allows CP users to be more independent and creative while building a model to represent real-world structures.

In CP context, there exists another important concept to represent the real-world structures: global constraints. They allow representing complex structures in a very direct way, most of the times with a single constraint. They also bring very effective constraint filtering and this can be stated as one of the most important properties of the constraint programming. Global constraints are discussed in Section 3.2.3.

3.2.1.1 Interval and Sequence Variables

The QCSP must be represented in terms of activities and resources to define some concepts which are widely used in the proposed CP model. In this representation, the tasks to be performed by quay cranes correspond to the activities and the quay cranes correspond to the resources. Then each activity A_i has a start time $start(A_i)$, an end time $end(A_i)$ and a processing time $process(A_i) = end(A_i) - start(A_i)$. Each QC is a unary resource, i.e., a QC can handle only one task at a time and this kind of resources and activities refers to a well-known disjunctive scheduling problem. In disjunctive scheduling, each resource can execute a single activity at a time and accordingly the activities which are assigned to the same resource shall not overlap. Therefore, if two different activities A_i and A_j are assigned to same unary resource, then:

$$start(A_i) + process(A_i) \leq end(A_j) \veebar start(A_j) + process(A_j) \leq end(A_i).$$

An interval variable (Laborie and Rogerie (2008)) is an interval of time during which an activity is executed. The decision here is to select when to execute this activity during the planning horizon. Each interval variable is characterized not only by a start time, but also with an end time and a processing time and the presence information, $presence(A_i)$. An interval variable can be optional; therefore, the activity can be left unexecuted. In these situations, unexecuted activity is represented by $presence(A_i) = False\ (or\ 0)$. On the other hand, all executed optional interval variables are represented by $presence(A_i) = True\ (or\ 1)$. This property of an interval variable helps to model when there are activities that can be executed on a set of alternative resources. In this study, a task can be executed on a set of QCs; therefore, optional interval variables are used to model this case.

A sequence variable (Laborie *et al* (2009)) is a decision variable for which the value is a permutation of some group of variables. Sequence variables can also keep transition times between interval variables. We can simply construct the same model without using any sequence variable; however, their presence brings more effective constraint filtering. Hence sequence variables are also preferred to implement the travel times of the QCs in this study.

3.2.1.2 Global Constraints

A global constraint in CP can represent the complex relationships between the problem variables as a single constraint. Usually it captures most of the problem variables, therefore provides faster and more effective domain reduction by using specialized filtering algorithms. The most widely known global constraint is $alldifferent(\forall v_i)$, where $v_i, i \in \{1, \dots, n\}$, is a decision variable with $domain(v_i) = [d_1, d_n]$. It ensures that each variable v_i must take a different value from its domain. In contrast, $n(n-1)/2$ constraints are required to represent such a relationship in a mixed-integer programming model.

There are many different global constraints in the literature and two of them are considered in this study. The $alternative(\forall A_i, Y_{ij})$ constraint (Beck and Fox (1999)) simply assigns each activity to a single resource. The property of being optional of the interval variables makes sense here. Assume that there are m resources in the problem and interval variable $Y_{ij'}$ indicates that activity i is assigned to a resource j'. Since each activity is assigned to only one resource, Y_{ij} variables are not taken into account when $j \neq j'$.

Furthermore a $disjunctive(\forall A_i | A_i \in P)$ global constraint ensures that the activities which are elements of some set P, should not overlap. That is,

$$start(A_i) + p(A_i) \leq end(A_k) \vee start(A_k) + p(A_k) \leq end(A_i) \quad \forall A_i, A_k \in P, i \neq k.$$

In other words, a disjunctive global constraint refers to disjunctive scheduling in which activities performed on a unary resource. On the other hand, activities can be performed on a resource that can process more than one activity at any moment. In CP context, this type of resources denoted by $cumulative\,(\,\forall A_i, C | A_i \in P\,)$ global constraint where C is the maximum capacity. Therefore, this cumulative constraint stands for a resource with capacity C and the number of activities performed on this resource must be less than or equal to C at any time. If $C = 1$, then this global constraint is simply reduced to a $disjunctive$ constraint.

There are numerous global constraints to represent the real-world structures other than presented in this section. More information can be found in “global constraint catalog” of Beldiceanu $et\ al$ (2010), which is the most comprehensive study over the global constraints.

3.2.2 Constraint Filtering

Constraint filtering is a systematic way to reduce the domains of decision variables by eliminating the inconsistencies in the model. This important process is accomplished by different algorithms.

The most general constraint filtering algorithm type is arc-consistency. Arc-consistency algorithms are mainly executed for all constraints except the global constraints. The most well-known algorithm of this type is AC-3 (Bessiere *et al* (2005)), which investigates all values in the domain of each variable, and removes the inconsistent ones. Note that, a value-variable assignment is *inconsistent* if at least one constraint is not satisfied.

A simple example can be given as follows: assume that there are three variables A, B and C with the same domain of $\{1,2,3,4,5\}$. Also there exist two constraints involving these variables: $A > B$ and $B = C + 1$. By a simple inference, a new constraint $A > C + 1$ can also be obtained. If every value of each variable is checked for these three constraints, some inconsistent values can be straightforwardly removed from their domains. As a result, initial domains are reduced to $A \in \{2,3,4,5\}$, $B \in \{2,3,4\}, C \in \{1,2,3\}$ by constraint filtering, which leads to the reduction of the number of possible assignments for these three variables from 125 to 36.

Additionally, global constraints have many specialized filtering algorithms. One can refer to the studies by Régin (1994), and Lopez-Ortiz *et al* (2003) for *alldifferent* constraint and by Baptiste and Le Pape (1996) and Vilim (2004) for $disjunctive$ constraint. These kinds of filtering algorithms usually offer more effective domain reductions by considering more than one relationship at once. Therefore in the constraint programming context it is really important to represent the model with global constraints as much as possible.

3.2.3 Search

Constraint filtering alone is not sufficient to find even a single feasible solution, unless all domains are reduced to a singleton value, which is almost impossible for any combinatorial problem. Thus, we need a search phase within reduced domains to find feasible solutions. The most common, yet the simplest systematic approach to be used in a search phase, is a tree-based constructive search called backtracking (Dechter *et al* (1998)).

Backtracking is a search method that incrementally attempts to extend a partial solution by assigning values to variables one by one toward a complete solution. The feasibility of each assignment is checked via constraint filtering to make domains consistent with the current partial solution. If a selection is inconsistent with any of the problem constraints, the search backtracks to an upper node and tries another selection.

We can impose a strategy that defines which variables and values to be fixed first, hence by applying different selection strategies the search can be completely directed.

The literature presents more complex constraint-based search methods (Focacci *et al* (2002)). One of these powerful methods is Large Neighborhood Search (LNS) (Shaw (1998)) in which a tree-based search (backtracking) is used with constraint programming to evaluate the cost and the feasibility of the move in the local search. Generally, this procedure is based on a process of continuous relaxation and re-optimization; an initial solution is constructed first and then improved iteratively until some stopping condition is satisfied (see Fig. 7).

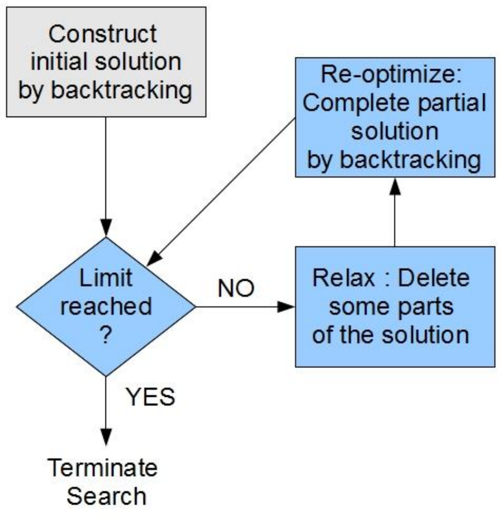

Construct initial solution by backtracking
Re-optimize: Complete partial solution by backtracking
Limit reached ?
NO
Relax : Delete some parts of the solution
YES
Terminate Search

Figure 7: General framework for LNS.

An iteration of relaxing a complete solution and re-optimizing the partial solution can be considered as the examination of a powerful neighborhood move and therefore the farthest parts of the search space can be reached by a single move. The process of re-optimization can use the full power of the constraint programming via constraint filtering and backtracking.

A different version of LNS is represented in Godard *et al* (1999) which generates initial solutions and re-optimizes partial solutions by using the setTimes algorithm of Le Pape (1994).

The general setTimes algorithm is presented above.

1. Let S to be the set of selectable (non-fixed) activities.
2. If all activities are fixed then exit with a solution. Otherwise go to step 3.
3. If set S is empty go to step 4. If set S is not empty:

   a) Select an activity from S which has the minimum earliest start time, that is, sort lowest elements of each domain increasingly and select the activity corresponding to first element of this sorted list. If there is a tie, select the one with smaller domain size (common case). If tie is not broken, then select one of them randomly.
   b) Create a choice point to allow backtracking and fix the start time of selected activity to the lowest element of its domain. Turn back to step 2.
4. Set S is empty, then backtrack to most recent choice point. Upon backtracking, mark the activity that was scheduled at the considered choice point as "not selectable" as long as lowest element in its domain has not changed. This prevents the algorithm to get stuck with same solutions by selecting different activities. Turn back to step 2.

By applying setTimes algorithm once, starting times of all tasks are fixed and a complete schedule is constructed. Godard *et al* (1999) also extend LNS by improving its relaxation part because they argue that the relaxation scheme of Shaw (1998) lacks of flexibility for scheduling problems. Accordingly, they relax an initial solution into resource temporal network of current solution via partial order schedules (Laborie (2003)), and then delete some parts of the solution in a randomized manner. By default, ILOG's CP Optimizer 12.3 uses the Self-Adaptive LNS approach of Laborie and Godard (2007), which simply extends the work of Godard *et al* (1999) by adding a learning scheme. Learning is a key factor in the robustness of the approach because it helps to converge on the most efficient neighborhoods and completion strategies. Note that, in this thesis we have used this version of LNS.

CHAPTER 4

MODELING AND SOLVING QCSP

The comprehensive constraint programming model for QCSP which consist of numerous properties of the problem is proposed in this section. Moreover, common deficiencies found in QCSP literature by Lee *et al* (2010), Bierwirth and Meisel (2009) and their corrections are introduced. The corrected version of MIP model is represented just before individually representing the functions of the constraints of the proposed CP model. Also two different extensions of the QCSP are also presented in this chapter.

For applicability of any QCSP study to real-world problems, constraints of the problem should be reflected properly into the model. Therefore, the main motivation in this study is to develop a model that not only works fast and is flexible for all sizes and types of practical instances, but also is highly applicable in real-world container terminals.

The aim during the modeling process is initially determined based on the methodology which is used to model and solve QCSP. Hence, constraint programming modeling context is widely investigated. Using effective global constraints instead of complex model structures whenever possible will be a key to success for any CP model, at least theoretically. Also keeping number of decision variables at minimum may significantly affect the solution speed and quality since CP is a technique that mainly works by reducing the domain of each decision variable. As a result, this CP model for QCSP is constructed based on these premises. To sum up, it is aimed to use as much as global constraints to model problem concepts while keeping the number of decision variables used to represent the whole model at minimum.

4.1 Common Deficiencies in QCSP Literature

The feasible solutions provided by previous QCSP studies may result in inappropriate and non-applicable schedules for the real-world problems. Hence this may cause some important consequences, such as crane interference and different kind of inefficiencies. In this section, common deficiencies that are found in QCSP literature are briefly discussed. Three important deficiencies of previous QCSP models are identified and then treated in two recent articles, Lee and Chen (2010) and Bierwirth and Meisel (2009). Although their problem classes are different, these deficiencies are related with both problem classes.

The first deficiency is caused by the travel times of QCs. Travel times of QCs between bays are supposed to be positive in most of the QCSP with container groups studies. Accordingly, the movements of QCs must be carefully reinvestigated to completely avoid potential crane interferences. Bierwirth and Meisel (2009) note that, despite the fact that there were some treatments proposed in Moccia *et al* (2006) and Sammarra *et al* (2007), some significant QC interferences are still exist on the feasible schedules based on their movements along the rail-track. They show that QCs may collapse or their safety margins may be violated in the schedules that are generated by previous models. This deficiency and its treatment are introduced in Section 4.5.10 and can be found in Bierwirth and Meisel (2009) in more details.

Secondly, Lee and Chen (2010) and Bierwirth and Meisel (2009) both identified similar deficiencies in two independent studies. They show that simultaneous positions of QCs may be disrespected in previous studies, especially when some of the cranes are idle during the makespan. Most of the time, other QCs behave as if these idle QCs do not exist on the rail-track in previous QCSP models. To get rid of this deficiency, there must be enough distance between any two non-adjacent QCs at any time to accommodate in-between QC(s) safely. QCs are huge equipment and simultaneous positions of QCs must be respected. This deficiency and its treatment are also introduced in more details in Section 4.5.12.

The last deficiency is identified by Lee and Chen (2010) and it is related with the effective usage of the berthing area. All QCs are operated on the same rail-track, therefore the effective allocation of the rail-track area is needed. In most of the real-world cases, the workload of container terminals is very intense; so that berth allocations for vessels can be very close. See the Figure 4.0 below for an example of such a situation.

Figure 4.0: Berthed container vessels

By this deficiency, they identified that if QCs which are allocated to complete loading and unloading tasks of a definite vessel, pass the boundary of this vessel, then these QCs enter other areas of rail-track, which are probably required to accomplish loading and unloading operations of other vessels. It is not practically effective even if it is not infeasible. To make the study more applicable in real-world problems, rail-track areas must be allocated to berthed vessels, and these areas must be used effectively by QCs. In this study, this fact is also taken into account by restricting the movements of QCs. This deficiency and its treatment are also investigated in Section 4.5.6.

4.2 MIP Model of QCSP

In this section the mixed-integer programming model proposed by Bierwirth and Meisel (2009) is introduced. The researchers build this model by taking the MIP model of Kim and Park (2004) and the improvements of Moccia *et al* (2006) and Sammarra *et al* (2007) as a basis. In this thesis, model of Bierwirth and Meisel (2009) is also extended with the correction of the deficiency noted by Lee and Chen (2010) for QCSP with container groups.

Sets and Parameters

q number of QCs assigned to the vessel;

n number of tasks of the vessel;

b number of bays of the vessel;

Ω set of tasks, $\Omega = \{1,2,\ldots n\}$;

Q set of QCs, $Q = \{1,2,\ldots,q\}$;

p_i processing time of task $i, \forall i \in \bar{\Omega}$;

l_i location (bay) of task $i, \forall i \in \bar{\Omega}$;

t travel time of a QC between two adjacent bays ($t_{0j} = t_{Tj} = 0$).

Dummy tasks 0 and $T = n + 1$ with processing times $p_0 = p_T = 0$ are also added to be able to model initial positions, then $\Omega^0 = \Omega \cup \{0\}$, $\Omega^T = \Omega \cup \{T\}$, $\bar{\Omega} = \Omega \cup \{0, T\}$;

ϕ set of precedence constrained task pairs;

δ number of in-between bays required to keep between adjacent vessels, namely safety margin, $\delta = 1$;

ψ set of all task pairs for which it is known in advance that they cannot be processed simultaneously, $\psi \supseteq \phi$. This set contains task pairs which are located on adjacent bays as well as precedence constrained task pairs. It also contains the fact that two QCs cannot operate at the same bay at the same time.

r_k ready time of crane $k, \forall k \in Q$;

L_k initial position of crane k, then $t_{0jk} = t|L_k\text{-}l_j|, \forall i \in \bar{\Omega}, \forall k \in Q$;

$$\Delta_{ijvw} = \begin{cases} \left(l_j - l_i - (1+\delta)(w-v)\right)t, & if\ (w > v), \left(l_j < l_i + (1+s)(w-v)\right) \\ \left(l_j - l_i + (1+\delta)(v-w)\right)t, & if\ (w < v), \left(l_j > l_i - (1+s)(v-w)\right) \\ 0 & otherwise; \end{cases}$$

$\theta = \{(i,j,v,w) \in \Omega^2 x Q^2 \,|\, (i<j) \wedge \left(\Delta_{ijvw} > 0\right)\}$;

M a very large number.

Decision Varibles

c_i	completion time of task i, $\forall i \in \bar{\Omega}$;
x_{ijk}	binary decision variable that takes value
	1 if tasks i and j are processed consecutively by crane k, $\forall k \in Q, \forall i, j \in \bar{\Omega}$,
	0 otherwise;
z_{ij}	binary decision variable that takes value
	1 if task j starts after the completion of task i, $\forall i, j \in \bar{\Omega}$,
	0 otherwise.

MIP Model

$$minimize\ c_T \qquad (M1)$$

$Subject\ to$:

$$\sum_{j \in \Omega^T} x_{0jk} = 1 \qquad (\forall k \in Q) \ (M2)$$

$$\sum_{j \in \Omega^0} x_{jTk} = 1 \qquad (\forall k \in Q) \ (M3)$$

$$\sum_{k \in Q} \sum_{j \in \Omega^T} x_{ijk} = 1 \qquad (\forall i \in \Omega) \ (M4)$$

$$\sum_{j \in \Omega^0} x_{jik} - \sum_{j \in \Omega^T} x_{ijk} = 0 \qquad (\forall i \in \Omega, \forall k \in Q) \ (M5)$$

$$c_i + t_{ij} + p_j - c_j \le M(1 - x_{ijk}) \qquad (\forall i, j \in \bar{\Omega}, \forall k \in Q) \ (M6)$$

$$c_i + p_j - c_j \le 0 \qquad (\forall (i,j) \in \phi) \ (M7)$$

$$c_i + p_j - c_j \le M(1 - z_{ij}) \qquad (\forall i, j \in \Omega) \ (M8)$$

$$c_j - p_j - c_i \le M z_{ij} \qquad (\forall i, j \in \Omega) \ (M9)$$

$$z_{ij} - z_{ji} = 0 \qquad (\forall (i,j) \in \psi)(M10)$$

$$\sum_{u\in\Omega^0} x_{uiv} + \sum_{u\in\Omega^0} x_{ujw} \le 1 + z_{ij} + z_{ji} \qquad (\forall(i,j,v,w) \in \theta)(M11)$$

$$c_i + \Delta_{ijvw} + p_j - c_j \le M(3 - z_{ij} - \sum_{u\in\Omega^0} x_{uiv} - \sum_{u\in\Omega^0} x_{ujw}) \qquad (\forall(i,j,v,w) \in \theta)\ (M12)$$

$$c_j + \Delta_{ijvw} + p_j - c_i \le M(3 - z_{ji} - \sum_{u\in\Omega^0} x_{uiv} - \sum_{u\in\Omega^0} x_{ujw}) \qquad (\forall(i,j,v,w) \in \theta)\ (M13)$$

$$r_k + t_{0jk} + p_j - c_j \le M(1 - x_{0jk}) \qquad (\forall j \in \Omega, Vk \in Q)(M14)$$

$$c_i \ge 0 \qquad (\forall i \in \bar{\Omega})(M15)$$

$$x_{ijk} \in \{0,1\} \qquad (\forall i,j \in \bar{\Omega}, \forall k \in Q)\ (M16)$$

$$z_{ij} \in \{0,1\} \qquad (\forall i,j \in Q)\ (M17)$$

$$x_{ijk} = 0 \qquad (\forall i,j \in \bar{\Omega}, \forall k \in Q: (l_i > b - 2(q-k)) \vee (l_i < 2k-1))\ (M18)$$

$$x_{ijk} = 0 \qquad (\forall i,j \in \bar{\Omega}, \forall k \in Q: \left(l_j > b - 2(q-k)\right) \vee \left(l_j < 2k-1\right))\ (M19)$$

The objective is to find the smallest completion time of the latest task and is denoted by (M1). The paths followed by QCs to complete tasks of the vessel are defined by constraint sets (M2) to (M6). Precedence relationships among tasks are defined by constraint set (M7). The variables z_{ij} are defined in constraint sets (M8) and (M9). In order to express a safety margin of one bay, Sammarra $et\ al$ (2007) include those pairs of tasks in set Ψ that belong to adjacent bays. Constraints set (M10) ensure that these tasks are not processed simultaneously. In constraint set (M11), the assignments of tasks to QCs that are realized in the schedule are identified. Here, $\sum_{u\in\Omega^0} x_{uiv} = 1$ if task i is processed by QC v, and $\sum_{u\in\Omega^0} x_{ujw} = 1$ if task j is processed by QC w. If both assignments take place, the left-hand 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 $z_{ij} = 1$, constraint set (M12) insert the minimum temporal distances calculated by Δ_{ijvw} between the completion time of task i and the starting time of task j. The corresponding case of $z_{ji} = 1$ is handled by constraint set (M13). Initial positions of QCs are defined by constraint set (M14). In this thesis, the deficiency is identified by Lee and Chen (2010) is also corrected for QCSP with container groups. Therefore constraint sets

(M18) and (M19) are added to MIP model to forbid some movements of assigned QCs to keep them within boundaries of the vessel during the makespan.

4.3 Constraint Programming Model

In this section, a new constraint programming model to QCSP is proposed.

Sets and Parameters

nbT	number of tasks of the vessel to be processed;
nbB	number of bays in the vessel;
nbC	number of quay cranes assigned to the vessel;
T	set of tasks, $T = \{1, \dots, nbT\}$;
B	set of bays, $B = \{1, \dots, nbB\}$;
C	set of quay cranes, $C = \{1, \dots, nbC\}$;
p_i	processing time of task $i, \forall i \in T$. Each quay crane is identical; therefore processing time for task i is same for all quay cranes;
$Prec$	set of precedence among tasks acquired from the stowage plan of a vessel, i.e., $Prec = \{< i, k >, i, k \in T\}$;
l_i	location (bay) of a task $i, \forall i \in T$;
s	the minimum required number of bays between two adjacent QCs at any time; namely safety margin;
TP	sum of processing times of all tasks, i.e., $\sum_{i \in T} p_i$;
S_b	set of tasks at bay b, i.e., $S_b = \{\forall i \in T: l_i = b\} \quad \forall b \in B$;
$early_i$	earliest starting time for task i; that is, total workload of predecessors of task i, i.e., $early_i = \sum_{<k,i> \in Prec} p_k \quad \forall i \in T$.

Decision Variables

X_i	an interval variable that represents the interval in which task i is processed (by any quay crane), $domain(X_i) = [early_i, TP], \forall i \in T$;

Y_{ij} an optional interval variable that represents the interval in which task i is processed by quay crane j, $domain(Y_{ij}) = [early_i, TP]$, $\forall i \in T, \forall j \in C$;

Z an integer variable that defines the makespan, i.e. the maximum completion time of all tasks by quay cranes, $domain(Z) = [LB, TP]$, where LB is a lower bound on the optimum value of the makespan. The procedure to generate lower bound LB is presented in Section 4.4.

CP Model

The constraint programming model is formulated as follows:

$minimize\ Z$

Subject to:

$$Z = max_{\forall i \in T, \forall j \in C}\ (end(Y_{ij})) \tag{1}$$

$$alternative(X_i, (Y_{ij} \mid \forall j \in C)) \qquad \forall i \in T \tag{2}$$

$$disjunctive(Y_{ij} \mid \forall i \in T) \qquad \forall j \in C \tag{3}$$

$$disjunctive(\ Y_{ij} \mid \forall i \in S_b, \forall j \in C) \qquad \forall b \in B \tag{4}$$

$$disjunctive(\ Y_{ij}, Y_{nm} \mid \forall i \in S_b, \forall n \in S_{b+1}\ \forall j, m \in C\ ) \qquad \forall b \in\ B/\{nbB\} \tag{5}$$

$$(l_i > nbB - 2(nbC - j)) \vee (l_i < 2j - 1) \rightarrow presence(Y_{ij}) = 0 \qquad \forall i \in T, \forall j \in C \tag{6}$$

$$disjunctive(Y_{ij}, Y_{nm}\ )$$
$$\forall i, n \in T, \forall j, m \in C, l_i \geq j, (j \neq nbC), (l_i \neq nbB), (l_n \geq l_i), (m \geq j), (l_n < l_i + (1+s)(m-j-1)+2) \tag{7}$$

$$end(Y_{ij}) \leq start(Y_{kr}) \qquad \forall < i, k > \in Prec, \forall j, r \in C, (i \neq k) \tag{8}$$

$$presence(Y_{ij}) \wedge presence(Y_{nm}) \rightarrow$$
$$\left(end(Y_{nm}) + l_n - l_i - (1+s)(m-j) \leq start(Y_{ij})\right) \veebar \left((end(Y_{ij}) + l_n - l_i - (1+s)(m-j) \leq\ start(Y_{nm})\right)$$
$$\forall i, n \in T, \forall j, m \in C, (m > j), (l_n < l_i + (1+s)(m-j)) \tag{9}$$

$$presence(Y_{ij})\ \wedge presence(Y_{nm}) \rightarrow$$

$$\left(end(Y_{nm}) + l_n - l_i + (1+s)(j-m) \leq start(Y_{ij}) \right) \veebar \left((end(Y_{ij}) + l_n - l_i + (1+s)(j-m) \leq start(Y_{nm}) \right)$$
$$\forall i, n \in T, \forall j, m \in C, (m < j), (l_n > l_i - (1+s)(j-m)) \qquad (10)$$

$$presence(Y_{ij}) \wedge presence(Y_{nj}) \rightarrow$$
$$\left(end(Y_{nj}) + |l_i - l_n| \leq start(Y_{ij}) \right) \veebar \left(end(Y_{ij}) + |l_i - l_n| \leq start(Y_{nj}) \right)$$
$$\forall i, n \in T, \forall j, m \in C, (i \neq n) \qquad (11)$$

The objective function is to minimize the completion time of the latest QC which is calculated by constraint (1). Constraint set (2) is a global constraint to assign each task to one and only one QC. The following two constraints are global constraints to forbid definite tasks to overlap. Global constraint set (3) ensures that tasks which are assigned to the same QC will not overlap. Also global constraint set (4) avoids the interference among QCs by not allowing the overlap of the tasks which are located in the same bay. Constraint set (5) ensures that two tasks that are located in adjacent bays cannot be processed simultaneously; that is, the safety margin is assumed to be one bay. Constraint sets (6) and (7) resolve two deficiencies that are identified by Lee and Chen (2010). Constraint set (6) is defined to keep QCs within the boundaries of the vessel. Constraint set (7) ensures that there will always be enough space between two cranes to accommodate in-between cranes, even if they are idle. Precedence relations among tasks are defined by constraint set (8). Constraint sets (9) and (10) together ensure that QCs cannot cross each other since they are operated on the same track. Travel times of QCs are implemented in constraint set (11). By using some logical operators directly, a two-indexed decision variable is sufficient to represent this feature. Moreover the correct treatment of travel times and safety margins proposed by Bierwirth and Meisel (2009) is embedded in (9) and (10).

A complete model can be constructed by using only these decision variables above; however, a set of sequence decision variables is added to the model to strengthen the inference among problem elements.

S'_j	a sequence variable that keeps the permutation of the tasks to be processed by QC j, $domain(S'_j) = \{permutation\ of\ Y_{ij} \mid i \in T\ and\ presence(Y_{ij}) = 1\}$,$\forall j \in C$, with transition distance matrix $T''_{in} = \lvert li - ln \rvert$, $\forall i, n \in T$. This matrix keeps the minimum time required between Y_{ij} and Y_{nj} if they are executed sequentially.

S_{ijnm} a sequence variable that keeps the relative order of tasks i and n processed by QCs j and m, respectively. $domain(S_{ijnm}) = \{order\ of\ Y_{ij}\ and\ Y_{nm}\}$, with transition distance matrix $T'_{ijnm} = |l_i - l_n| + (1+s)(m-j),\ \forall i,n \in T, \forall j,m \in C$.

By default, a sequence variable orders the set of decision variables according to their starting times; however, if it is used within a *disjunctive* global constraint, this ordered set of variables are also forced to be non-overlapping. These sequence variables are auxiliary in the model. Hence they will be active in the model only when related conditions occurred.

For more effective filtering and to break the symmetry, constraint sets (9) and (10) are replaced by a new constraint set which is constructed with S_{ijnm} sequence variables:

$$disjunctive(S_{ijnm}, T') \qquad \forall i,n \in T,\ \forall j,m \in C, m \geq j, l_n \leq l_i + (1+s)(m-j) \quad (12)$$

Also constraint set (11) is replaced by a new constraint set (13) for faster reductions of domains. This new travel time constraint set consists of S'_j and keeps the transition distances among the locations of couple of tasks which are sequentially processed by QC j.

$$disjunctive(S'_j, T'') \qquad \forall j \in C \qquad (13)$$

Since constraint set (13) also ensures that tasks which are processed by the same QC should not overlap, constraint set (3) is no longer required and can be deleted from the model.

The minimum required time between Y_{ij} and Y_{nm} variables is defined by transition distance matrix T', considering bay positions and the indices of QCs. For example, if QC 2 is assigned to task i and QC 4 is assigned to task n where i and n are located at bay 9 and bay 8, respectively, then this constraint ensures that there must be at least 5 time units of temporal distance between the processing of task i and n because of crane interference and safety margin requirements, regardless of their relative order (see Figure 4.1).

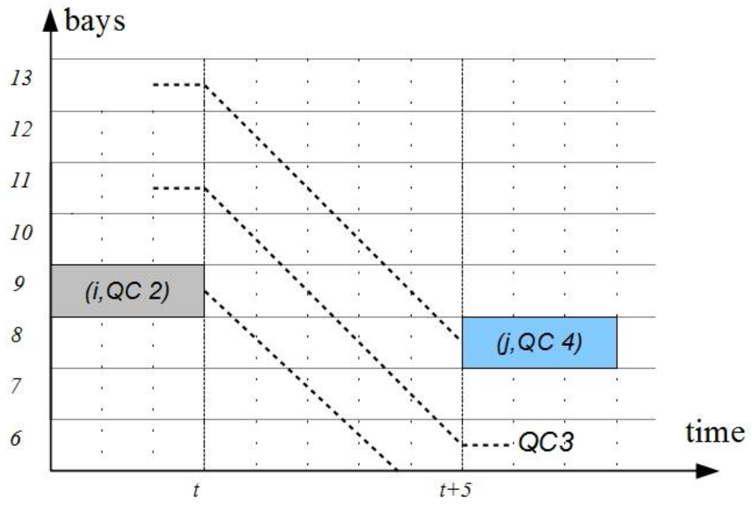


Figure 4.1: An example of constraint set 12.

4.4 Lower Bounds for QCSP

Two simple lower bounds are calculated for the makespan of QCSP with container groups. It would be very helpful to generate tight lower bound values for QCSP instances because of two main reasons. First, a tight lower bound allows determining the solution quality of the CP results more accurately. Moreover, a tight lower bound helps the CP model to terminate earlier with an optimal result when $CP\ result$ equals to $lower\ bound$.

These two lower bounds are presented below:

Theoretical LB:

The first lower bound is widely used in uniform parallel machine scheduling problems. In this simple lower bound, the total duration of all tasks is calculated and then divided by the total number of machines. That is,

$$\boxed{LB1 = \frac{\sum_{i \in T} p_i}{nbC},}$$

where p_i and nbC refer to the processing time of task i and the total number of QCs in the instance, respectively. In this case, it is assumed that machines are working without any interruptions during the makespan. For example, consider an instance with three tasks and two identical machines. Processing times for the tasks are 5, 17 and 8, respectively. This lower bound is 10, while the minimum makespan is 17, therefore the gap between optimal value and this preemptive lower bound is 70%. If $Number\ of\ tasks\ /Number\ of\ Machines$ ratio grows, $LB1$ may probably end up with smaller gaps because of the higher possibility of better distribution of total workload.

The MIP model:

In this study, the results show that most of the time the simplest parallel-machine scheduling lower bound TP/nbC provides considerably closer values to the optimal solution value. However, in some instances the gap with this simple lower bound reaches 20%. It is observed that, this result is due to the excessive workload of some bays and/or some operational restrictions caused by safety distances. After this observation, a simple yet powerful bin-packing based mixed-integer programming model was developed to find tighter lower bounds (LB) for QCSP with container groups, especially in such cases described above.

In this relaxed QCSP model, all sets and parameters are the same with those of the proposed CP model. Additional parameters and decision variables are presented below:

R_{ij} a binary decision variable that takes value 1 if task i is assigned to $QC\ j$, otherwise 0, $\forall i \in T, \forall j \in C$;

A_{jb} a binary decision variable that takes value 1 if $QC\ j$ is assigned to at least one task of bay b, otherwise 0, $\forall b \in B, \forall j \in C$;

LB an integer decision variable that represents a lower bound on the makespan;

$Load_b$ a parameter that calculates the total workload of two adjacent bays; that is,

$$Load_b = \sum_{i \in S_b} p_i + \sum_{i \in S_{b+1}} p_i \qquad \forall b \in B/\{nbB\};$$

M a very large number.

Then, the mixed-integer programming model is formulated as follows:

minimize LB

Subject to:

$$\sum_{j \in C} R_{ij} = 1 \qquad \forall i \in T \qquad (14)$$

$$LB \geq TP/nbC \qquad (15)$$

$$R_{ij} = 0 \qquad \forall i \in T, \forall j \in C, l_i > nbB - 2(nbC - j) \text{ or } l_i < 2j - 1 \qquad (16)$$

$$\sum_{i \in T} R_{ij} < MA_{jb} \qquad \forall j \in C, \forall b \in B \qquad (17)$$

$$LB \geq \sum_{i \in T, j \in C} R_{ij} p_i + \sum_{j \in C, b \in B} A_{jb} - 1 \qquad (18)$$

$$LB \geq max(Load_b, \forall b \in B - \{nbB\}) + 1 \qquad (19)$$

$$R_{ij} \geq 0, A_{jb} \geq 0 \qquad \forall i \in T, \forall j \in C, \forall b \in B \qquad (20)$$

In this model, each task is assigned to a QC by constraint set (14) while minimizing the workload of the densest QC. Constraint (15) defines the parallel-machine scheduling pre-emptive lower bound. Task-to-QC assignments are also restricted by the constraint set (16) which determines the rail-track area for the vessel. In order to add travel times in the simplest form, QC-to-bay assignments are tracked for each QC by using two-indexed binary decision variables in (17). Constraint (18) defines one lower bound with respect to the workload of QCs. If a QC is assigned to m different bays, then $(m-1)$ time unit is added to the total workload of this QC. $(m-1)$ time units are added rather than m because the initial position of the QC and one of the assigned bays can be the same; therefore, the initial travel of this QC before starting to process assigned tasks will not be needed. Also the safety margin requirements are considered from a different, but a simpler, perspective. It is not possible to process two adjacent bays simultaneously with two different QCs because of the safety margin requirements. First listing total workloads of each bay and then selecting the maximum total workload of two adjacent bays will give us another potential lower bound for the instance. Call this value V, then LB is always greater than or equal to $V + 1$ because of the safety margin requirements; that is, if these two adjacent bays are processed by the same QC, one unit of travel time is required for that QC to travel between these bays. On the other hand, if these two bays are processed by two different QCs, the second QC cannot start processing the second bay before the first QC leaves the other bay because of the safety margin. Therefore, again one time unit is required. Constraint set (19) defines this lower bound.

Non-crossing, non-interference and precedence constraints are not considered in this MIP model, hence it represents a relaxed version of the QCSP. Nevertheless, the model generates very tight lower bounds for the QCSP quickly. For example, in one of the instances of Meisel and Bierwirth (2011), TP/nbC is 1000, however the LBobtained from the above MIP is 1174. Note that the best observed result for this instance is also listed as 1174 in their study. In addition, the MIP model is solved to the optimality in 2 minutes, most of the times. For most of the instances, it is solved in less than 30 seconds.

4.5 Descriptions of the Constraints

In this section, each constraint of the CP model is introduced in more detail with the help of relevant bay-time illustrations. Corresponding MIP constraints (if any) are also identified for each constraint in the CP model. Note that there may not be a hundred percent correspondence between two models, since they are built by using two different modeling techniques.

4.5.1 Makespan constraint

$$Z \geq LB \tag{0}$$

$$Z = max_{\forall i \in T, \forall j \in C} \left(end\left(Y_{ij}\right)\right) \tag{1}$$

Since the objective function of the problem is to minimize the makespan which is denoted by an integer decision variable Z, the definition of the makespan must be declared in the objective function or as a problem constraint (1). The term makespan refers to the completion time of the latest element of the schedule. In other words, it is the completion time of the quay crane which ends processing lastly among other QCs. The makespan may also be defined directly in the objective function line. However, while using the CP Optimizer, it can be ineffective to define more complex objective functions (i.e. weighted sum of makespan and completion times of QCs) directly in the objective function line.

Constraint (0) defines a lower bound for each QCSP instance. In Section 4.4, the procedure to obtain this value is introduced in details. Lower bound can be added as a problem constraint or it can simply cut the domain of decision variable Z at the beginning; that is, $domain(Z) = [LB, TP]$. Computational experiments with different instances show that the latter consistently reached better schedules in a shorter time; therefore, we deleted constraint (0) from the model. The reason behind such a solution performance difference is not known exactly because of the black-box nature of the commercial off-the-shelf software. However, most likely, keeping the search space tighter at the beginning helps the constraint programming engine to find good schedules more effectively.

4.5.2 Assignment Constraint

$$assignment\left(X_i, \left(Y_{ij} \,\middle|\, \forall j \in C\right)\right) \quad \forall i \in T \tag{2}$$

The assignment global constraint ensures that each task must be assigned to a one and only one quay crane. It has a very critical function in the model because these task-to-QC assignments are required to be able to investigate other constraints which consist of optional variables. Note that, the whole model, except this constraint, is constructed by only using the Y_{ij} optional variables.

At the beginning of the execution of the model, all optional interval variables are set to be inactive. Then, this constraint is processed firstly to determine which optional variables are active in the model. Then, search space is investigated based on the current task-to-crane assignments. These task-to-crane assignments are continuously modified by backtracking and LNS algorithms. This iterative procedure continues until an optimal solution is found and its optimality is also proved. More formally, if $X_{i'}$ variable is assigned to $Y_{i'j'}$ optional interval variable, then $Y_{i'j'}$ variable becomes active in the model and its domain set to be $domain(Y_{i'j'}) = [0, TP]$. At the same time other $Y_{i'j}$ variables for all $j \in C$ where $j \neq j'$ becomes inactive and their domain set to be $domain(Y_{i'j}) = \{\emptyset\}$.

Note that, this constraint set of CP corresponds to constraint sets (M4) and (M11) of MIP model.

4.5.3 Non-Overlapping Constraint

$$disjunctive\left(Y_{ij} \mid \forall i \in T\right) \quad \forall j \in C \tag{3}$$

It is previously described in the global constraints section in details that what disjunctive global constraint does and which propagation algorithm is working in the background. As also described in the same section, in simple terms, two tasks *j* and *j'* can only be non-overlapping if any of them ends before the other one starts. In this case, disjunctive global constraint ensures that all tasks assigned to the QC *j* must be non-overlapping. More formally, Y_{ij} interval variables where $\forall i \in T$ that are assigned to the same QC by assignment constraint (3) are non-overlapping for each QC j where $\forall j \in C$ because of this global constraint.

This constraint not only makes complex functions easy to represent in the model, but also it has very effective and well-studied propagation algorithms. This is why disjunctive global constraint is one of the most well known constraints in the constraint programming context, and is also used to model several problem concepts in this study.

Note that this constraint set of CP model partially corresponds to the constraint set (M10) of MIP model.

4.5.4 Non-Interference Constraint

$$disjunctive\left(Y_{ij} \mid \forall i \in S_b, \forall j \in C\right) \qquad \forall b \in B \tag{4}$$

In QCSP, the tasks which are located in the same bay should not overlap to have a feasible schedule in terms of non-overlapping of tasks. Such cases are prevented in the model by using another constraint, non-interference. In other words, non-interference constraint indicates that in a single bay two QCs cannot work simultaneously because QCs are working on the same rail-track. This constraint itself is not sufficient to prevent all possible interferences among QCs, therefore we also need to keep some definite safety distance between two adjacent QCs. Note that this constraint set of CP model partially corresponds to the constraint set (M10) of MIP model.

4.5.5 Safety Margin Constraint

$$disjunctive\left(Y_{ij}, Y_{nm} \,\middle|\, \forall i \in S_b, \forall n \in S_{b+1}\, \forall j, m \in C \right) \qquad \forall b \in B - \{nbB\} \qquad (5)$$

Since all QCs are operated on the same rail-track, margins between QCs must be carefully investigated. Therefore while scheduling QCs, we need to keep a pre-determined space between two adjacent QCs at any time, which is called safety distance or safety margin. In parallel with the literature, safety margin is assumed to be one bay length in this study. That is, two QCs cannot work on adjacent bays simultaneously.

We need this margin because of two reasons. First, potential interference of arms of QCs is prevented. Second, potential collusion of bodies of QCs during their travels among the rail-track are also prevented by this constraint.

In Figure 4.2 below, QCs operate and travel within the minimum allowable distance to each other since safety distance of 1 bay defines the minimum distance must be kept between two adjacent vessels. Note that this constraint of CP model corresponds to the constraint (M10) of MIP model.

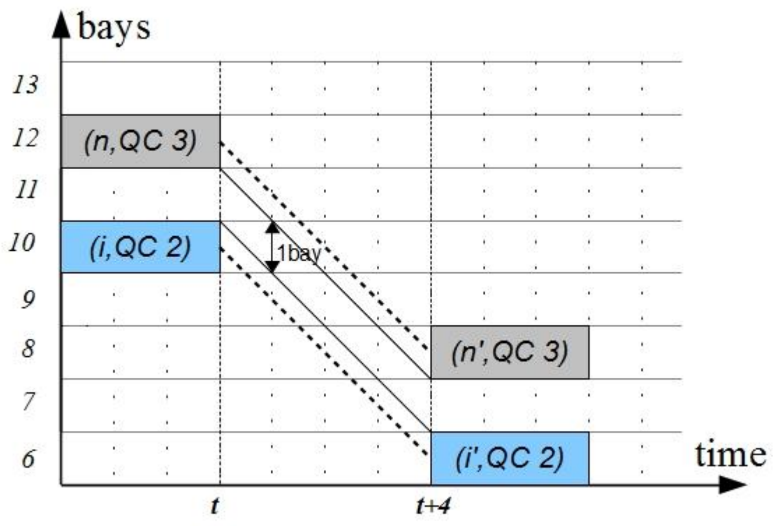


Figure 4.2: Safety margin constraint.

4.5.6 Non-Allowed Assignments Constraint

$$(l_i > nbB - 2(nbC - j)) \vee (l_i < 2j - 1) \rightarrow presence(Y_{ij}) = 0 \qquad \forall i \in T, \forall j \in C \quad (6)$$

As noted in common deficiencies section, Lee and Chen (2010) claim that all QCs assigned to a vessel must stay between the leftmost and the rightmost bays of the vessel. We can simply see these two bays as the boundaries of the vessel, and it is assumed that the rail-track area between these boundaries is allocated to QCs which are assigned to the vessel. Accordingly, if a QC passes these boundaries, it will probably enter rail-track areas which are allocated for another vessel because of very intense berth allocations. Note that, most of the time a CT practitioner tends to keep the distance between berth positions of vessels at minimum (Vis and de Koster (2004)). Therefore QCs that pass out these boundaries will probably cause QC collusions or considerable inefficiencies in QC scheduling for other vessels by blocking other QCs to reach some bays.

Consider a generic instance with nbB bays and nbC QCs. Then some task-to-QC assignments at the left side is restricted, for example QC 2 cannot reach to tasks which are located at bays 1 and 2 because of QC 1, and similarly QC 3 cannot reach to tasks located at bays 1, 2, 3 and 4 because of QC 1 and QC 2. It is similar for all QCs and can be generally represented as $(l_i < (1 + s)(j - 1))$. Note that, by constructing this set of constraints, we assumed that there is a safety margin of 1 bay, that is $s = 1$. For right side of the vessel, similar restrictions are required to keep QCs between boundaries of the vessel during makespan. That is, QC $nbC - 1$ cannot reach bays nbB and $nbB + 1$ because of the presence of QC nbC on a rail-track, and it goes on for all QCs $(l_i > nbB - (1 + s)(nbC - j))$. Then a task-to-QC assignment is forbidden if at least one of these two conditions is satisfied. This is denoted by OR (∨) operator.

Consider an instance with $nbC = 3, nbB = 10$ and task 5 is located at bay 9. We should check whether it is possible to assign task 5 to QC 2:

$$(9 > 10 - 2(3 - 2)) \vee (9 < 2(2) - 1) \rightarrow presence(Y_{ij}) = 0$$

$(TRUE) \vee (FALSE) \rightarrow presence(Y_{ij}) = 0$

$TRUE \rightarrow presence(Y_{ij}) = 0$.

Also, in our computational experiments we observed that, in at least one of the optimal solutions (if any), QCs stay within the boundaries without using this constraint. However, other optimal solutions will cause inefficiencies or QC collusions. Therefore presence of this constraint is very important, especially for keeping QCs within the boundaries in non-optimal schedules.

Note that this constraint set of CP model corresponds to constraint sets (M18) and (M19) of MIP model.

4.5.7 Travel Time Constraint

$$presence(Y_{ij}) \wedge presence(Y_{nj}) \rightarrow \Big(end(Y_{nj}) + |l_i - l_n| \leq start(Y_{ij})\Big) \veebar \Big(end(Y_{ij}) + |l_i - l_n| \leq start(Y_{nj})\Big) \quad \forall i,n \in T, \forall j,m \in C, (i \neq n) \qquad (11)$$

In QCSP, each QC travels along the rail-track to process tasks located at different bays. This horizontal movement requires some amount of time which is called travel time. In this study, it is assumed that for a QC to travel from one bay to an adjacent bay takes one unit of time.

In most of the previous QCSP studies, a set of three-indexed decision variables that keeps the sequence information is used. This 0-1 integer decision variable X_{ijk} takes value 1 if and only if task i is processed just after task j at QC k (see the section of MIP model). Therefore travel time was easily added to the model easily by the help of such a decision variable. In this proposed CP model of QCSP, however, only a two-indexed decision variable, which is not keeping the sequence information, is preferred. This two-indexed decision variable is enough to model travel times correctly without bringing any modeling difficulty because it is possible to represent such similar problem constraints straightforwardly in CP context.

Constraint set (7) ensures that between executions of any two tasks which are assigned to the same QC, there must be a time equal to the absolute value of the difference between their locations (bays) of these tasks. This constraint works correctly without requiring any sequence information because it states not only the minimum required time between successive tasks, but also denotes minimum required times between any two tasks assigned to the same QC. Also it is no more required to be known which task is going to be processed before other tasks that are assigned to the same QC because exclusive-or logical operator can be directly used in any CP model. As a result, this constraint is enough to express travel time of a single QC correctly, without requiring any other additional decision variable. In Figure 4.3, the illustration shows the schedule generated by the help of this constraint. It shows correct travel times implied for a single QC. For example, tasks 1, 2, 3 and 4 are located at bays 2, 5, 5 and 6, respectively. Note that in this instance QC is initially located at bay 1.

Also note that this constraint set of CP model corresponds to the constraint sets (M6) of MIP model.

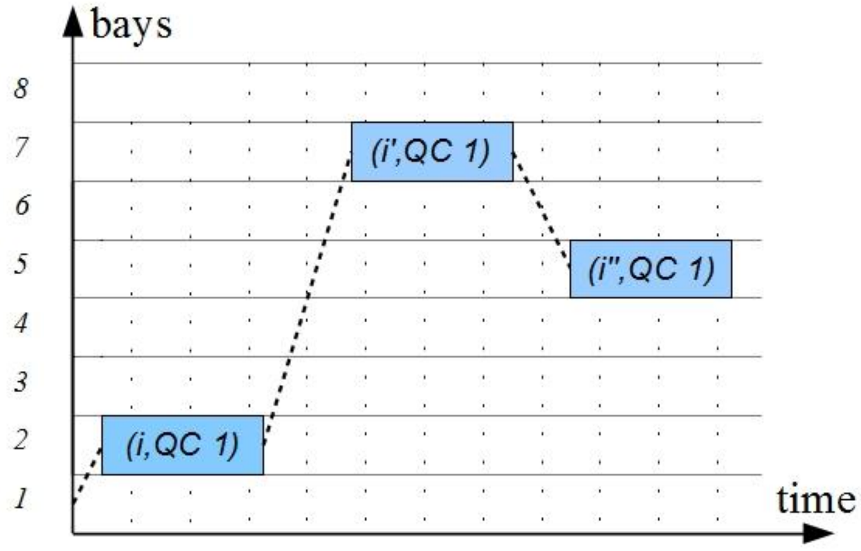


Figure 4.3: Travel times of a QC.

4.5.8 Revised Travel Time Constraint

$disjunctive(S'_j, T'')$	$\forall j \in C$	(13)

The constraint set (5) is replaced by constraint set (13), which consists of set of sequence variables correlated with disjunctive global constraints. As previously said, using global constraints rather than others would probably end up with better domain reductions. Therefore constraint set (13) is preferred to represent travel times. Let's remind definition of this variable :

> S'_j is a sequence variable that keeps the permutation of the tasks to be processed by QC j, $domain(S'_j) = \{permutation\ of\ Y_{ij} \mid i \in T\ and\ presence(Y_{ij}) = 1\}$ $\forall j \in C$, with transition distance matrix $T''_{in} = |li - ln|$, $\forall i, n \in T$. Namely, this matrix keeps the minimum time required between Y_{ij} and Y_{nj} if they are executed sequentially.

In more details, S'_j keeps the permutation of tasks which are assigned to QC j based on their starting times. That is, S'_j by itself only defines the sequence based on starting times and to make these tasks also non-overlapping we need to correlate these variables with disjunctive constraints, since a QC can process at most one task simultaneously. As a result, constraint set (13) not only provides a more direct representation of travel times, but also it is faster and more efficient from (6) by far.

Note that this constraint set of CP model corresponds to the constraint sets (M2) to (M6) of MIP model.

4.5.9 Precedence Constraint

$$end(Y_{ij}) \leq start(Y_{kr}) \qquad \forall < i, k > \in Prec, \forall j, r \in C, (i \neq k) \qquad (8)$$

Constraint set (8) simply defines precedence relationships among tasks. That is, if task i is defined to precede task k, it is ensured by this simple set of temporal constraints for all QCs. There is no precedence relationship among QCs, because precedence relationships among tasks are obtained from their locations on a vessel, denoted by a stowage plan. Precedence constraints in CP-scheduling context may reduce domains significantly and therefore make good solutions easier to be found. For example if Y_{ij} is present and task i precedes task k, then the CP engine straightforwardly reduce domains of $Y_{kj'}$, that is, $domain(Y_{kj'}) = [end(Y_{ij}), TP]$.

Note that this constraint of CP model corresponds to the constraint (M7) of MIP model.

4.5.10 Non-Crossing Constraint

$$presence(Y_{ij}) \wedge presence(Y_{nm}) \rightarrow$$
$$\Big(end(Y_{nm})+l_n - l_i - (1+s)(m-j) \leq start(Y_{ij})\Big) \veebar \Big((end(Y_{ij}) + l_n - l_i - (1+s)(m-j) \leq start(Y_{nm})\Big)$$
$$\forall i, n \in T, \forall j, m \in C, (m > j), (l_n < l_i + (1+s)(m-j)) \quad (9)$$

$$presence(Y_{ij}) \wedge presence(Y_{nm}) \rightarrow$$
$$\Big(end(Y_{nm})+l_n - l_i + (1+s)(j-m) \leq start(Y_{ij})\Big) \veebar \Big((end(Y_{ij})+l_n - l_i + (1+s)(j-m) \leq start(Y_{nm})\Big)$$
$$\forall i, n \in T, \forall j, m \in C, (m < j), (l_n > l_i - (1+s)(j-m)) \quad (10)$$

At a single port, all QCs are located on the same rail-track; therefore they cannot cross each other. Assume that all QCs are labeled in increasing order from left to right where QC 1 and QC nbC indicate the rightmost and the leftmost QCs, respectively. That is, it must be ensured that QC j always stays between two adjacent QCs, QC $j-1$ and QC $j+1$ (if both exist), otherwise the schedule will be infeasible.

To ensure a QC not to cross other QC during operations, we need a two-way control of QC positions. Therefore at any time QC j ($\forall j \in C, j < nbC$) must stay at the left side of QC $j+1$. Also we need to control QC positions from other side; that is, QC j ($\forall i \in C, j > 1$) must stay at the left side of QC $j-1$. By applying this two-way control for each QC, non-crossing of QCs is ensured. Of course QCs can pass some relative positions at different times. That is, QC j can work at bay 10 and QC $j+1$ can work at bay 7. This situation can also occur if two operations are held in different times. In other saying, such a situation can occur if and only if these two operations are non-overlapping. At its simplest form, non-crossing of QCs can be restricted by constraint sets (9') and (10').

$$presence(Y_{ij}) \wedge presence(Y_{nm}) \rightarrow$$
$$\left(end(Y_{nm}) \leq start(Y_{ij})\right) \veebar \left((end(Y_{ij}) \leq start(Y_{nm})\right)$$
$$\forall i,n \in T, \forall j,m \in C, (m > j), (l_n > l_i) \qquad (9')$$

$$presence(Y_{ij}) \wedge presence(Y_{nm}) \rightarrow$$
$$\left(end(Y_{nm}) \leq start(Y_{ij})\right) \veebar \left((end(Y_{ij}) \leq start(Y_{nm})\right)$$
$$\forall i,n \in T, \forall j,m \in C, (m < j), (l_i > l_n) \qquad (10')$$

However, these sets of constraints are not sufficient to ensure feasibility of schedules. There must be a definite space of time between these two non-overlapping operations because of travel times and safety margins associated with QCs.

Accordingly, travel times and safety distances must be correctly implemented into non-crossing constraint. In most of the QCSP literature, this amount of space of time is miscalculated; it causes interferences and QC collusions, therefore the schedules generated in these studies are not feasible. This deficiency is identified and corrected by Bierwirth and Meisel (2009). They claimed that, between Y_{ij} and Y_{nm} there must be Δ_{ijnm} unit space of time is needed.

$$\Delta_{ijnm} = \begin{cases} l_n - l_i - (1+s)(m-j), & if\ (m > j), \left(l_n < l_i + (1+s)(m-j)\right) \\ l_n - l_i + (1+s)(j-m), & if\ (m < j), \left(l_n > l_i - (1+s)(j-m)\right) \\ 0 & otherwise \end{cases}$$

By adding Δ_{ijnm}into (9’) and (10’), they turned into set of constraints (9) and (10).

Consider Figure 4.4 below. Tasks i and n have processing times of 20 and 15, respectively. Also QC 4 and QC 5 are assigned to process tasks i and n located at bays 11 and 9, respectively without causing any infeasibility. By constraint sets (9) and (10), it is ensured that these two operations are not only non-overlapping but also there are at least 4 time units between end of the former and start of the latter. More information about this correction can be found in Bierwirth and Meisel (2009).

Note that this constraint set of CP model corresponds to the constraint sets (M12) and (M13) of MIP model.

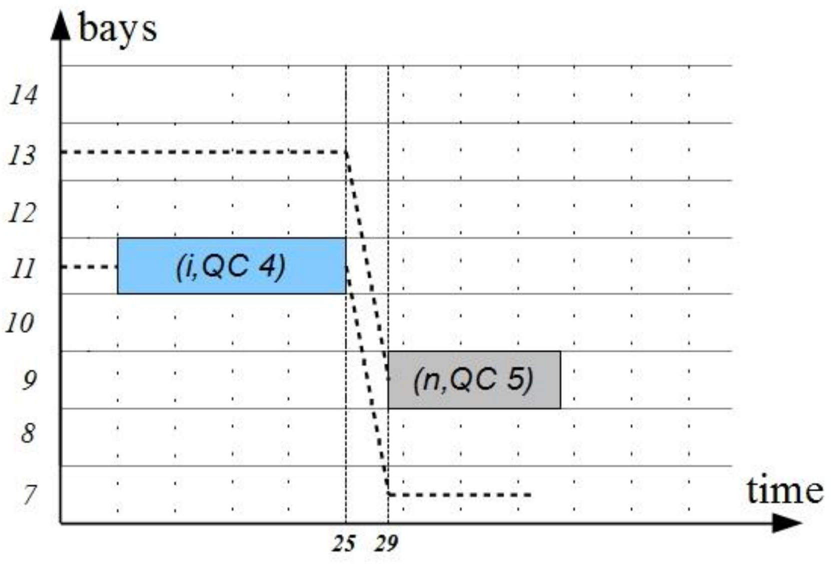


Figure 4.4: Non-crossing constraint.

4.5.11 Revised Non-Crossing Constraint

$$disjunctive(S_{ijnm}, T') \qquad \forall i,n \in T,\ \forall j,m \in C, m \geq j, l_n \leq l_i + (1+s)(m-j) \quad (12)$$

While testing and verifying the correctness of non-crossing constraint, it is observed that there is symmetry between these set of constraints. That is, non-crossing of each QC *j* to other QC *j'* is checked twice. Consider the instance presented in previous section and in Figure 4.4. If constraint sets (9) and (10) are examined carefully, it can be seen that (9) checks QC 4's crossing of QC 5, while (10) checks QC 5's crossing of QC 4. Clearly one of them is enough to ensure non-crossing of QCs.

This brings a considerable computational work and should be removed. By replacing constraints sets (9) and (10) with constraint set (12) we also removed the symmetry which exists in constraint sets (9) and (10). For more effective constraint filtering and lesser computational effort, this symmetry is broken with this new set of revised non-crossing constraint. Therefore, a set of constraint (12), which consists of set of sequence variables correlated with disjunctive global constraints, is constructed.. Let's remind the definition of the variable :

S_{ijnm} is a sequence variable that keeps the relative ordering of tasks i and n processed by QCs j and m, respectively. $domain(S_{ijnm}) = \{order\ of\ Y_{ij}\ and\ Y_{nm}\}$, with transition distance matrix $T'_{ijnm} = |l_i - l_n| + (1+s)(m-j), \forall i,n \in T,\ \forall j,m \in C.$

These sequence variables are auxiliary in the model. Hence they are considered by the engine only when related conditions occurred. For more effective filtering and to break the symmetry, constraint sets (9) and (10) are replaced by a new constraint set (12) which is constructed by S_{ijnm} sequence variables

Note that this constraint set of CP model also corresponds the constraint sets (M12) and (M13) of MIP model.

4.5.12 Correction of the Deficiency Caused by Idle QCs

$$disjunctive(Y_{ij}, Y_{nm})$$

$$\forall i,n \in T, \forall j,m \in C, l_i \geq j, (j \neq nbC), (l_i \neq nbB), (l_n \geq l_i), (m \geq j), (l_n < l_i + (1+s)(m-j-1)+2) \qquad (7)$$

Physical presence (existence) of idle QCs on a rail-track must be respected. That is, between two non-adjacent QCs there should be enough space to be able to accommodate in-between QCs safely. In most of the QCSP literature, physical presence of idle QCs is overlooked and this will end up with potential QC collusions. Therefore the schedules generated by the methods presented in these studies may be infeasible. This deficiency is identified and corrected in two independent studies of Bierwirth and Meisel (2009) and Lee and Chen (2010). Figure 4.5 shows a schedule containing this deficiency.

This schedule is considered to be feasible in previous QCSP models, even though it is infeasible. QCs 2 and 4 move along the rail-track as if QC 3 is not exist because in previous studies safety margin requirements are only considered for active QCs. As a result QCs 2 and 4 both violate the safety margin requirements of QC 3. In the proposed CP model, constraint set (7) is added to correct this modeling error. This constraint set ensures that decision variables Y_{ij} and Y_{nm} must not overlap if $l_i \geq j, (j \neq nbC), (l_i \neq nbB), (l_n \geq l_i), (m \geq j), (l_n < l_i + (1+s)(m-j-1)+2)$. In the previous example

$l_i = 8$ and $l_n = 10$, hence conditions are satisfied. Therefore Y_{ij} and Y_{nm} must be non-overlapping. Assume that $l_i = 7$ and $l_n = 11$, then Y_{ij} and Y_{nm} can be overlapping since QC 3 can be safely accommodated between these two non-adjacent QCs even if it is idle. In such a situation QC 3 is located at bay 9, while bays 8 and 10 are kept empty because of safety requirements.

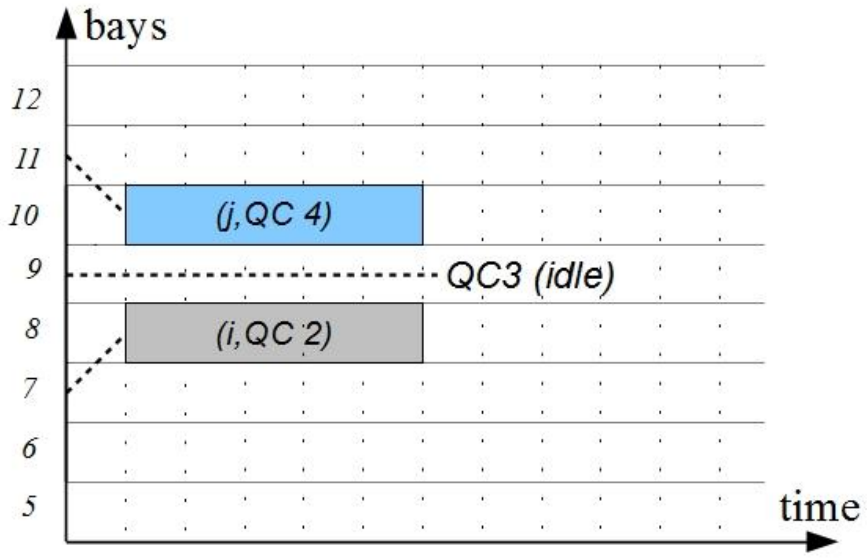


Figure 4.5: A modeling deficiency.

4.6 QCSP with Ready Times

The main aim in this section is to test whether CP can cope with different extensions of the problem, rather than making a comprehensive study on this subject; QCSP with ready times. Meta-heuristics often provide good solution quality for different types of combinatorial problems which CPLEX cannot cope even with their small instances. However, problem-specific heuristics and meta-heuristics are often implied by considering very-specific problem configurations (constraints); that is, they are not flexible to even small additions to the problem (Blum and Roli (2003)). They may be totally unsuitable for the new constraints or may have resulted in poor solution quality. On the other hand in real-world problems, there exist more side constraints than initially projected. Therefore it is important to use a method that can cope with different extensions of the problem; a method that not only is able to add new constraint easily, but also will end up with same good solution quality for different extensions of the initial QCSP problem.

Accordingly, we studied a different version of QCSP, in which QCs have individual ready times. This problem has some practical relevance in any container terminal because it is not desirable that QCs are staying idle without being assigned to any vessel. Instead of this, a QC which is completed its work on a departing vessel will be assigned to another vessel which has already started its handling operations to shorten its completion time. We obtain the constraint programming model for QCSP with ready times by extending the previous QCSP model.

Differently from previous QCSP models with ready times, dummy tasks are added to the model for representing the unavailability of QCs. In other words, it is assumed that these subsequent QCs are processing dummy tasks outside the boundaries of the vessel before their ready times. By this way, we are able to convert the initial QCSP model into QCSP with ready times straightforwardly, while keeping all constraints of QCSP model valid.

New and revised parameters for the QCSP with ready times are listed below.

nbC'	number of QCs that operate for the vessel from beginning to end;
nbL	number of subsequent QCs that will be able to start operating for the vessel from the left-side at their ready times;
nbR	number of subsequent QCs that will be able to start operating for the vessel from the right-side at their ready times;
T'	revised set of tasks, $T' = \{1 - nbL, \dots, nbT + nbR\}$;
B'	revised set of bays, $B' = \{1 - nbL(1+s), \dots, nbB + nbR(1+s)\}$;
C'	revised set of quay cranes, $C' = \{1 - nbL, \dots, nbC' + nbR\}$;
$ready_j$	ready time for each QC $j \in T'$.

T, B and C in the QCSP model are replaced by revised sets T', B' and C', respectively, in the QCSP with ready times model. All constraints of the QCSP model also exist in the extended model. In addition, we need to fix the time-positions of the variables that corresponds to dummy tasks by adding the constraints below:

$$presence(Y_{ii}) = 1 \qquad \forall i \in T'/(T \cup \{nbT+1,..,nbT+nbR\}) \qquad (21)$$

$$presence(Y_{nbT+i\,nbC+i}) = 1 \qquad \forall i \in T'/(T \cup \{1-NbL,..,0\}) \qquad (22)$$

$$end(Y_{ij}) \leq ready_j \qquad \forall i \in T', \forall j \in C' \qquad (23)$$

Constraint sets (21), (22) and (23) also reduce the domains of the dummy variables by considering them as fixed time intervals. That is, if the ready time for QC j' is 100, then the dummy task i is created and $domain(X_{i'})$ and $domain(Y_{i'j'})$ are reduced to [0,100] with $p_{i'}$ =100.

$$start(Y_{ij}) \geq ready_j \qquad \forall i \in T, \forall j \in C'/C \qquad (24)$$

Also domains of Y_{ij} decision variables related with subsequent QCs are reinvestigated by constraint set (24). As a result, the domains of these variables for all tasks other than the dummy variables are reduced to $[ready_j, TP]$.

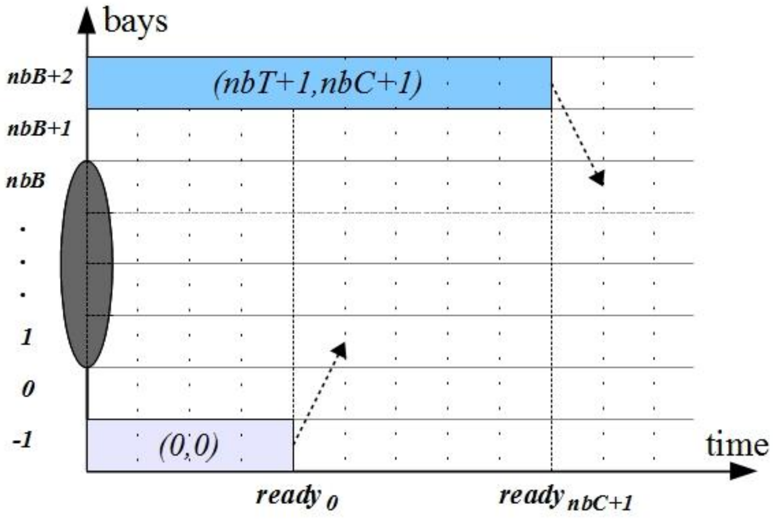


Figure 4.6: Representation of QCSP with ready times.

This extension of the problem is illustrated in Figure 4.6 where the rectangles labeled by (i, j) indicating that dummy task i is processed by QC j. Therefore each subsequent QC can travel in the direction pointed out by arrows in the illustration and starts operating for the vessel after its ready time. Note that, nbC QCs are already operating from the beginning to the end within the bays 1 to nbB.

Also, QCSP model can be converted into QCSP with time windows (Meisel (2011)) straightforwardly by considering the point of view of unavailability of QCs.

4.7 QCSP with Time Windows

As noted in Meisel (2011), in practice, QCs are frequently redeployed among vessels to speed up the operations of high-priority vessels. As a result, the extension of the problem with time windows can be considered as the most appropriate way to represent real-world scheduling operations of QCs. In QCSP with time windows, QCs can have different time windows in which they are available to operate for the vessel. In this study these QCs with time windows are assumed to be operating for adjacent vessels at any time except these pre-defined intervals.

Differently from the literature, we built QCSP with time windows model from the perspective of QC unavailability rather than considering available time intervals. Accordingly, each unavailable period for QCs is represented as a dummy task. These dummy tasks must be performed and are added to the model as fixed intervals; that is, the execution times and the positions of these tasks are fixed. As a result, domains of decision variables related with these tasks are reduced to a single value. Dummy bays are also considered to implement travel times and safety margins correctly. QCSP with time windows model is built by extending the QCSP model developed in Section 4.1. Hence, these dummy tasks are defined on an extended set of bays to denote the waiting locations of these QCs during their unavailability period (at either the left or the right side of the vessel), by considering the non-crossing constraint. For instance, in Figure 4.7, QC 0 and QC $nbC + 1$ are associated with a left and a right side of the boundary, respectively. Note that each QC of this type can only be associated with one side. QCSP with time windows model is then built by extending the QCSP model developed in Section 4. Additional and revised parameters for this extended model are listed below:

nbC'	number of QCs which are available during the makespan;
nbL'	number of QCs which have at least one unavailability and are also associated with left-side;
nbR'	number of QCs which have at least one unavailability and are also associated with right-side;

B''	extended set of bays, i.e., $B'' = \{1 - nbL'(1+s), \dots, 1, \dots, nbB, \dots, nbB + nbR'(1+ s)\}$;
C''	extended set of QCs, i.e. $C'' = \{1 - nbL', \dots, 1, \dots, nbC', \dots, nbC' + nbR'\}$;
TL_j	number of unavailable periods for left associated QC j, $\forall j \in \{1 - nbL', \dots, 0\}$;
TR_j	number of unavailable periods for right associated QC j, $\forall j \in \{nbC + 1, \dots, nbC + nbR'\}$;
T''	extended set of tasks, i.e. $T'' = \{1 - \sum TL_j, \dots, 1, \dots, nbT, \dots, nbT + \sum TR_j\}$;
DL_j	set of indices of dummy tasks which represent the unavailability of QC j related with left side. $DL_j = \{i - \sum_{k=j}^{0}(TL_k) \mid \forall i \in \{1, \dots, TL_j\}\}$ $\forall j \in \{1 - nbL', \dots, 0\}$;
DR_j	set of indices of dummy tasks which represent the unavailability of QC j related with right side. $DR_j = \{nbT' + i - \sum_{k=nbC+1}^{j}(TR_k) \mid \forall i \in \{1, \dots, TR_j\}\}$ $\forall j \in \{nbC + 1, \dots, nbC + nbR'\}$;
SL_i	starting times for each unavailable period of left-associated QCs ;
SR_i	starting times for each unavailable period of right-associated QCs.

Note that, sets T, B and C of the QCSP model are replaced by the revised sets T'', B'' and C'', respectively, and p_i and l_i are also extended by considering the durations of unavailability and the dummy bays for retaining unavailable QCs. Also some of the existing constraints are revised and new constraints are added to the extended model.

$$Z \geq max(end(Y_{ij}) \qquad \forall\, i \in \{1, \dots, nbT\}, \forall j \in \{1, .., nbC'\} \quad (1')$$

$$presence(Y_{ij}) = 1 \qquad \forall j \in \{1 - nbL', \dots, 0\}, \forall i \in DL_j \quad (25)$$

$$presence(Y_{ij}) = 1 \qquad \forall j \in \{nbC' + 1, \dots, nbC' + nbR'\}, \forall i \in DR_j \quad (26)$$

$$start(Y_{ij}) = SL_i \qquad \forall j \in \{1 - nbL', \dots, 0\}, \forall i \in DL_j \quad (27)$$

$$start(Y_{ij}) = SR_i \qquad \forall j \in \{nbC + 1, \dots, nbC' + nbR'\}, \forall i \in DR_j \quad (28)$$

Constraint set (1') represents the revisited constraint set (1), in which dummy tasks are excluded from the calculation of the makespan. Constraint sets (25) and (26) ensure that dummy tasks must be executed by corresponding QCs. Constraint sets (27) and (28) ensure that all dummy tasks are fixed intervals, by fixing the starting times. For instance, if QC j' is unavailable at time period [80,120], then the processing time of the related dummy task i' is 40 and the corresponding domains are fixed as $domain(X_{i'})$=[80,120] and $domain(Y_{i'j'})$=[80,120]. Also the durations of dummy tasks are excluded from the domains of other tasks operated by related QCs because of the combination of constraint sets (27), (28) and (13). For example $domain(Y_{ij}) = [\,early_i, TP] - [\,SL_i,\, SL_i + p_i] \;\forall\; i\; \in T, \forall j \in \{1 - nbL', \dots, 0\}$ for left-associated QCs. A general representation of QCSP with time windows can be found in Figure 4.7.

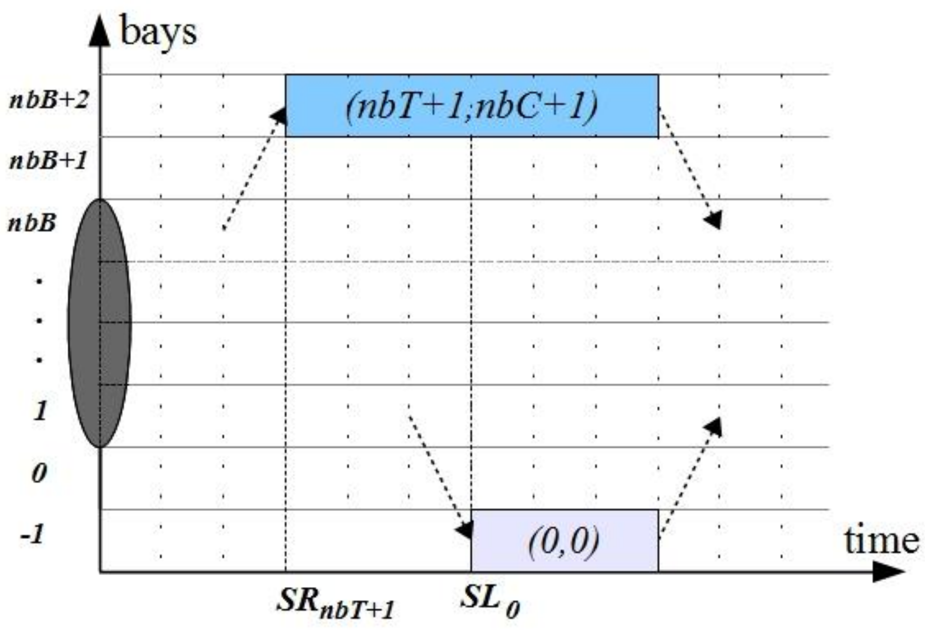


Figure 4.7: QCSP with time windows.

In this generalized instance, there are two QCs with unavailability. QC 0 is at the leftmost and QC $nbC + 1$ is at the rightmost of all QCs assigned to the vessel. They are assigned to adjacent vessel from time SR_{nbT+1} and SL_0 to SR_{nbT+1} + p_{nbT+1} and SL_0+p_0, however, they can operate for the vessel except these intervals plus required travel times. It is important to point out that while some QCs are unavailable because of redeployments, available QCs can travel within the boundaries of the vessel from end to end (bay 1 to bay nbB) by only considering non-crossing constraint.

4.8 Advantages of Using CP for QCSP

In the constraint programming context, $nbT(nbC + 1) + 1$ variables are enough to represent the whole problem, while the corresponding mixed-integer programming model has $nbT^2(nbC + 1) + 3nbTnbC + 3nbC + nbT + 1$ variables. We can see the effects of these numbers with an instance of 20 bays, 30 tasks and 4 quay cranes. For this instance, the proposed CP model and the MIP model can be represented with 151 and 4903 decision variables, respectively. This significant difference is mainly due to the synthesis of rich modeling tools of CP and the efficient modeling by using global constraints. On the other hand, there is no significant difference between the models with respect to the number of constraints. However, most of the constraints only consist of Y_{ij} variables in the CP model. The fact of having a very small number of variables and a large number of global constraints with strong relations helps the constraint programming model to work effectively for QCSP.

In the real-world problems, it is possible that there exist additional constraints other than the ones presented here. For example, quay cranes may have certain time-windows due to different reasons according to Meisel (2011). Another example is the case that task i has to be completed before a given time because there is another vessel in the terminal which is urgently waiting to receive task i before departure. The wide variety of such additional constraints can be easily added to the proposed CP model. Most of the time, however, such additional constraints will not make the problem harder to solve by using CP. Each new constraint that consists of existing variables may probably help to strengthen the inferences, prune more domains, and consequently, will reduce the size of the search space. Therefore, solving more complex problems with CP can be easier. Hence, constraint programming can be an appropriate method to solve not only QCSP but also different extensions of the problem.

CHAPTER 5

COMPUTATIONAL EXPERIMENTS

5.1. Design of Computational Experiments

Computational experiments were conducted to test the performance of the proposed CP model for QCSP with container groups. In this study, QCSPgen (Meisel and Bierwirth (2011)) is used to generate the test instances. There exists one other group of benchmark instances in the literature (Kim and Park (2004)). However in these instances, the number of bays equals to number of tasks, which is unrealistic. Hence we did not use this set of instances.

QCSPgen is a benchmark instance generator for QCSP that allows comparing different models and solution procedures. This generator also provides the most realistic benchmark instances in the literature. QCSPgen is constructed by considering four principles; purpose, comparability, unbiasedness and reproducibility. First of all Meisel and Bierwirth (2011) claim that QCSPgen is generated for the purposes of demonstrating the ability of a procedure or comparing competing procedures regarding one certain QCSP model as well as across different QCSP models. Accordingly, in this thesis, proposed constraint-programming model is fairly compared with the ones of Meisel and Bierwirth (2011) and Meisel (2011) by using QCSPgen. For better comparisons, the researchers compose QCSPgen instances from stowage plan and QC data, to allow considering the service of a vessel under variable QC assignments, travel times and safety margins. Moreover QCSPgen is unbiased; that is, it is not constructed by considering the advantages or abilities of a specific QCSP model. Therefore the generation process is designed by considering typical dimensions of container vessels and reasonable handling volumes by carefully considering task processing times, spatial distributions over bays and precedence relations are generated with well known randomization techniques. As a result, it generates sets of different but identically structured benchmark instances. The benchmark instances generated by this generator are completely reproducible by the help of random seeds. That is, if we generate an instance with same parameters and the random seed, then we always obtain the same instance. It allows that the same data can be instantly reproduced by any researcher around the world, and it creates a quick and fair comparison environment. More detailed information about QCSPgen and the benchmark sets created by this instance generator software can be found in Meisel and Bierwirth (2011).

In this study, Set B and Set C of Meisel and Bierwirth (2011) are since instances with this structure reflect the real-world cases. In all Set B instances, the number of bays (nbB) and the number of QCs (nbC) were taken as 15 and 4, respectively. Also six different numbers of tasks (nbT) were selected as 45, 50, 55, 60, 65 and 70. For the experiments with Set C, six different nbT were selected from 75 to 100 with the same pattern with Set B and nbB and nbC were increased to 20 and 6, respectively. For each number of tasks, ten different instances were generated by using random seeds 1 to 10, in line with the literature. Therefore a total of 120 QCSP benchmark instances were generated by QCSPgen. Since the search phase of CP has randomness (Laborie and Godard (2007)), 10 trials with different random seeds were run for each instance. Set C and Set B benchmark instance sets generated by QCSPgen provides different average number of tasks per bay ratios ranging 3 to 5, which are presented in Table 5.1.

Table 5.1: Task-per-bay ratios.

nbT $(Set\ B)$	$Task-per-bay$ $ratio$	nbT $(Set\ C)$	$Task-per-bay$ $ratio$
45	3	75	3.75
50	3.33	80	4
55	3.67	85	4.25
60	4	90	4.50
65	4.33	95	4.75
70	4.67	100	5

In the instances of Kim and Park (2004) number of tasks per bays is distributed from 0 to 4 uniformly, with an average of 1 task per bay. This means that there will be lots of bays without any task, which is also contradictory to the real-world cases. On the other hand, task-per-bay ratios ranging 3 to 5 with very lesser number of empty bays are considered to be more appropriate to represent the real-world scheduling of QCs. Hence, the benchmark instances of Kim and Park (2004) is disregarded not only in this study but also in recent QCSP literature.

QCSP is one of the problems to be solved frequently in container terminals. Therefore it is important to find a good solution in a short time. Even though CP is a powerful approach for QCSP with respect to the problem structure, it may have a disadvantage as CP may not result in an optimal solution within a reasonable time. Hence, CP is not used as an exact method in our study, which brings the question of when to terminate a CP search. In our experiments, after generating a lower bound value in a very short time, we let CP search the solution space to find a feasible solution F with its objective function value F' and, each time a better solution is found, the constraint $Z \leq F'$ is added to the model. Note that, the incumbent value is taken as LB, if the optimal solution cannot be found by CPLEX within 2-minutes. In all Set B instances LB refer to the optimal solution of the corresponding lower bound model, however in Set C, approximately just one third of LB values refer to optimal ones. Note that, the time spent for solving lower bound model (MIP) is not added to the time limit, that is, in the worst case $(time\ limit - 2)$ minutes is left for CP.

Obviously, when F' is equal to the lower bound (LB), the search is terminated with an optimal solution. Otherwise, CP search will be stopped after $(1.5nbT/10)$ minutes for Set B and half an hour for Set C with the best solution found so far. The results of the computational experiments were compared with the results for Set B and Set C instances presented by Meisel and Bierwirth (2011) and Legato *et al* (2012).

Next, a set of new instances were generated to test the QCSP model with ready times. We assume that three QCs are operating for the vessel from the beginning to end. Also one QC will be available after 300 time-units at the left side of the vessel, and two QCs will be available, one after 600 and the after 900 time units at the right side. Therefore, all Set C instances were modified based on this scheme and then used for testing our QCSP model with ready times. A general representation of QCSP with ready times instances used in this study is given in Figure 5.8.

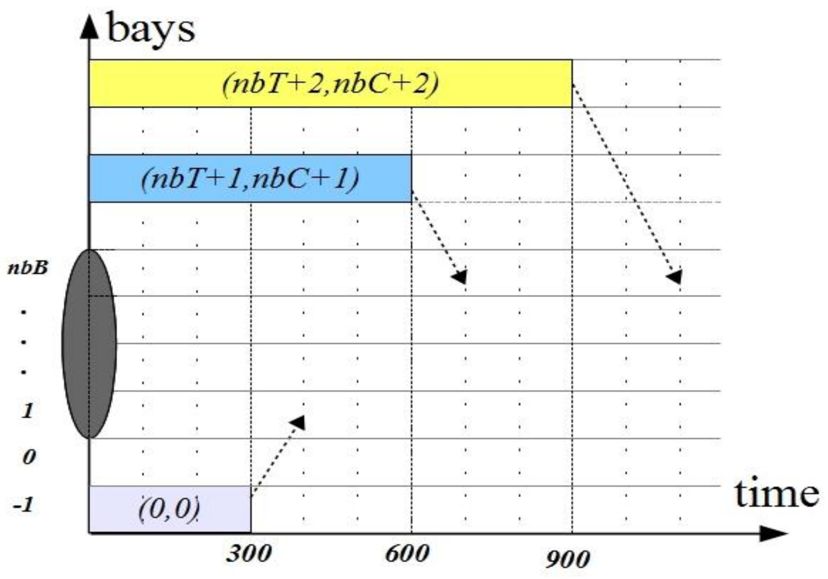


Figure 4.8: QCSP with ready times instance.

To test the QCSP model with time windows, we applied the third instance generation pattern designed by Meisel (2011). In this pattern, six QCs jointly start the operation, but a subset of them (three QCs) is temporarily removed from the vessel to operate for another vessel located at the right side between times 400 and 800, and then turned back to the vessel.

This structure can be seen in Figure 5.9. This situation frequently occurs when QCs at a vessel of low priority are temporarily removed for accelerating the operation of a vessel of higher priority. As with Meisel (2011), these new instances were obtained by extending 10 Set C instances with $nbT = 80$.

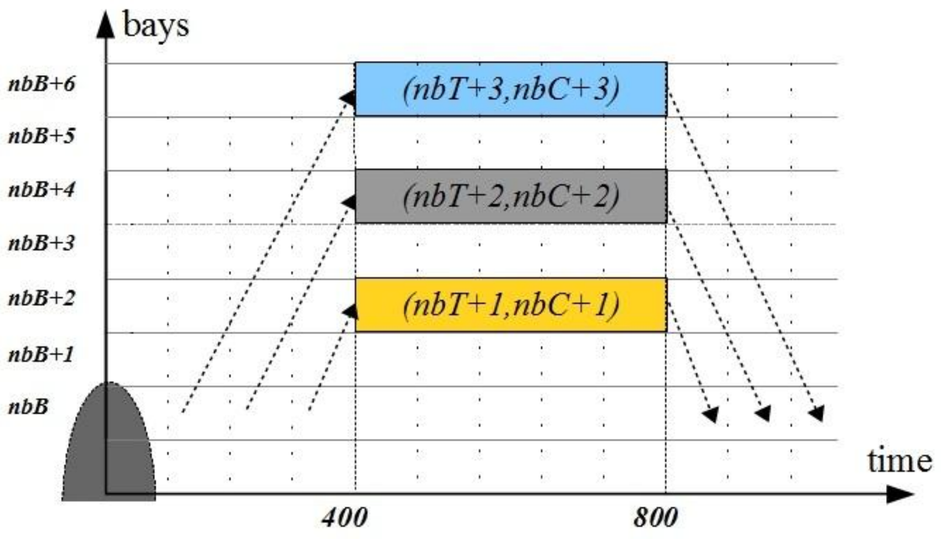


Figure 5.9: QCSP with time windows instance.

Note that, for the computational experiments of QCSP with ready times and QCSP with time windows, we start solving the models immediately rather than generating LB first and then adding it to the model as a tighter lower bound. In these two models, only pre-emptive lower bound is used and that is why no instances were terminated with an optimal solution. Time-limits for these extensions were set as half an hour. For QCSP with time windows 10-minute results are also presented to make a fair comparison with the results of Meisel (2011).

Overview of the computational experiments used in this study is presented in Table 5.2. Note that the number of different instances of each instance size is represented as $(nbT)\ x\ number\ of\ instances$, and time limit is represented in minutes and $LB1$ stands for pre-emptive parallel machine scheduling lower bound.

Table 5.2: Overview of computational experiments.

$Instance\ Sets$	$QCSP\ SET\ B$	$QCSP\ SET\ C$	$Ready\ T.\ SET\ C$	$Time\ W.\ SET\ C$
(size) x #	(45)x10	(75)x10	(75)x10	-
(size) x #	(50)x10	(80)x10	(80)x10	(80)x10
(size) x #	(55)x10	(85)x10	(85)x10	-
(size) x #	(60)x10	(90)x10	(90)x10	-
(size) x #	(65)x10	(95)x10	(95)x10	-
(size) x #	(70)x10	(100)x10	(100)x10	-
Time-limit	1.5nbT/10	30	30	10 & 30
Total ins.	60	60	60	10
Total Runs	600	600	600	100
LB	MIP	MIP	$LB1$	$LB1$

5.2 Results

IBM ILOG's CP Optimizer 12.3 and CPLEX 12.3 were used to solve the constraint programming and the mixed-integer programming models, respectively. All tests were conducted on a personal computer with 2.53 GHz processor and 4 GB ram.

First, to show that the proposed CP model is a viable tool for solving QCSP, we analyzed the performance of CP by considering the percentage deviation of CP results from the lower bound LB, i.e., $GAP_{CP} = 100(\overline{Z_{CP}} - \overline{LB}) / \overline{LB}$. Since CP was run for 10 trials with different random seeds for each instance, Z_{CP} means the average result of these 10 trials. Moreover, there are ten different instances for each instance size in Set B and Set C. Hence, $\overline{Z_{CP}}$ and $\overline{LB}$ indicate the average values of 10 instances of the same size.

Second, we show the improvement of the new lower bound values $\overline{LB}$ over the average lower bound values ($\overline{LB'}$) produced by Meisel and Bierwirth (2011), i.e. $GAP_{LB} = 100(\overline{LB} - \overline{LB'})/ \overline{LB'}$. These $\overline{LB'}$ values were derived by CPLEX within a 2-hour time limit.

After showing the success of the proposed CP model on finding near optimal solutions, we compared the solution time of CP model to a UDS heuristic (Bierwirth and Meisel (2009)) and a TPN procedure (Legato *et al* (2012)). We disregarded the gap between the solution times of our method and UDS heuristic because the TPN procedure surpasses the UDS heuristic at each instance size. Hence, the percentage gap between the average solution times of the CP model (t_{CP}) and the TPN procedure (t_{TPN}) is defined as $GAP_t = 100\,(\overline{t_{CP}} - \overline{t_{TPN}})/ \overline{t_{TPN}}$. We also listed the average relative standard deviation $\%RSD = 100(\sigma_{CP}/\mu)$ between trials for each instance size.

As previously denoted, each instance set consists of 10 different instances for each size. Since each instance set consists of instances with six different nbT, there exist 60 different instances in each Set B and Set C. Therefore the results for all 60 instances were represented by taking the average of 10 different trials as well as more detailed average results for each nbT.

5.2.1 Results of Set B for QCSP

Results of the computational experiments for Set B were presented in Tables 5.3 and 5.4. For Set B, the new lower bound LB which is generated in, at worst, 30 seconds provides 1.46% tighter bounds for the problem. Therefore, the average GAP_{CP} value of 0.62% indicates that the proposed CP model is a good alternative to solve the QCSP. The maximum average gap is observed for $nbT = 60$ and it is 0.9%. Furthermore, very low average relative standard deviation values over 10 trials indicate the robustness of the CP approach. During computational experiments for Set B, in 193 of 600 trials, the search was terminated with the optimal result within the time limit, mainly by the help of new lower bound values. Detailed results of 10 trials were also listed in Appendix B.

Table 5.4: Average results for Set B.

$Instances$		$Lower\ Bounds$		%	$makespan$				$Time\ (minutes)$			%	%
nbT	nbC	LB'	$\overline{LB}$	GAP_{LB}	$\overline{Z_{UDS}}$	$\overline{Z_{TPN}}$	$\overline{Z_{CP}}$	$\%RSD$	UDS	TPN	CP	GAP_t	GAP_{CP}
45	4	754.3	770.5	2.15	775.8	775.8	773.4	0.09	11.88	5.73	5.27	-8.03	0.38
50	4	753.4	763.1	1.29	770.9	770.9	769.3	0.09	20.85	13.78	6.8	-50.65	0.81
55	4	753.6	767.1	1.79	771.9	771.9	771.8	0.12	17.97	10.36	7.38	-28.76	0.61
60	4	753.1	764.0	1.45	771.1	771.1	770.9	0.18	21.95	22.47	8.07	-64.09	**0.90**
65	4	753.5	765.9	1.65	769.0	769.0	768.7	0.13	35.3	19.6	8.22	-58.06	0.37
70	4	753.1	756.3	0.42	762.1	761.9	761.3	0.14	37.18	22.59	8.9	-60.6	0.66
$Averages$		753.5	764.5	1.46	770.1	770.1	769.2	0.13	24.19	15.76	7.44	-52.78	0.62

5.2.2 Results of Set C for QCSP

Results of computational experiments for Set C were presented in Tables 5.5 and 5.6 below. For Set C, the solution quality is similar to the results of Set B. Even though the time-limit is set to 30 minutes, the average solution time for this instance set is approximately 16 minutes, because in 494 out of 600 trials, the search was terminated with an optimal value earlier than the time limit (mostly within 10 minutes). The reason for this is not only longer time limit, but also the tightness of the new lower bound values for Set C instances which have more intense workload for QCs than Set B. This fact also resulted in an even lower average relative standard deviation of 0.04% and GAP_{CP} value of 0.29%. Detailed results of 10 trials were also listed in Appendix C.

Table 5.3: Results for each instance of Set B.

$group-size$	ins	LB	Z_m	Z_{CP}	GAP_{CP}	GAP_M	$group-size$	ins	LB	Z_m	Z_{CP}	GAP_{CP}	GAP_M
45	1	754	758	758.6	0.61	0.08	60	1	778	781	779.6	0.21	-0.18
45	2	759	759	759.0	0.00	0.00	60	2	754	756	759.2	0.69	0.42
45	3	754	759	759.2	0.69	0.03	60	3	754	758	760.0	0.80	0.26
45	4	783	789	783.0	0.00	-0.76	60	4	755	765	765.7	1.41	0.09
45	5	754	758	760.4	0.85	0.32	60	5	754	760	760.6	0.88	0.08
45	6	765	789	771.5	0.85	-2.22	60	6	754	758	760.8	0.90	0.37
45	7	795	798	796.0	0.13	-0.25	60	7	754	786	779.0	3.32	-0.89
45	8	754	759	759.6	0.75	0.11	60	8	754	757	758.5	0.60	0.22
45	9	797	797	797.0	0.00	0.00	60	9	784	785	785.0	0.13	0.00
45	10	790	792	790.0	0.00	-0.25	60	10	799	805	800.0	0.13	-0.62
$Avg.$		770.5	775.8	773.4	0.38	-0.30	$Avg.$		764	771.1	770.9	0.90	-0.02
50	1	754	774	774.0	2.65	0.00	65	1	754	758	759.2	0.69	0.16
50	2	768	771	769.0	0.13	-0.26	65	2	799	799	799.0	0.00	0.00
50	3	768	772	769.6	0.21	-0.31	65	3	801	803	802.0	0.12	-0.12
50	4	754	765	763.4	1.25	-0.21	65	4	754	758	758.8	0.63	0.10
50	5	754	762	762.4	1.11	0.05	65	5	754	758	758.5	0.60	0.07
50	6	754	765	765.3	1.52	0.08	65	6	754	757	758.8	0.63	0.23
50	7	775	782	775.0	0.00	-0.90	65	7	754	757	757.5	0.46	0.07
50	8	753	761	758.5	0.73	-0.29	65	8	754	756	756.5	0.33	0.07
50	9	797	798	798.0	0.13	0.00	65	9	754	758	759.5	0.73	0.20
50	10	754	759	758.5	0.60	-0.05	65	10	781	786	782.0	0.13	-0.51
$Avg.$		763.1	770.9	769.3	0.81	-0.19	$Avg.$		765.9	769	768.7	0.37	-0.03
55	1	754	758	758.5	0.58	0.10	70	1	754	766	761.0	0.93	-0.65
55	2	773	783	775.3	0.32	-0.94	70	2	754	764	765.8	1.56	0.24
55	3	777	779	779.0	0.26	0.00	70	3	754	760	759.6	0.74	-0.05
55	4	754	759	763.0	1.18	0.50	70	4	754	760	757.2	0.42	-0.37
55	5	754	758	760.8	0.90	0.36	70	5	754	757	757.0	0.40	0.00
55	6	787	789	787.0	0.00	-0.25	70	6	760	761	761.2	0.16	0.03
55	7	764	768	770.3	0.83	0.30	70	7	754	759	758.6	0.61	-0.05
55	8	754	767	765.7	1.55	-0.17	70	8	754	758	757.8	0.51	-0.04
55	9	800	801	800.0	0.00	-0.12	70	9	753	757	759.0	0.80	0.26
55	10	754	757	759.8	0.76	0.30	70	10	772	779	774.4	0.31	-0.59
$Avg.$		767.1	771.9	771.9	0.61	0.01	$Avg.$		756.3	762.1	761.3	0.66	-0.12

Table 5.5: Results for each instance of Set C.

$group - size$	ins	LB	Z_M	Z_{CP}	GAP_{CP}	GAP_M	$group - size$	ins	LB	Z_M	Z_{CP}	GAP_{CP}	GAP_M
75	1	1177	1178	1177.0	0.00	-0.08	90	1	1003	1014	1010.6	0.76	-0.34
75	2	1003	1011	1014.6	1.16	0.36	90	2	1003	1020	1013.6	1.06	-0.63
75	3	1181	1182	1181.0	0.00	-0.08	90	3	1003	1011	1011.6	0.86	0.06
75	4	1103	1107	1103.0	0.00	-0.36	90	4	1057	1063	1057.0	0.00	-0.56
75	5	1185	1192	1185.0	0.00	-0.59	90	5	1062	1062	1062.2	0.02	0.02
75	6	1118	1123	1118.0	0.00	-0.45	90	6	1193	1193	1193.0	0.00	0.00
75	7	1192	1200	1192.0	0.00	-0.67	90	7	1105	1108	1105.0	0.00	-0.27
75	8	1166	1174	1166.0	0.00	-0.68	90	8	1086	1094	1086.0	0.00	-0.73
75	9	1170	1074	1170.0	0.00	8.94	90	9	1072	1075	1072.0	0.00	-0.28
75	10	1188	1188	1188.0	0.00	0.00	90	10	1049	1049	1049.0	0.00	0.00
$Avg.$		1148	1143	1149.5	0.12	0.64	$Avg.$		1063	1069	1066.0	0.27	-0.27
80	1	1172	1173	1172.0	0.00	-0.09	95	1	1173	1174	1173.0	0.00	-0.09
80	2	1003	1023	1021.0	1.79	-0.20	95	2	1086	1090	1086.0	0.00	-0.37
80	3	1003	1013	1015.6	1.26	0.26	95	3	1003	1014	1013.0	1.00	-0.10
80	4	1196	1202	1196.0	0.00	-0.50	95	4	1135	1138	1135.0	0.00	-0.26
80	5	1029	1036	1029.0	0.00	-0.68	95	5	1137	1144	1137.0	0.00	-0.61
80	6	1109	1117	1109.0	0.00	-0.72	95	6	1052	1055	1053.0	0.10	-0.19
80	7	1193	1201	1193.0	0.00	-0.67	95	7	1164	1173	1164.0	0.00	-0.77
80	8	1011	1040	1016.0	0.49	-2.31	95	8	1003	1015	1010.0	0.70	-0.49
80	9	1192	1192	1192.0	0.00	0.00	95	9	1019	1019	1019.0	0.00	0.00
80	10	1201	1207	1201.0	0.00	-0.50	95	10	1003	1011	1010.0	0.70	-0.10
$Avg.$		1111	1120	1114.5	0.35	-0.54	$Avg.$		1078	1083	1080.0	0.25	-0.30
85	1	1047	1049	1047.0	0.00	-0.19	100	1	1004	1014	1013.2	0.92	-0.08
85	2	1003	1017	1012.0	0.90	-0.49	100	2	1097	1104	1098.0	0.09	-0.54
85	3	1024	1027	1025.4	0.14	-0.16	100	3	1100	1107	1100.0	0.00	-0.63
85	4	1180	1186	1181.0	0.08	-0.42	100	4	1198	1202	1198.0	0.00	-0.33
85	5	1076	1082	1076.0	0.00	-0.55	100	5	1003	1015	1014.2	1.12	-0.08
85	6	1003	1010	1011.2	0.82	0.12	100	6	1135	1136	1135.0	0.00	-0.09
85	7	1193	1195	1193.0	0.00	-0.17	100	7	1095	1098	1095.0	0.00	-0.27
85	8	1097	1105	1097.0	0.00	-0.72	100	8	1151	1151	1151.0	0.00	0.00
85	9	1003	1010	1011.4	0.84	0.14	100	9	1003	1023	1014.8	1.18	-0.80
85	10	1166	1166	1166.0	0.00	0.00	100	10	1003	1015	1013.4	1.04	-0.16
$Avg.$		1079	1085	1082.0	0.28	-0.24	$Avg.$		1079	1087	1083.3	0.43	-0.30

Table 5.6: Average results for Set C.

nbT	nbC	LB	UDS	TPN	$\overline{Z_{CP}}$	$\%RSD$	UDS	TPN	CP	GAP_t	GAP_{CP}
75	6	1138.3	1142.9	1142.9	1139.6	0.01	60.0	51.9	8.4	-83.8	0.11
80	6	1110.9	1120.4	1120.3	1114.9	0.05	54.1	48.0	14.6	-69.6	0.36
85	6	1079.2	1084.7	1084.7	1082.8	0.04	60.0	54.8	20.7	-62.2	0.33
90	6	1063.9	1068.9	1068.8	1066.4	0.06	60.0	56.4	13.6	-75.9	0.23
95	6	1077.5	1083.3	1082.9	1080.2	0.01	60.0	57.5	17.5	-69.6	0.25
100	6	1078.9	1086.5	1085.3	1083.4	0.06	60.0	60.0	19.0	-68.3	**0.42**
$Averages$		1091.5	1097.8	1097.5	1094.6	0.04	59.0	54.8	15.6	-71.6	0.29

Moreover, the overall solution quality of the proposed CP model is almost similar to the UDS and TPN methods. An insignificant advantage of our results in Set B and Set C instances is probably caused by the inexistence of pre-defined initial positions in this study. In just a very few instances of Set B and Set C, non-unidirectional schedules provides considerable (more than 1%) improvement to unidirectional results. Therefore, we can state that overall solution quality of unidirectional schedules for QCSP (or independent-unidirectional schedules) is very similar to the non-unidirectional schedules.

The proposed CP model also cuts the previous best solution time by averaging 52.78% and 71.70% for Set B and Set C; in other words, the problem is solved approximately 2 and 3.5 times faster than previous fastest solution times in the literature for Set B and Set C, respectively, which is a significant improvement for solving QCSP.

5.2.3 Results of Set C for QCSP with Ready Times

Results of computational experiments for Set B are presented in Tables 5.7 and 5.8.

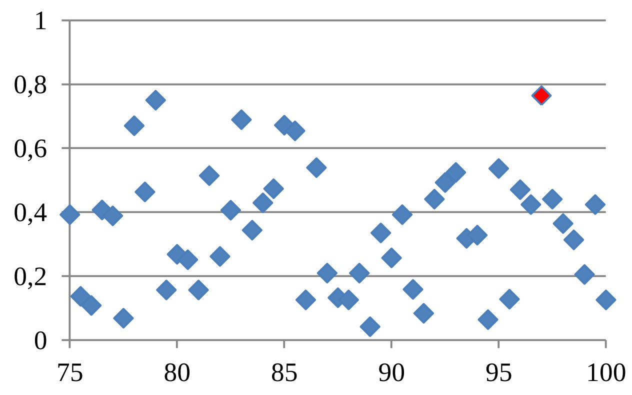
Table 5.7: Results for each instance of Set C (with ready times).

$group-size$	ins	LB	Z_{CP}	GAP_{CP}	$group-size$	ins	LB	Z_{CP}	GAP_{CP}
75	1	1302	1319.6	1.35	90	1	1302	1317.4	1.18
75	2	1302	1321.2	1.47	90	2	1302	1320.4	1.41
75	3	1302	1319.0	1.31	90	3	1302	1316.4	1.11
75	4	1302	1319.6	1.35	90	4	1302	1330.2	2.17
75	5	1302	1331.6	2.27	90	5	1302	1320.8	1.44
75	6	1302	1320.6	1.43	90	6	1302	1316.0	1.08
75	7	1302	1332.6	2.35	90	7	1302	1346.0	3.38
75	8	1302	1331.8	2.29	90	8	1302	1327.4	1.95
75	9	1302	1320.8	1.44	90	9	1302	1320.6	1.43
75	10	1302	1314.8	0.98	90	10	1302	1323.8	1.67
$Avg.$		1302	1323.2	1.63	$Avg.$		1302	1323.9	1.68
80	1	1302	1322.4	1.57	95	1	1302	1324.2	1.71
80	2	1302	1316.4	1.11	95	2	1302	1317.4	1.18
80	3	1302	1315.8	1.06	95	3	1302	1337.4	2.72
80	4	1302	1321.8	1.52	95	4	1302	1324.2	1.71
80	5	1302	1321.8	1.52	95	5	1302	1342.0	3.07
80	6	1302	1330.2	2.17	95	6	1302	1321.2	1.47
80	7	1302	1335.2	2.55	95	7	1302	1322.2	1.55
80	8	1302	1318.8	1.29	95	8	1302	1316.6	1.12
80	9	1302	1323.0	1.61	95	9	1302	1325.2	1.78
80	10	1302	1322.4	1.57	95	10	1302	1340.8	2.98
$Avg.$		1302	1322.8	1.60	$Avg.$		1302	1327.1	1.93
85	1	1302	1329.8	2.14	100	1	1302	1323.2	1.63
85	2	1302	1331.2	2.24	100	2	1302	1337.0	2.69
85	3	1302	1316.8	1.14	100	3	1302	1324.2	1.71
85	4	1302	1330.8	2.21	100	4	1302	1323.2	1.63
85	5	1302	1322.0	1.54	100	5	1302	1327.6	1.97
85	6	1302	1313.0	0.84	100	6	1302	1321.8	1.52
85	7	1302	1316.2	1.09	100	7	1302	1321.4	1.49
85	8	1302	1328.0	2.00	100	8	1302	1328.6	2.04
85	9	1302	1319.4	1.34	100	9	1302	1345.0	3.30
85	10	1302	1320.8	1.44	100	10	1302	1321.4	1.49
$Avg.$		1302	1322.8	1.60	$Avg.$		1302	1327.3	1.95

Table 5.8: Results for QCSP with ready times.

nbT	nbC	$\overline{LB}$	$\overline{Z_{CP}}$	$\%RSD$	$\overline{t_{CP}}$	GAP_{CP}
75	6	1302	1323.2	0.35	30.0	1.63
80	6	1302	1322.8	0.38	30.0	1.60
85	6	1302	1322.8	0.30	30.0	1.60
90	6	1302	1323.9	0.30	30.0	1.68
95	6	1302	1327.1	0.41	30.	1.93
100	6	1302	1327.3	0.36	30.0	**1.95**
$averages$		1302	1324.5	0.35	30.0	1.72

Table 5.9: Individual %RSD values.



The results of the computational experiments for QCSP with ready times were presented in Tables 5.8 and 5.9. In comparision to the GAP_{CP} values for QCSP, the gap is increased because in this case we compared Z_{CP} values with preemptive lower bound not with a specific lower bound model for QCSP with ready times, as we did for QCSP. After investigating the detailed results of Set C for QCSP, we observed that the solution qualities of some definite instances (which are not detected as an optimal, and are not terminated within 30 minutes) are similar to the results for QCSP with ready times. Detailed results of 10 trials were also listed in Appendix D.

In Table 5.9 we presented the $\%RSD$ values for all 60 instances of Set C. The maximum relative standard deviation observed is 0.765%, which indicates that the approach still has very low variance between different trials. However, these levels of variation are consistently higher than the $\%RSD$ for QCSP instances because even in a single trial we cannot terminate the search with an optimal value within the time limit. Like GAP_{CP} values, these $\%RSD$ values are also similar to $\%RSD$ of some specific QCSP instances which are not detected as optimal.

5.2.4 Results of Set C for QCSP with Time Windows

Results of computational experiments for Set C were presented in Tables 5.10 and 5.11. Note that only a subset of 10 instances of Set C (with $nbT = 80$) is selected similar with the literature.

Table 5.10: Results of QCSP with time windows instances.

			$10\ mins.$		$30\ mins.$		
nbT	$ins.$	LB	$\overline{Z_{CP}}$	GAP_{CP}	$\overline{Z_{CP}}$	GAP_{CP}	$imp\ \%$
80	1	1208	1240.2	2.67	1235.8	2.30	0.36
80	2	1208	1255.2	3.91	1252.2	3.66	0.25
80	3	1208	1234.4	2.19	1232.4	2.02	0.17
80	4	1208	1316.2	8.96	1308.6	8.33	0.63
80	5	1208	1260.6	4.01	1249.6	3.44	0.57
80	6	1208	1323.6	9.57	1320.0	9.27	0.30
80	7	1208	1287.2	6.56	1284.6	6.34	0.22
80	8	1208	1285.4	6.41	1283.6	6.26	0.15
80	9	1208	1238.6	2.53	1236.2	2.33	0.20
80	10	1208	1274.0	5.46	1271.6	5.26	0.20
$averages$		1208	1271,5	5.23	1267.5	4.92	0.30

Table 5.11: Average results for QCSP with time windows.

			$10\ minutes$					$30\ minutes$		
nbT	nbC	$\overline{LB}$	$\overline{Z_{CP}}$	GAP_{CP}	$\%RSD$	$\overline{Z_M}$	GAP_M	$\overline{Z_{CP}}$	GAP_{CP}	$\%RSD$
80	6	1208	1271.9	5.23	0.54	1348.9	11.66	1267.46	4.92	0.37

The results from Tables 5.10 and 5.11 indicate that even though GAP_{CP} jumped to 5.23%, the solution quality is still decent, when compared to the results ($\overline{Z_M}$) of Meisel (2011), which is 11.66% (GAP_M). We observe that the unavailability of some QCs during the makespan prevent us from finding solutions very near to preemptive lower bound. That is, for any QC with at least one unavailable period, there is a considerable amount of idle time between the completion time of its last assigned task and the starting time of its unavailability. This unavailability occurs because most of the time a suitable task to be assigned to these idle times cannot be found due to their processing times, locations and precedence relations. As a result, these idle times of QCs ends up with a significant deviation from the pre-emptive lower bounds. Detailed results of 10 trials were also listed in Appendix E.

The improvement of GAP_{CP} values in the additional 20 minutes is relatively low (0.3%), thus we can claim that 10 minutes time limit is enough for the convergence of our approach for solving QCSP with time windows. As expected, the average $\%RSD$ decreased by 0.17% because of the longer time limit. Differently from QCSP, we observed that non-unidirectional schedules can provide very significant improvement in the solution quality. For QCSP with time windows which is the most realistic extension of the original problem. This important finding indicates the importance of generating non-unidirectional schedules especially when QC operations of whole berth are managed together as desired by the practitioners, rather than generating QC schedules for each vessel separately.

CHAPTER 6

CONCLUSION

In this thesis we studied quay crane scheduling which an important seaside operation for any container terminals. It is a common problem for container terminals because all loading and unloading operations of berthed vessels accomplished by QCs. Since there exists different versions of this problem in the literature, we define our problem as QCSP with container groups with travel times, safety distances, precedence relationships, non-crossing and non-interference constraints.

For applicability of any QCSP study to the real-world problems, constraints of these problems should be reflected properly into the model. Therefore the modeling corrections for some deficiencies which are identified in recent literature are taken into account. Since these deficiencies were identified by different researchers, our study is the first which is fixed all of the errors in a single model. Then, a constraint programming model for this version of QCSP is proposed. By using rich modeling tools of the CP, the number of variables in the QCSP is reduced by an order of number of tasks.

The computational experiments show that the proposed CP model proved itself as a fast and a convenient alternative to obtain near optimal solutions for QCSP. The solution quality of the proposed CP model for QCSP has a very little advantage over the most efficient solution methods in the literature while reducing the solution time significantly. For approximately half of the benchmark instances we found better solutions which are independent from initial positions of QCs.

Moreover, QCSP with ready times and QCSP with time windows models are also discussed to show that CP is a proper method for solving not only QCSP but also its more complex extensions. The computational experiments indicate that the non-unidirectional schedules are able to provide significant improvements over the unidirectional ones for the most realistic cases with time windows.

A further research on the search technique for solving QCSP can be dispensable, because we have already observed a very high solution quality for even the largest instances with a large number of optimal solutions. Therefore, the future research could address duplicating the success of constraint programming into solving very-large scaled integrated operations of berth allocations and QC scheduling. Also the CP models presented in this thesis can be converted into QC operations management software by using C++ with constraint propagation libraries. Hence we can get rid of the dependency to commercial software.

Moreover, by developing new lower bound s rather than using theoretical preemptive lower bounds for QCSP with ready times and time windows models, we may significantly reduce the solution time by helping the model to terminate with optimal results.

REFERENCES

Baptiste P. and Le Pape, C. (1996) Edge-Finding Constraint Propagation Algorithms for Disjunctive and Cumulative Scheduling. *Proceedings of the Fifteenth Workshop of the U.K. Planning Special Interest Group*, Liverpool, United Kingdom, 1996.

Bartak, R. (2005), Constraint propagation and backtracking-based search, *First International Summer School on CP*, 2005.

Beck, J. C., and Fox, M. S. (1999), Scheduling Alternative Activities, *Proceedings of the 16th National Conference on Artificial Intelligence,* Orlando, Florida, USA, 18-22, July 1999.

Beldiceanu N., Carlsson M., and Rampon J. (2010), Global constraint catalog (second edition),Technical Report T2010:07, SICS.

Bessiere, C., Regin, J.C., Yap, R.H.C. and Zhang, Y. (2005), An optimal coarse-grained arc consistency algorithm, *Artificial Intelligence*, **165(2)**,165–185.

Bierwirth, C. and Meisel, F. (2009), A Fast Heuristic for Quay Crane Scheduling with Interference Constraints, *Journal of Scheduling*, **12**, 345-360.

Bierwirth, C. and Meisel, F. (2010), A Survey of Berth Allocation and Quay Crane Scheduling Problems in Container Terminals, *European Journal of Operational Research*, **202(3)**, 615-627.

Blazewicz, J., Cheng, T.C.E., Mahowiak, M. and Oğuz, C. (2011), Berth and Quay Crane Allocation: A Moldable Task Scheduling Model, *Journal of Operational Research Society*, **62(7)**, 1189-1197.

Daganzo, C.F. (1989), The Crane Scheduling Problem, *Transportation Research B,* **23(3)**, 159-175.

Dechter, R. and Frost, D (1998), Backtracking Algorithms for Constraint Satisfaction Problems: a survey, *Constraints, International Journal*, **12**, 33-42.

Godard, D., Laborie, P. and Nuijten,W. (2005), Randomized Large Neighborhood Search for Cumulative Scheduling. *ICAPS 2005*, 81-89.

Ebeling , C. E. (2009), Evolution of a Box, *Invention and Technology* **23** (4): 8–9.

Focacci, F., Laburthe, F. and Lodi, A. (2002), Local Search and Constraint Programming, In Glover F. and Kochenberger, G., editors, Handbook of Metaheuristics, *International Series in Operations Research & Management Science 57*. Kluwer Academic Publishers, Norwell, MA, 2002.

Guinet, A. (1993), Scheduling Sequence-Dependent Jobs on Identical Parallel Machines to Minimize Completion Time Criteria, *International Journal of Production Research*, **31(7)**, 1579–1594.

Hansen, P., Oğuz, C. and Mladenović, N. (2008), Variable Neighborhood Search for Minimum Cost Berth Allocation, *European Journal of Operational Research*, **191(3)**, 636-649.

Kim, K.H. and Park, Y.M. (2004), A Crane Scheduling Method for Port Container Terminals, *European Journal of Operation Research*, **156**,752–768.

Kim K.H., Park Y.M. and Ryu K.R. (2000), Deriving decision rules to locate export containers in container yards, *European Journal of Operational Research*, **124**, 89–101.

Laborie, P. and Godard, D. (2007), Self-Adapting Large Neighborhood Search: Application to single-mode scheduling problems, *Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications*, Paris, France, 28-31, August 2007.

Laborie, P. and Rogerie, J. (2008), Reasoning with Conditional Time-intervals, *Proceedings of the 21st International FLAIRS Conference*, Coconut Grove, Florida, USA, 15-17, May 2008.

Laborie, P., Rogerie, J., Shaw, P. and Vilim, P. (2009), Reasoning with Conditional Time-intervals - Part II: an Algebraical Model for Resources, *Proceedings of the 22nd International FLAIRS Conference*, Sanibel Island, Florida, USA, 19-21, May 2009.

Le Pape, C. (1994), Implementation of resource constraints in ILOG Scheduler, *Intelligent Systems* Engineering, **3(2)**, 55 – 66.

Lee, D-H. and Chen, J.H. (2010), An Improved Approach for Quay Crane Scheduling with Non-Crossing Constraints, *Engineering Optimization*, **42(1)**, 1-15.

Lee, D-H., Wang, H.Q. and Miao, L. (2008), Quay Crane Scheduling with Non-Interference Constraints in Port Container Terminals, *Transportation Research E*, **44**, 124–135.

Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P. (1977), Complexity of Machine Scheduling Problems, *Annals of Discrete Mathematics*, **1**, 342-362.

Legato, P., Trunfio, R. and Meisel, F. (2012), Modeling and Solving Rich Quay Crane Scheduling Problems, *Computers & Operations Research*, **39**, 2063–2078.

Lim, A., Rodrigues, B. and Xu, Z. (2007), A m-Parallel Crane Scheduling Problem with a Non-Crossing Constraint, *Naval Research Logistics*, **54(2)**, 115–127.

Lopez-Ortiz, A., Quimper, C.G., Tromp, J. and van Beek, P. (2003), A Fast and Simple Algorithm for Bounds Consistency of the alldifferent Constraint, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'2003)*, 245–250, 2003.

Lustig, I.J. and Puget, J-F. (2001), Program Does Not Equal Program: Constraint Programming and Its Relationship to Mathematical Programming, *Interfaces*,**31(6)**,29-53.

Mackworth, A.K. (1977), Consistency in Networks of Relations, *Artificial Intelligence*, **8(1)**, 99-118.

Meisel, F. (2011), The Quay Crane Scheduling with Time Windows, *Naval Research Logistics*, **58(7)**, 619-636.

Meisel, F. and Bierwirth, C. (2011), A Unified Approach for the Evaluation of Quay Crane Scheduling Models and Algorithms, *Computers & Operations Research*, **38**, 683 –693.

Moccia, L., Cordeau, J.F., Gaudioso, M. and Laporte, G. (2006), A Branch-and-Cut Algorithm for the Quay Crane Scheduling Problem in a Container Terminal, *Naval Research Logistics*, **53(1)**, 45–59.

Nishimura, E., Imai, A., Papadimitriou, S. (2001), Berth Allocation Planning in the Public Berth System by Genetic Algorithms, *European Journal of Operational Research*, **131**, 282–292.

Qiu L., Hsu W.J., Huang S.Y. and Wang H. (2002), Scheduling and routing algorithms for AGVs: a survey, *International Journal of Production Research*, **40**, 745–760.

Park, Y.M., Kim, K.H. (2003), A Scheduling Method for Berth and Quay Cranes, *OR Spectrum*, **25(1)**, 1–23.

Peterkofsky, R.I. and Daganzo, C.F.(1990), A Branch-and-Bound Solution Method for the Crane Scheduling Problem, *Transportation Research Part B*, **24(3)**, 159–172.

Régin, J.-C. (1994), A Filtering Algorithm for Constraints of Difference in CSP, *12th National Conference on Artificial Intelligence*, (AAAI-94), 362–367.

Sammarra, M., Cordeau, J.F., Laporte, G. and Monaco, M.F. (2007), A Tabu Search Heuristic for the Quay Crane Scheduling Problem, *Journal of Scheduling*, **10(4-5)**, 327–336.

Shaw, P. (1998), Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In *CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming), Lecture Notes in Computer Science*, **1520**, 417–431, 1998.

Stahlbock, R. and Voß, S. (2008), Operations Research at Container Terminals: a Literature Update, *OR Spectrum*, **30(1)**, 1–52.

Steenken, D., Voß, S. and Stahlbock, R. (2004), Container Terminal Operation and Operations Research – a Classification and Literature Review, *OR Spectrum*, **26(1)**, 3–49.

Tavakkoli-Moghaddam, R., Makui, A., Salahi, S., Bazzazi, M. and Taheri, F. (2009), An Efficient Algorithm for Solving a New Mathematical Model for a Quay Crane

Scheduling Problem in Container Ports, *Computers and Industrial Engineering*, **56 (1)**, 241–248.

Vilim, P. (2004), O(n log n) Filtering Algorithms for Unary Resource Constraint, *Proceedings of the First International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR'04),* 319–334, Nice, 2004.

Vis, I.F.A. and de Koster, R. (2003), Transshipment of Containers at a Container Terminal: an Overview, *European Journal of Operational Research*, **147(1)**, 1–16.

Wilson, I.D. and Roach, P.A. (2000), Container stowage planning: A Methodology for Generating Computerised Solutions, *Journal of the Operational Research Society*, **51**, 1248–1255.

http://shippingandfreightresource.com/2009/04/16/container-stowage-planning-and-how-it-works/.

ILOG, CPLEX Optimization Studio, User's Manual (2011).

APPENDIX A

OPL MODEL AND DATA FOR QCSP WITH TIME-WINDOWS

MODEL:

```
using CP;

 int NbTask=...;
 int NbCrane=...;
 int NbBay=...;
 int NbLeft=...;
 int NbRight=...;


 range cranes=(1-NbLeft)..NbCrane+NbRight;
 range tasks=(1-NbLeft)..NbTask+NbRight;
 range bays=(1-2*NbLeft)..NbBay+2*NbRight;


  tuple prec
 {
 int i;
 int j;
 }

 {prec} Phi=...;
 int p[tasks]=...;
 int b[tasks]=...;
 int total=sum(i in tasks)p[i];
 int largest=max(i in tasks)p[i];
 int tottal=2000;
 int workload[r in bays]=sum(i in tasks:b[i]==r)p[i];
 int largestbay=max(r in bays)workload[r];
 int dummy_ready[cranes]=...;
 int xx[tasks][cranes]=...;
 int earliest[i in tasks]=maxl(sum(k in tasks:<k,i> in Phi)p[k],0);
 int lastt[tasks]=...;
 int ttt[i in tasks]=...;
 int eee[i in tasks]=maxl(earliest[i],ttt[i]);
 int son[i in tasks]=minl(tottal,lastt[i]);
int largee[i in tasks][j in cranes]=xx[i][j]+dummy_ready[j];
int largestt[i in tasks][j in cranes]=minl(tottal,largee[i][j]);

dvar interval acts[i in tasks] in earliest[i]..son[i] size p[i];
dvar interval actOnRes[i in tasks][j in cranes] optional in
eee[i]..largestt[i][j] size p[i];
int M=total;
 dvar int makespan in 0..total;

int Typet[j in 1..NbBay][k in cranes] = (NbBay+2*NbRight+1)*(k)+(j);
tuple triplet2 { int id1; int id2; int value; };
{triplet2} dist2 = { <Typet[b[i]][j],Typet[b[n]][m],ftoi(abs(b[i]-b[n]+2*(m-
j)))> | i,n in 1..NbTask,j,m in cranes: i!=n && j!=m && b[n]>=2*m-1 &&
b[n]<=(NbBay-(NbCrane-m)*2) && b[i]>=2*j-1 && b[i]<=(NbBay-(NbCrane-j)*2)};


 tuple trickk
 {int i;
 int j;
}
```

```
 {trickk} deneme[i in 1..NbTask][j in cranes][n in 1..NbTask][m in
cranes]={<i,j>,<n,m>};

dvar sequence seq[j in cranes] in all(i in tasks:b[i]>=2*j-1 && b[i]<=(NbBay-
(NbCrane-j)*2)) actOnRes[i][j] types all(i in tasks:b[i]>=2*j-1 &&
b[i]<=(NbBay-(NbCrane-j)*2))i;


dvar sequence seqq[i in 1..NbTask][j in cranes][n in 1..NbTask][m in cranes]
in all(<a,c> in deneme[i][j][n][m]:m>j && i!=n && j!=m &&b[n]>=2*m-1 &&
b[n]<=(NbBay-(NbCrane-m)*2) && b[i]>=2*j-1 && b[i]<=(NbBay-(NbCrane-
j)*2))actOnRes[a][c] types all(<a,c> in deneme[i][j][n][m]:m>j && i!=n && j!=m
&& b[n]>=2*m-1 && b[n]<=(NbBay-(NbCrane-m)*2) && b[i]>=2*j-1 && b[i]<=(NbBay-
(NbCrane-j)*2))Typet[b[a]][c];

 {int} TaskOnBay[t in bays]={j|j in tasks: b[j]==t};

 {int} t1[i in 1..NbTask][j in cranes][m in cranes]={n|n in 1..NbTask:i!=n &&
b[n]<b[i]+2*(m-j) && b[n]>=2*m-1 && b[n]<=(NbBay-(NbCrane-m)*2) && b[i]>=2*j-1
&& b[i]<=(NbBay-(NbCrane-j)*2) && j!=NbCrane+NbRight && m!=(1-NbLeft) && m>j
};
    {int} t3[i in 1..NbTask][j in cranes][m in cranes]={n|n in 1..NbTask: i!=n
&& b[n]!=b[i] && j<NbCrane+NbRight-1 && b[i]!=NbBay+2*NbRight && b[i]>=2*j-1
&& b[i]<=(NbBay-(NbCrane-j)*2) && m>1-NbLeft+2 && b[n]>b[i]+1 && m>j+1 &&
b[n]<b[i]+2*(m-j-1)+2 && b[n]>=2*m-1 && b[n]<=(NbBay-(NbCrane-m)*2)};


 int Maxi[t in bays]=maxl((max(i in TaskOnBay[t])i),0);
int Card[t in bays]=card(TaskOnBay[t]);


  minimize makespan;

  subject to
  {
       presenceOf(actOnRes[83][6])==1;
      presenceOf(actOnRes[81][4])==1;
      presenceOf(actOnRes[82][5])==1;

    forall(i in tasks)
    alternative(acts[i], all(j in cranes:b[i]>=2*j-1 && b[i]<=(NbBay-(NbCrane-
j)*2)) actOnRes[i][j]);



  forall(j in cranes)
  noOverlap(seq[j],dist);




         makespan==max(t in 1..NbBay:Card[t]>0)endOf(acts[Maxi[t]]);


  forall(t in 1..19)
  noOverlap(append(all(i in TaskOnBay[t])acts[i],all(k in
TaskOnBay[t+1])acts[k]));



forall(t in bays:Card[t]>1)
  {
  forall(i in TaskOnBay[t]: i<Maxi[t])
  {
   endBeforeStart(acts[i],acts[i+1]);

  forall(j in cranes:t>=2*j-1 && t<=(NbBay-(NbCrane-j)*2))
    {
```

```
              before(seq[j],actOnRes[i][j],actOnRes[i+1][j]);


    }

  }

}


      forall(i in 1..NbTask,j in cranes: j<NbCrane+NbRight-1 &&
b[i]!=NbBay+2*NbRight && b[i]>=2*j-1 && b[i]<=(NbBay-(NbCrane-j)*2))
    {
              forall(m in cranes: m>j+1 )
              forall(n in t3[i][j][m])
       {

       noOverlap(append(actOnRes[i][j],actOnRes[n][m]));
       }
    }

    forall(i in 1..NbTask,j in cranes:b[i]>=2*j-1 && b[i]<=(NbBay-(NbCrane-
j)*2))
       {

                  forall(m in cranes:j!=NbCrane+NbRight && m!=(1-NbLeft) && m>j
)
                  forall(n in t1[i][j][m])
                  {
                     noOverlap(seqq[i][j][n][m],dist2);
                  }

       }


  };
```

DATA:

```
NbTask=80;
NbBay=20;


NbCrane=3;

NbRight=3;

NbLeft=0;


dummy_ready=[0,0,0,400,400,400];

p = [ 22, 46,166, 70, 99, 98, 21, 99, 10,234,  6, 71,190, 12, 18, 19, 92,
23,107,  5, 24, 33, 19, 20, 41, 22, 82,160, 26,139, 88,104, 65, 62, 39,  4,
75, 42,  5,122,221, 30, 17, 31, 21,158, 28,197,191,  7, 71, 58,106, 26, 75,
12,142, 19, 28, 46, 90,120,  4, 37, 23,190, 50,137,144,195,369, 34, 50,158,
62,150, 26, 26, 10, 61,400,400,400];
```

```
b = [  1,  1,  1,  1,  2,  2,  2,  2,  3,  3,  3,  4,  4,  4,  4,  4,  4,  5,
5,  5,  5,  6,  6,  7,  7,  7,  8,  8,  8,  8,  8,  8,  9,  9,  9,  9,  9,  9,
10, 10, 10, 10, 11, 11, 11, 12, 12, 12, 13, 13, 13, 13, 13, 13, 13, 13, 14,
15, 15, 15, 15, 16, 16, 16, 17, 17, 17, 17, 17, 18, 18, 19, 19, 19, 20, 20,
20, 20, 20, 20,22,24,26];
Phi = {<  1,  2>,<  1,  3>,<  2,  3>,<  1,  4>,<  2,  4>,<  3,  4>,<  5,  6>,<
5,  7>,<  6,  7>,<  5,  8>,<  6,  8>,<  7,  8>,<  9, 10>,<  9, 11>,< 10, 11>,<
12, 13>,< 12, 14>,< 13, 14>,< 12, 15>,< 13, 15>,< 14, 15>,< 12, 16>,< 13,
16>,< 14, 16>,< 15, 16>,< 12, 17>,< 13, 17>,< 14, 17>,< 15, 17>,< 16, 17>,<
18, 19>,< 18, 20>,< 19, 20>,< 18, 21>,< 19, 21>,< 20, 21>,< 22, 23>,< 24,
25>,< 24, 26>,< 25, 26>,< 27, 28>,< 27, 29>,< 28, 29>,< 27, 30>,< 28, 30>,<
29, 30>,< 27, 31>,< 28, 31>,< 29, 31>,< 30, 31>,< 27, 32>,< 28, 32>,< 29,
32>,< 30, 32>,< 31, 32>,< 33, 34>,< 33, 35>,< 34, 35>,< 33, 36>,< 34, 36>,<
35, 36>,< 33, 37>,< 34, 37>,< 35, 37>,< 36, 37>,< 33, 38>,< 34, 38>,< 35,
38>,< 36, 38>,< 37, 38>,< 39, 40>,< 39, 41>,< 40, 41>,< 39, 42>,< 40, 42>,<
41, 42>,< 43, 44>,< 43, 45>,< 44, 45>,< 46, 47>,< 46, 48>,< 47, 48>,< 49,
50>,< 49, 51>,< 50, 51>,< 49, 52>,< 50, 52>,< 51, 52>,< 49, 53>,< 50, 53>,<
51, 53>,< 52, 53>,< 49, 54>,< 50, 54>,< 51, 54>,< 52, 54>,< 53, 54>,< 49,
55>,< 50, 55>,< 51, 55>,< 52, 55>,< 53, 55>,< 54, 55>,< 49, 56>,< 50, 56>,<
51, 56>,< 52, 56>,< 53, 56>,< 54, 56>,< 55, 56>,< 58, 59>,< 58, 60>,< 59,
60>,< 58, 61>,< 59, 61>,< 60, 61>,< 62, 63>,< 62, 64>,< 63, 64>,< 65, 66>,<
65, 67>,< 66, 67>,< 65, 68>,< 66, 68>,< 67, 68>,< 65, 69>,< 66, 69>,< 67,
69>,< 68, 69>,< 70, 71>,< 72, 73>,< 72, 74>,< 73, 74>,< 75, 76>,< 75, 77>,<
76, 77>,< 75, 78>,< 76, 78>,< 77, 78>,< 75, 79>,< 76, 79>,< 77, 79>,< 78,
79>,< 75, 80>,< 76, 80>,< 77, 80>,< 78, 80>,< 79, 80>};


lastt=[      9999  9999  9999  9999  9999 9999   9999  9999  9999  9999
      9999  9999  9999  9999  9999  9999  9999  9999  9999  9999  9999
      9999  9999  9999  9999  9999  9999  9999  9999  9999  9999  9999
      9999  9999  9999  9999  9999  9999  9999  9999  9999  9999  9999
      9999  9999  9999  9999  9999  9999  9999  9999  9999  9999  9999
      9999  9999  9999  9999  9999  9999  9999  9999  9999  9999  9999
      9999  9999  9999  9999  9999  9999  9999  9999  9999  9999  9999
      9999  9999  9999  9999  800   800  800     ];

xx=[
              [     9999 9999   9999 9999   9999  9999  ],
              [     9999 9999   9999 9999   9999  9999  ],

                                   ...

              [     9999 9999   9999 9999   9999  9999  ],
              [     9999 9999   9999 9999   9999  9999  ],
              [     9999 9999   9999 9999   9999  9999  ],
              [     9999 9999   9999 400    9999  9999  ],
              [     9999 9999   9999 9999   400   9999  ],
        [     9999 9999   9999 9999   9999  400   ]]; (93th row)

ttt=[0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      400   400   400];
```

APPENDIX B

NEW SET B RESULTS FOR QCSP

Best results observed in 10 trials are presented.

$size$	ins	LB	Z_{CP}
45	1	754	758*
45	2	759	759*
45	3	754	758**
45	4	783	783**
45	5	754	758*
45	6	765	771**
45	7	795	796**
45	8	754	759*
45	9	797	797*
45	10	790	790**
$Avg.$		770.5	772.9
50	1	754	774*
50	2	768	769**
50	3	768	768**
50	4	754	763**
50	5	754	762*
50	6	754	765*
50	7	775	775**
50	8	753	758**
50	9	797	798*
50	10	754	758**
$Avg.$		763.1	769.0

$size$	ins	LB	Z_{CP}
55	1	754	758*
55	2	773	775**
55	3	777	779*
55	4	754	763
55	5	754	758*
55	6	787	787**
55	7	764	770
55	8	754	765**
55	9	800	800**
55	10	754	757*
$Avg.$		767.1	771.2
60	1	778	779**
60	2	754	756*
60	3	754	758*
60	4	755	765*
60	5	754	759**
60	6	754	758*
60	7	754	779**
60	8	754	758
60	9	784	785*
60	10	799	800**
$Avg.$		764	769.7

$size$	ins	LB	Z_{CP}
65	1	754	758*
65	2	799	799*
65	3	801	802**
65	4	754	758*
65	5	754	758*
65	6	754	757*
65	7	754	757*
65	8	754	756*
65	9	754	758*
65	10	781	781**
$Avg.$		765.9	768.4

$size$	ins	LB	Z_{CP}
70	1	754	759**
70	2	754	765
70	3	754	759**
70	4	754	756**
70	5	754	757*
70	6	760	761*
70	7	754	757**
70	8	754	757**
70	9	753	757*
70	10	772	773**
$Avg.$		756.3	760.2

* Previous best results are repeated.

** New best results are observed.

(note that, Z_{CP} data are independent from initial positions of QCs)

APPENDIX C

NEW SET C RESULTS FOR QCSP

Best results observed in 10 trials are presented.

$size$	ins	LB	Z_{CP}
75	1	1177	1177**
75	2	1003	1013
75	3	1181	1181**
75	4	1103	1103**
75	5	1185	1185**
75	6	1118	1118**
75	7	1192	1192**
75	8	1166	1166**
75	9	1170	1170**
75	10	1188	1188*
$Avg.$		1148.3	1149.3
80	1	1172	1172**
80	2	1003	1021**
80	3	1003	1013*
80	4	1196	1196**
80	5	1029	1029**
80	6	1109	1109**
80	7	1193	1193**
80	8	1011	1013**
80	9	1192	1192*
80	10	1201	1201**
$Avg.$		1110.9	1113.9

$size$	ins	LB	Z_{CP}
85	1	1047	1047**
85	2	1003	1012**
85	3	1024	1025**
85	4	1180	1181**
85	5	1076	1076**
85	6	1003	1010*
85	7	1193	1193**
85	8	1097	1097**
85	9	1003	1010*
85	10	1166	1166*
$Avg.$		1079.2	1081.7
90	1	1003	1009**
90	2	1003	1012**
90	3	1003	1010**
90	4	1057	1057**
90	5	1062	1062*
90	6	1193	1193*
90	7	1105	1105**
90	8	1086	1086**
90	9	1072	1072**
90	10	1049	1049*
$Avg.$		1063.3	1065.5

$size$	ins	LB	Z_{CP}
95	1	1173	1173**
95	2	1086	1086**
95	3	1003	1013**
95	4	1135	1135**
95	5	1137	1137**
95	6	1052	1053**
95	7	1164	1164**
95	8	1003	1009**
95	9	1019	1019*
95	10	1003	1010**
$Avg.$		1077.5	1079.9

$size$	ins	LB	Z_{CP}
100	1	1004	1013**
100	2	1097	1098**
100	3	1100	1100**
100	4	1198	1198**
100	5	1003	1014**
100	6	1135	1135**
100	7	1095	1095**
100	8	1151	1151*
100	9	1003	1009**
100	10	1003	1013**
$Avg.$		1078.9	1082.6

* Previous best results are repeated.

** New best results are observed.

(note that, Z_{CP} data are independent from initial positions of QCs)

APPENDIX D

NEW SET C RESULTS FOR QCSP WITH READY TIMES

Best results observed in 10 trials are presented.

$size$	ins	LB	Z_{CP}
75	1	1302	1315
75	2	1302	1320
75	3	1302	1318
75	4	1302	1314
75	5	1302	1327
75	6	1302	1320
75	7	1302	1323
75	8	1302	1327
75	9	1302	1312
75	10	1302	1313
$Avg.$		1302	1318.9
80	1	1302	1318
80	2	1302	1312
80	3	1302	1314
80	4	1302	1314
80	5	1302	1319
80	6	1302	1326
80	7	1302	1327
80	8	1302	1315
80	9	1302	1317
80	10	1302	1316
$Avg.$		1302	1317.8

$size$	ins	LB	Z_{CP}
85	1	1302	1319
85	2	1302	1325
85	3	1302	1315
85	4	1302	1322
85	5	1302	1320
85	6	1302	1312
85	7	1302	1315
85	8	1302	1326
85	9	1302	1319
85	10	1302	1315
$Avg.$		1302	1318.8
90	1	1302	1314
90	2	1302	1316
90	3	1302	1315
90	4	1302	1329
90	5	1302	1316
90	6	1302	1310
90	7	1302	1338
90	8	1302	1320
90	9	1302	1317
90	10	1302	1323
$Avg.$		1302	1319.8

$size$	ins	LB	Z_{CP}
95	1	1302	1317
95	2	1302	1316
95	3	1302	1332
95	4	1302	1319
95	5	1302	1332
95	6	1302	1314
95	7	1302	1318
95	8	1302	1313
95	9	1302	1321
95	10	1302	1337
$Avg.$		1302	1321.9

$size$	ins	LB	Z_{CP}
100	1	1302	1322
100	2	1302	1327
100	3	1302	1322
100	4	1302	1315
100	5	1302	1326
100	6	1302	1317
100	7	1302	1313
100	8	1302	1325
100	9	1302	1342
100	10	1302	1315
$Avg.$		1302	1322.4

APPENDIX E

NEW SET C RESULTS FOR QCSP WITH TIME WINDOWS

Best results observed in 10 trials are presented.

			$10\ min.$	$30\ min.$
$size$	ins	LB	Z_{CP}	Z_{CP}
80	1	1208	1225	1225
80	2	1208	1248	1246
80	3	1208	1229	1228
80	4	1208	1307	1305
80	5	1208	1245	1241
80	6	1208	1320	1319
80	7	1208	1277	1277
80	8	1208	1283	1283
80	9	1208	1234	1234
80	10	1208	1270	1270
$Avg.$		1208	1263.8	1262.8