# Mathematical Models of Seaside Operations in Container Ports and their Solution

A thesis submitted for the degree of

**Doctor of Philosophy**

Department of Mathematical Sciences
University of Essex

by

**Ghazwan Alsoufi**

July, 2017

*Dedicated to*

My wife for her patience, kindness, help and support.

My dear children Humam, Harith and Reham.

My affectionate parents and all who continually pray for my fortune.

# Acknowledgements

I would like to express my deepest gratitude to Almighty Allah, for giving me strength, courage and the best blessing to complete this work and guide me all the way in my life. This thesis is the culmination of input, work and encouragement of many people who have helped and accompanied me for the four years that I have spent at University of Essex. First of all, my deepest gratitude goes to my family for their endless love and unconditional support. I could not have achieved this without the continuous support and love of my family. Mom, thank you for everything you did. I would like to express my sincerest thanks to my supervisors, Prof. Abdel Salhi and Dr. Xinan Yang for steering me through and pushing me further than I can imagine. Their eloquence, sharpness and great sense of humor have impressed me deeply and made my every meeting and discussion with them always full of joy and unforgettable memories. They have not only provided me their professional expertise and valuable guidance I needed to complete my Ph.D. study, but also been real friends and true mentors. I feel extremely fortunate to work with them. They led me to expand my knowledge, extend my vision and share with me their academic insights. They have always provided me the freedom to choose my research subjects and encouraged me to be involved in various types of research activities. During our academic discussions, I was very impressed by their enthusiasm and ability in thinking extreme while keeping the real life applicability of the research high. I learned a lot from them and I am very grateful to have them as my advisor.

A huge debt of gratitude is extended to my dear wife Sarah ... There are no words that can help me to express my gratitude to you. You were always with me on rainy days and at sleepless nights. You smiled when I smiled and you provided the shoulder when I cried. You kept me going and you made this real. You are my pulse, my blood, my love...other best

Visiting the port of Felixstowe brought us meaningful insights into the fascinating world of shipping in general and more specifically into maritime container transportation. Therefore I would also like to express sincere thanks to Tom Corkhill and Stephen Peck for supplying us with a number of important ideas.

For financial support, many many thanks go to the Iraqi Ministry of Higher Education and Scientific Research for sponsoring my scholarship.

One of the best parts of attending Essex University has been the wonderful opportunity to be surrounded by the many interesting, intellectually stimulating and impressive fellow students. The list includes: Dr. Omar, Dr. Khattab, Dr. Wesam, Dr. Manhal and Dr. Majeed. Friends in Iraq: Mohammed, Ibraham and Ammar, who make my personal life colorful and shining. Finally, specially thanks to all Department of Mathematical Sciences staff for their help and Essex University for their services.

# Declaration

The work in this thesis is based on research carried out in the Department of Mathematical Sciences, University of Essex, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification, and it is all my own work, unless referenced, to the contrary, in the text.

# Abstract

Operational Research and Optimization are fundamental disciplines which, for decades, provided the real-world with tools for solving practical problems. Many such problems arise in container ports. Container terminals are important assets in modern economies. They constitute an important means of distributing goods made overseas to domestic markets in most countries. They are expensive to build and difficult to operate. We describe here some of the main operations which are faced daily by decision makers at those facilities. Decision makers often use Operational Research and Optimization tools to run these operations effectively. In this thesis, we focus on seaside operations which can be divided into three main problems:

- the Berth Allocation Problem (BAP),
- the Quay Crane Assignment Problem (QCAP),
- the Quay Crane Scheduling Problem (QCSP).

Each one of the above is a complex optimization problem in its own right. However, solving them individually without the consideration of the others may lead to overall suboptimal solutions. For this reason we will investigate the pairwise combinations of these problems and their total integration In addition, several important factors that affected on the final solution. The main contributions of this study are modelling and solving of the:

- Robust berth allocation problem (RBAP): a new efficient mathematical model is formulated and a hybrid algorithm based on Branch-and-Cut and the Genetic Algorithm is used to find optimal or near optimal solutions for large scale instances in reasonable time.

- Quay crane assignment and quay crane scheduling problem (QCASP): a new mathematical model is built to simultaneously solve QCASP and a heuristic based on the Genetic Algorithm is developed to find solutions to realistic instances in reasonable time.

- Berth allocation, quay crane assignment and quay crane scheduling problem (BACASP): an aggregate model for all three seaside operations is proposed and to solve realistic instances of the problem, an adapted variant of the Genetic Algorithm is implemented.

Keywords: berth allocation; quay crane assignment; quay crane scheduling; terminal operations; genetic algorithm

# Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| **BACAP** | Berth Allocation and Quay Crane Assignment Problem |
| **BACASP** | Berth Allocation, Quay Crane Assignment and Quay Crane Scheduling Problem |
| **BAP** | Berth Allocation Problem |
| **B&B** | Branch and Bound |
| **B&C** | Branch and Cut |
| **CBC** | Combinatorial Benders' Cuts |
| **GA** | Genetic Algorithm |
| **GDP** | Gross domestic product |
| **GRASP** | Greedy Randomized Adaptive Search Procedure |
| **ILP** | Integer Linear Programming |
| **LP** | Linear Programming |
| **MILP** | Mixed Integer Linear Programming |
| **QCAP** | Quay Crane Assignment Problem |
| **QCASP** | Quay Crane Assignment and Quay Crane Scheduling Problem |
| **QCSP** | Quay Crane Scheduling Problem |
| **RBAP** | Robust Berth Allocation Problem |
| **TEU** | Twenty-foot Equivalent Unit |

# Chapter 1

# Introduction

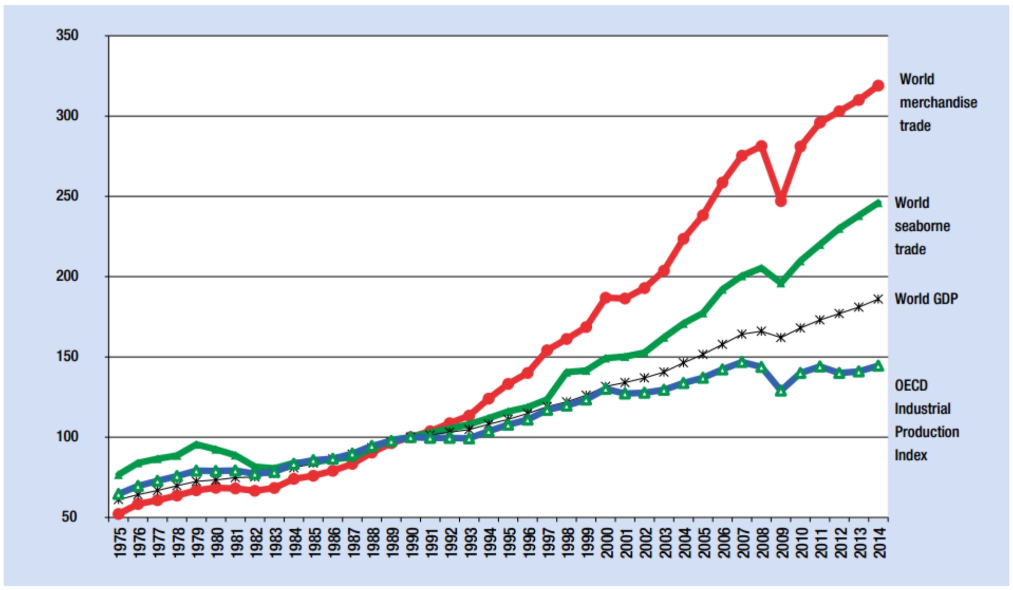
## 1.1 Containerization and Trends in Maritime Logistics: Research Background

The transportation of general cargo underwent a huge change towards the end of the 1950s with the appearance of the first ocean container ships. Prior to this, the handling and transportation of general cargo was a very slow and labour-intensive procedure. The majority of cargo was moved using building pallets and cranes, which hoisted them into the ships holds. This meant that the cargo being transported was susceptible to damage. The introduction of containers meant that cargo could be handled in a quick and simple manner. Containers also helped to facilitate the international transport of goods. The cargo was loaded at the dispatch site and only unpacked when it reached its collection point. Containers come in the form of metal boxes and have uniform measurements.

The five typical lengths include 20ft, 40ft, 45ft, 48ft and 53ft. The latter two sizes are not used for global shipping but are generally used in domestic rail freight transport. Twenty-foot Equivalent Units (TEU) are the standard units that are used to describe container capacity and throughput measurements. One TEU is equivalent to one standard 20ft long, 8ft wide container, although TEUs are rough measures. Different types of goods require specialised containers. For example, frozen goods require temperature-controlled containers, these are referred to as reefers, which is a shortened term for refrigerated containers. Organic product may be shipped in ventilated containers, bulk minerals may be transported in open-top bulktainers and tank containers may be used for bulk liquids or gases.

During the 1990s there was major growth in the area of global container transportation. As a result, container terminal operators and ocean carriers experienced certain changes in the way their businesses were run. In order to meet ever-increasing shipping demands, carriers reacted by introducing new routes and more regular services. As a result, ocean carriers and freight service companies ploughed money into new equipment and state-of-the-art Decision Support Systems (DSS). Container vessels also increased in size to meet this demand, with so-called mega-ships of up to 18,000 TEUs being built [1].

More than 80% of global commodities are transported via the sea, resulting in the extensive and significant growth of international seaborne trade. Figure 1.1 illustrates the developing relationship between the growth in world GDP and seaborne trade between 1975 and 2014, as highlighted in the 2015 report of the United Nations Conference on Trade and Development (UNCTAD) [85].



*Sources:* UNCTAD secretariat, based on OECD Main Economic Indicators, June 2015; United Nations Department of Economic and Social Affairs, 2015; LINK Global Economic Outlook, June 2015; UNCTAD *Review of Maritime Transport*, various issues; WTO, appendix table A1a, World merchandise exports, production and gross domestic product, 1950–2012; WTO press release 739, 14 April 2015.

**Figure 1.1:** *The OECD Industrial Production Index and indices for world GDP, merchandise trade and seaborne shipments (1975 - 2014) (base year 1990 = 100)*

In recent times, the receptivity of trade to GDP growth may have diminished. Nonetheless, the need for ocean carriers and the volumes of international seaborne trade are dictated by worldwide economic expansion and trading requirements. According to initial figures, it has been estimated that the volume of world seaborne shipments in 2014 grew at the same rate as in 2013 namely by 3.4%. Extensions to volumes surpassed 300 million tons, bringing the total to 9.84 billion. According to UNCTAD 2015, this represents approximately four fifths of total world merchandise trade (UNCTAD,2015) [85].

Before the world financial crisis of 2008, global seaborne trade was experiencing a boom. Indeed, by 2007 world seaborne trade amounted to approximately 8.02 billion tons of cargo. It was a different story in 2009, however, when the world GDP decreased by 1.9% and the biggest reduction in global yield was reported since the 1930s. Subsequently, world seaborne trade experienced a major decline in 2009 with a drop in volume of 4.5% [27].

As part of the infrastructure of worldwide containerised shipping, port container terminals act as important links between land transportation and seaborne shipping. There has been significant growth in the area of port container handling as a result of container trade growth. Due to major increases in the transshipment of goods at container terminals, steps have been made to try to expand the throughput of ports. Reportedly, global containerised trade grew by approximately 5.3% in 2014, stretching to 171 million TEUs (UNCTAD,2015) [85]. See Figure 1.2.

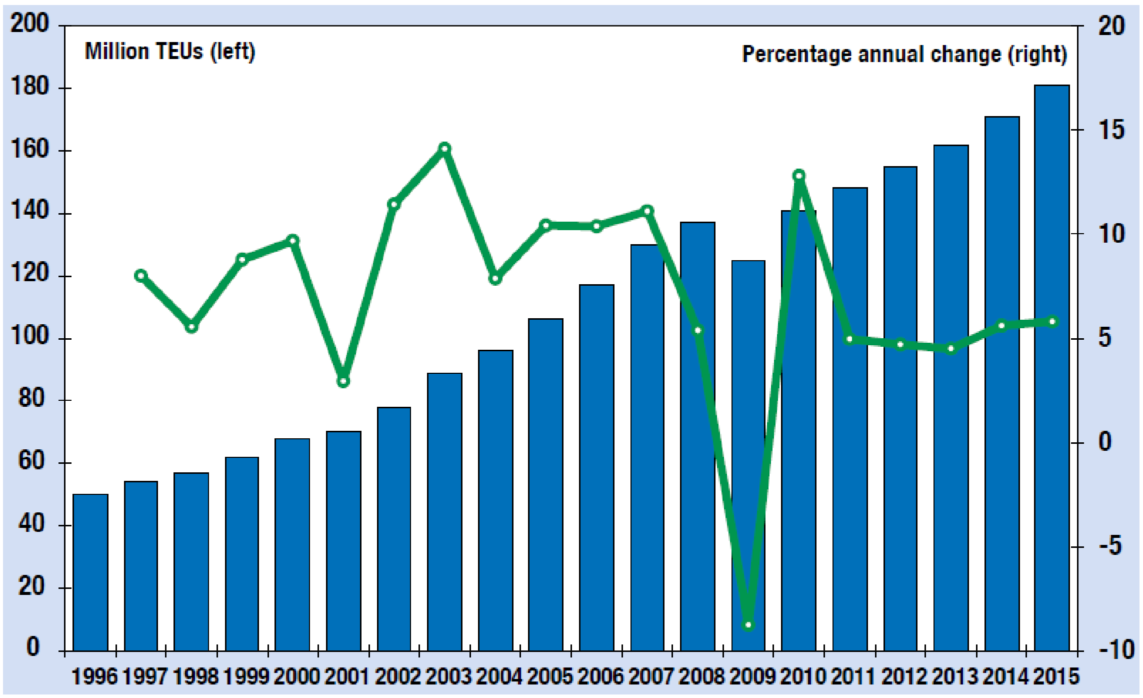


**Figure 1.2:** *Global containerized trade, 1996-2015 (million TEUs and percentage annual change)*

A two-year study into the future of the maritime industry predicted that by the year 2030 global seaborne trade will stand at between 19 and 24 billion tons a year. At present this figure stands at 9 billion tons with approximately 90% of world trade by volume being transported by sea [12].

## 1.2 An Overview of the Container Terminal and Quayside Operations

There are two scarce resources that need to be utilized with care along the quay of port container terminals, berths and quay cranes. The berth is a platform (a linear stretch) where arriving vessels will be moored. Quay cranes are industry-standard equipment for loading and discharging containers to/from vessels. A quay crane is a special type of gantry crane having a large steel frame, which is positioned along the wharf (or quay) alongside a berthed vessel. Quay cranes are very expensive (around 6 million British pound per individual machine) and quay cranes along the berth are mounted on the same track, which forbids them from crossing each other at any instant. Traditionally, there are three key problems needed to be addressed for quayside operations: the Berth Allocation Problem (BAP), the Quay Crane Assignment Problem (QCAP) and the Quay Crane Scheduling Problem (QCSP).

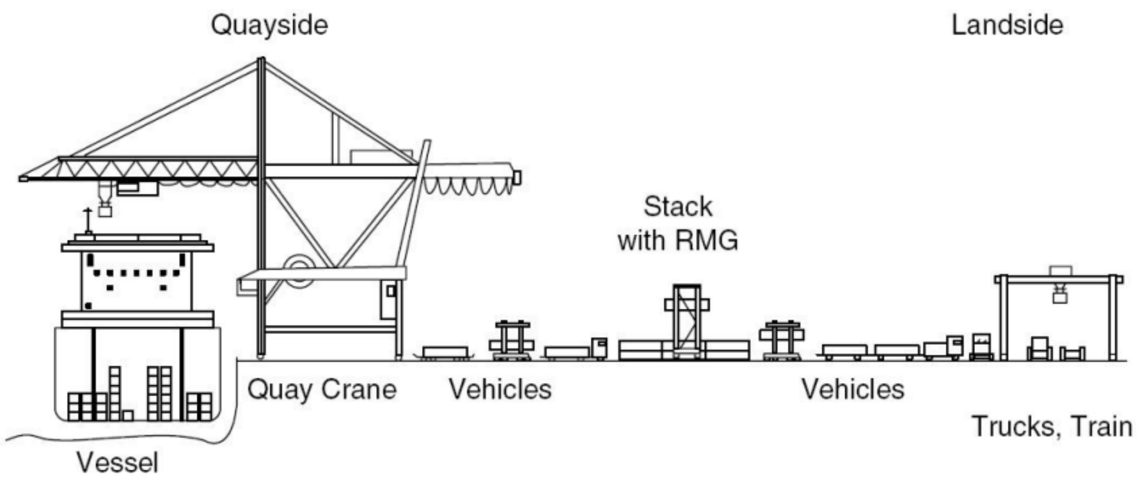


**Figure 1.3:** *Schematic representation of a container terminal (Steenken et al., 2004)*

Once a vessel arrives at the container terminal (port), a number of containers must be unloaded/loaded from/onto the vessel to the storage yard and vice versa. The yard is storage space at the port where containers are stored temporarily until they are transferred to their destinations. How far the berthing position is from the yard storage space impacts on the time required to move a container and therefore the efficiency of processing a vessel. As a result, a “best berthing position” is nominated for every single incoming vessel, which is the closest berth position to the allocated yard storage space. If the best berthing position cannot be guaranteed, a “movement cost” for the additional time and effort for moving the container would have to be paid. To complete the seaside operations, there is a number of quay cranes which will be used to unload/load these containers from the deck of vessels and vice versa.

The problem of BAP determines the berthing times and position for a set of vessels within the planning horizon. This is done by taking into consideration some factors such as the length, the expected arrival/departure time and the processing time of each vessel. The vessel has to be moored for unloading/loading the containers when it arrives at a seaport. For this purpose, a number of berths are constructed at container terminals. No doubt, the utilization of berths has its immediate impact on the overall utilization of the terminal. Accordingly, on the operational level, the decision of allocating berth space for vessels is crucial.

The QCAP, also referred to as the crane split problem [27], is the decision to allocate quay cranes to container vessels with respect to the constraints of quay crane availability and accessibility. There are two reasons preventing the decision makers from assigning too many quay cranes to a vessel. The first reason is the cost of building quay cranes which is very expensive and the second one is the quay crane constraints. However, if the number of quay cranes assigned to a vessel was too low, the vessel will spend more time on handling. Accordingly, the tardiness of the vessel will affect the operators of this vessel and at the same time, the container terminal plan since other vessels will have to wait until this vessel handling finishes. The opposite case is when the number of quay cranes which are assigned to a vessel is more than the number that it is needed, this leads to a high handling cost for this vessel following the high working cost of these quay cranes. Consequently, vessels that need to be handled after this vessel, will moor and wait until the quay cranes become available.

The QCSP, given the fixed number of quay cranes deployed to a certain vessel, the target is to schedule each quay crane of the vessel to a task (bay) so as to perform the tasks in an optimal sequence. With the optimal sequence, the processing time of the vessel can be minimized.

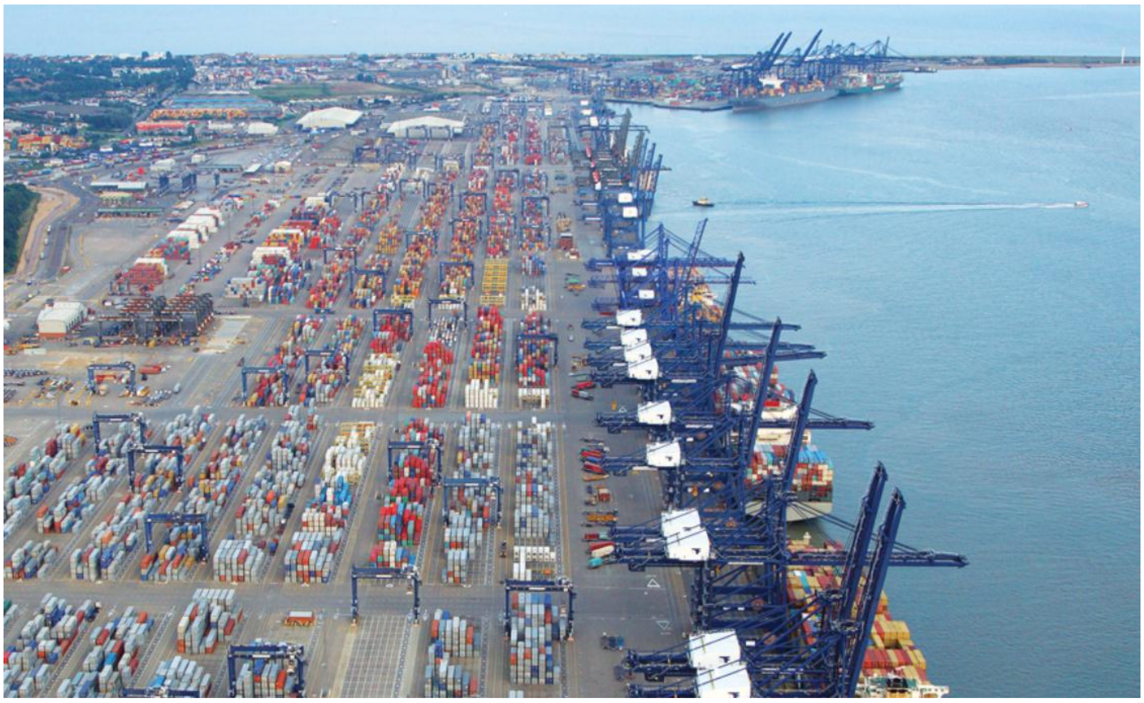


**Figure 1.4:** *A container terminal at the Port of Felixstowe*

## 1.3 Research Objectives, Scope and Organization

Growth in globalization has significantly increased the demand for containerized maritime transport services. As a result, the competition among port container terminals has become acute, which drives the managers in port container terminals to pursue seamless flows of containers through terminals to keep the operational costs as low as possible. To this end, operational research and optimization methods have become very important in the operations management in port container terminals. From the angle of operations research and management science, this thesis aims to design models and devise the corresponding solving methods for the quayside operations in port container terminals. This enables port managers to come up with viable and cost-effective scheduling plans for quayside operation problems in a rapid manner.

As mentioned before, berths and quay cranes are the most crucial resources at a container terminal. They are not only the most expensive equipment, but also the initiates of container flow within the terminal. Efficient and effective utilization of berths and quay cranes is therefore essential for overall port throughput. Many researchers have concentrated on berth and quay crane planning; however, recent trends and changes in maritime logistics, like the introduction of integrating the seaside operations problems of flexible continuous berth structures, have created gaps in the current literature. In this thesis, we hope to fill many of these gaps.

The seaside operations are currently solved individually. Decision makers at the container terminals, however, prefer to solve these operations problem in an aggregate way in order to manage the port near optimality. This thesis presents a comprehensive study on how to formulate an aggregate of quayside operations in container ports models and to develop suitable algorithms to solve them. The dissertation is in six chapters:

Chapter 1 is an introduction which provides the general background of containerization, a brief introduction to quayside operations, the objectives and the scope of the thesis.

Chapter 2 addresses the mathematical background necessary for solving these problems and reviews the previous studies. The literature review is divided into five parts each review on one of BAP, QCSP, BACAP, QCASP and BACASP operations problem.

In Chapter 3, we present a mixed integer programming (MIP) formulation for the robust berth allocation problem (RBAP) which is defined for a container terminal with a continuous berth. Arriving vessels can be assigned to any position of the berth, but we need to add a time buffer between vessels in order to mitigate the uncertainty of real situations. A new weight to estimate this time buffer is proposed, unlike in the previous models that depended on the shipping line's reputation of punctuality or simply added a constant time buffer for each vessel. By implementing the new time buffer we could overcome the delay of a vessel if it was during its traveling to the port or during its handling at the container terminal. The Branch-and-Cut (B&C) method as implemented in CPLEX, is used to find the exact solution for experimental instances and a hybridisation algorithm based on B&C and GA as a solution approach has been developed to solve large scale instances of the RBAP. Computational experiments show that the algorithm is capable to provide high quality solutions in relatively short computation times.

In Chapter 4, we focus on the quay crane assignment and quay crane scheduling operations as QCASP, which is the problem of assigning a variable number of quay cranes to each vessel moored at a container terminal and simultaneously finding the optimal sequence in which to process the vessel. An efficient mixed integer programming model with non-interference among quay cranes constraints has been introduced. A Branch-and-Cut method is used to find the optimal solution for instances of small scale as implemented in CPLEX. In this method, the number of constraints and decision variables increase exponentially with the number of vessels, tasks and quay cranes. Being an exact method B&C cannot achieve an optimal solution in reasonable times. A meta-heuristic algorithm based on GA is proposed to solve large problem instances and to find the optimal or near optimal solution in reasonable amounts of time. Numerical experiments show that the proposed GA is capable of solving QCASP and finding the optimal or near optimal solution efficiently.

In Chapter 5, we introduce the simultaneous berth allocation, quay crane assignment and quay crane scheduling problem (BACASP) formulation which is considered as the most important problem in container terminals. Each one of its individual operations is usually approached sequentially by terminal operators. A new aggregate mathematical model is proposed which combines these quayside operations and solves the resulting problem to optimality when on a small scale. A meta-heuristic based on GA is developed to solve large scale BACASP problems. The experimental results show that the proposed GA is capable of solving BACASP and finding the optimal or near optimal solution in reasonable time. Exact methods like the Combinatorial Benders' Cuts (CBC) algorithm were tested and found that they cannot solve this type of problem.

In Chapter 6, we draw concluding remarks and present future research suggestions and directions related to this research topic. In summary, the research presented in this thesis provides new insights on how to model the seaside operations at container terminals and introduces a set of potent tools to handle the challenging issues arising in this field. To cope with uncertainty, several concerns and suggestions are highlighted as well in this chapter.

# Chapter 2

# Literature Review

There are many studies dealing with the various operations that arise in container terminals. In this chapter, we review literature that addressed the seaside operations. Comprehensive overviews of the three problems (seaside operations) considered here can be found in the papers of Steenken et al. [82], Bierwirth et al. [4] and Bierwirth et al. [5]

## 2.1 The Berth Allocation Problem (BAP)

BAP is the problem of allocating a berthing time and a berthing position to each vessel that arrives at a container terminal. The main objective is to minimize the total vessels turnaround time, which is the sum of the waiting and handling times of each vessel. Handling time is the time it takes to unload/load the containers from/onto a vessel, depending on the distance between the berthing position and the desired berthing position of the vessel. The desired berthing position is that which has the minimum distance from the pre-allocated yard storage in the port, where the containers will be stored until they are transferred to intermediary or final destinations. The decision makers in the container terminal should consider the best berthing time as well as the best berthing position together, so as to minimize the turnover time and the movement cost of the vessel simultaneously.

The BAP with continuous wharf is the normal setting in most modern container ports as it offers more flexibility. Our study also assumes a continuous wharf setting. Under similar problem settings, Li et al. [49] formulated the problem in a "multiple-job-on-one-processor" model, where multiple jobs refer to several vessels and one processor refers to a single berth. A small vessel moored at a berth may share the berth with other vessels if their total length does not exceed the length of the wharf. Lim et al. [51, 52] formulated a BAP and showed it is NP-complete. The authors used a graph theoretic representation to capture the problem succinctly and proposed a heuristic for its solution. Moon [63] proposed a mixed integer linear programming model whose objective is to minimize the tardiness of vessels. A heuristic procedure is suggested for searching a near optimal solution. Goh et al. [24] discussed the methods of modelling BAP and proposed several approaches to solve it such as the Randomized Local Search (RLS), (GA) and Tabu Search (TS). Kim et al. [37] formulated BAP as a mixed integer program and applied a simulated annealing algorithm to find near optimal solutions. Guan et al. [25] developed two inter-related BAP mathematical models the objective functions of which are to minimize the total weighted finishing time of vessels. A tree search procedure was used to solve the first model. This provides a good lower bound which then speeds up the tree search procedure applied to the second one. Imai et al. [31] addressed BAP in a multi-user container terminal and introduced a nonlinear programming model to represent it. The authors presented a heuristic for BAP in a continuous wharf setting. In this paper, they addressed their preference of the flexible berth layout which has become very important especially in busy hub ports where ships of various sizes dock. Wang et al. [89] transformed BAP into a multiple stage decision making problem and a new multiple stage search method, namely the stochastic beam search algorithm, was used to solve it. Lee [46] proposed a neighborhood-search based heuristic to determine the berthing time and berthing position for each incoming vessel to the continuous berth stretch. In their method, the First-Come-First-Served rule, clearance distance between vessels and the possibility of vessel shifting, were considered. Lee et al. [42] studied the continuous and dynamic BAP in order to minimize the total weighted flow time. The authors follow the mathematical model of Guan et al. [25]. Two versions of the Greedy Randomized Adaptive Search Procedures (GRASP) heuristic [71] were developed to find near optimal solutions. Ganji et al. [21] proposed a GA for large scale BAPs to find optimal or near optimal solutions. Note that their model is restricted to find the optimal berthing time. Berthing position is not considered. Also, here, uncertainty is not addressed. Cheong et al. [10] solved BAP by using multi-objective optimization in order to minimize concurrently the three objectives of makespan, waiting time and degree of deviation from a predetermined priority schedule. These three objectives represent the interests of both port and ship operators. Unlike most existing approaches in single-objective optimization, multi-objective evolutionary algorithm (MOEA) incorporates the concept of Pareto optimality which is used when solving the multi-objective optimization problem. de Oliveira et al. [17] proposed a heuristic to solve a continuous case of BAP. This heuristic is based on the application of the clustering search method with the simulated annealing metaheuristic.

In addition to what has been mentioned above, there are few more recent studies which dealt with BAP under uncertainty. Moorthy et al. [64] studied how to design a robust berth template for the special requirements of transshipment hubs. Zhen et al. [93] studied the berth allocation problem under uncertain arrival time or operation time of vessels and proposed a two-stage stochastic programming model. A meta-heuristic approach is proposed for solving the above problem in large-scale realistic environments. Xu et al. [90] studied a robust berth allocation problem. Time buffers are inserted between the vessels occupying the same berthing location to give room to handle uncertain delays. Using total departure delay of vessels as the service measure and the length of time buffer as the robustness measure, the authors formulated robust BAP or RBAP to balance the service level and plan robustness. Based on the properties of the optimal solution, the researchers developed a robust berth scheduling algorithm that integrates simulated annealing and Branch-and-Bound algorithms. This work considers tardiness only in its objective rather than the preferred berthing location which weakens the connections with the yard management problem. It also uses a constant time buffer which is independent of the reputation of the shipping line and the expected processing time. Zhen et al. [92] proposed a proactive strategy for making robust baseline schedules of BAP. A bi-objective nonlinear optimization model for minimizing cost and maximizing robustness is introduced. However, the model is not complete as some constraints used in the model depended on its solution. As a result the author did not even attempt to solve the optimization model. In order to solve large scale instances the authors employed the Squeaky Wheel Optimization (SWO) method [32] to change the vessel inserting sequences. To evaluate their solution approach to large scale instances, they used extensive simulations and justified their findings against some practical criteria.

## 2.2 The Quay Crane Assignment Problem (QCAP)

QCAP is considered after the berth allocation problem is solved, i.e. after the berthing time and position for each vessel are determined. The number of quay cranes that will be assigned to every vessel is very important; thus, QCAP is another challenge to deal with in container terminals. QCAP is to determine the optimum number of quay cranes for each vessel to unload/load containers from/onto vessels so that the throughput of quay cranes is maximized and, at the same time, minimizing the idle time of quay cranes. By choosing a suitable number of quay cranes for each vessel, the handling time of vessels will be minimized, as a result the quay cranes will finish the processing time of each vessel in optimal time. The quay cranes are lined up alongside the quay and they are mounted on the same rail and so cannot cross each other. There are two reasons that prevent the port from increasing the number of quay cranes. The first is the high cost of building these quay cranes and the second reason is the interference between these quay cranes since they move on the same rail and cannot cross each other. Note that solving QCAP results in one of the key decisions for quayside operations. However, as pointed out by Bierwirth and Meisel [4], in practice, QCAP is not difficult. If the load on a mooring vessel is known, which is almost always the case, and the throughput of a quay crane is also known, we are able to work out how many quay cranes would be needed to deal with the load in a given time frame. Therefore, the solution of QCAP is, in practice, a matter of a few arithmetic calculations. It follows that those who deal with QC assignment, with a lot of experience, may solve these problems without relying on a systematic approach. That is why, presumably, the problem has received little attention in academic research. Due to the impact on vessels' handling times, however, crane assignment decisions are involved in some advanced berth planning models. In light of this, in this thesis, there is no intension to provide in-depth study of QCAP alone. It will only be involved when integration models for quayside operations are considered. Note that the research work presented in this thesis involves only deterministic models. If we were to look at it closely, the objective would be to minimise the time to handle the load subject to using as many QC's as necessary, without exceeding the total number of available QC's.

## 2.3 The Quay Crane Scheduling Problem (QCSP)

QCSP is solved to find the optimal sequence of arranging tasks (holds) in order to minimize the finishing time of handling a vessel. Performing the tasks in a good sequence leads to minimize the time of unloading/loading containers from/onto the holds of the vessels by the quay cranes. So, there is a chance for quay crane to move to another task in the same vessel or in the other vessel and so on.

Lim et al. [53,54], proposed two dynamic programming algorithms, proved NP-completeness of the problem and provided heuristics to solve the crane scheduling problem with spatial constraints, a probabilistic tabu search, and a squeaky wheel optimization heuristic. Kim and Park [38] studied QCSP and formulated it as a mixed integer programming. They used Branch-and-Bound (B&B) in conjunction with GRASP to overcome the difficulties of B&B on its own. Moccia et al. [61] formulated QCSP as a vehicle routing problem with additional constraints like the precedence relationships between tasks. CPLEX was used to solve small scale instances and a developed Branch-and-Cut (B&C) algorithm that incorporated several families of valid inequalities to solve large scale instances. Zhu et al. [94] formulated QCSP as an integer programming model. They proved that QCSP is NP-complete and designed a B&B algorithm to solve it to optimality. A simulated annealing meta-heuristic with effective neighbourhood search is designed to find near optimal solutions for larger scale instances. Ng et al. [66] proposed a heuristic to solve QCSP. The heuristic first decomposes the difficult multi-crane scheduling problem into easier subproblems by partitioning the vessel into a set of non-overlapping zones. The resulting subproblems for each possible partition are solved optimally by a simple rule. An effective algorithm for finding tight lower bounds is developed by modifying and enhancing a lower-bounding procedure proposed in the literature. Sammarra et al. [81] proposed a Tabu Search heuristic to minimize the completion time for unloading and loading containers. They considered precedence as well as non-simultaneity between tasks. They observed that QCSP can be decomposed into a routing problem and a scheduling problem. Lee et al. [44,45] proposed a mixed integer linear programming model for QCSP. The authors proved that the QCSP is NP-complete and proposed an approximation algorithm based on GA to obtain a near optimal solution. Lee et al. [43] provided a mixed integer programming model for QCSP with considering the priority of every ship bay. A GA is proposed to solve large scale instances. Bierwirth and Meisel [3] noticed the shortcomings of earlier models of QCSP, and in particular with respect to the quay crane interference avoiding constraints which did not do the job properly. They revised one such model due to Sammarra et al. [81] to take care more appropriately of interference between quay cranes. They also proposed a Unidirectional Schedule (UDS) heuristic when the quay cranes do not change moving direction from their initial position and have identical directions of movement either from upper to lower bays or vice versa. Meisel [57] provided a mathematical formulation of the QCSP and a tree-search-based heuristic solution method. Monaco et al. [62] introduced a mixed integer mathematical model for the QCSP and a heuristic algorithm to get feasible solutions to the large scale of the problem. Furthermore, the authors took into account one-way constraints on the crane movements. Kaveshgar et al. [33] introduced an efficient GA for QCSP. Their algorithm improved the efficiency of GA search by using an initial solution based on the S-LOAD rule and by reducing the number of genes in the chromosomes to reduce search time. Chung and Choy [14] proposed a variant of GA to solve Kim and Park's model of QCSP [38]. Their results compared well with those obtained by most well established algorithms. Nguyen et al. [67] suggested two representations of QCSP one for GA and the other for Genetic Programming (GP) [40]. GA uses permutation to decide the priority of tasks, whereas GP relies on a priority function to calculate it. Unsal et al. [86] proposed a constraint programming approach for solving QCSP. Chung et al. [13] constructed a GA with a workload balancing heuristic, which is capable of considering the loading conditions of different quay cranes during the reassignment of task-to-QC. The idea is modelled as a fuzzy logic controller to guide the mutation rate and mutation mechanism of the GA. As a result, the proposed algorithm does not require any predefined mutation rate. Moreover, the GA can more adequately reassign tasks to QCs according to the QCs loading condition throughout the evolution. Legato et al. [48] proposed a new approach based on the combination of a specialized (B&B) and heuristic algorithms. A cost-effective solution technique that incorporates the local branching method within a refined B&B algorithm is proposed and its effectiveness is assessed by numerical comparisons against the latest algorithm available in literature. Chen et al. [8] studied a special strategy for the quay crane scheduling problem that forces quay cranes to move unidirectionally during the scheduling. In this paper, the researchers proposed a mathematical formulation of the unidirectionally cluster-based quay crane scheduling problem that can be solved by a standard optimization solver. A relaxed formulation is devised as well to obtain a tighter lower bound for the unidirectional cluster-based QCSP. Guo et al. [26] adapted the idea of generalized extremal optimization (GEO) to solve the QCSP with respect to various interference constraints. The resulting GEO is termed the modified GEO. A randomized searching method for neighbouring task-to-QC assignments to an incumbent task-to-QC assignment is developed in executing the modified GEO. In addition, a unidirectional search decoding scheme is employed to transform a task-to-QC assignment to an active quay crane schedule.

## 2.4 The Berth Allocation and Quay Crane Assignment Problem (BACAP)

BACAP is the problem of allocating berthing time and berthing position to the vessels which are waiting in the queue line and simultaneously determinating the number of quay cranes to assign to each one of these vessels. There is a relationship between these two problems and for this reason researchers tend to solve them simultaneously, i.e. the researchers addressed berth allocation problem (BAP) and quay crane assignment problem(QCAP) at the same time.

Meisel et al. [58] introduced a new objective function that aims to reduce the idle time of the quay cranes to solve the integrated problem of BACAP. A heuristic scheduling algorithm based on a priority-rule method is implemented to solve it. Legato et al. [47], addressed QCAP with a predetermined berth position and time following the solution of BAP. They assumed that quay cranes could not move between vessels before all tasks are performed and the processing of the concerned vessels is finished. A mathematical model was presented to determine the optimum number of quay cranes for each vessel that is ready for processing. Meisel and Bierwirth [59] combined BAP and QCAP into BACAP. The proposed problem is formulated taking into account some of the real issues faced by the decision maker at the port. In addition to the mathematical model, they also suggested two meta-heuristic approaches for the problem: the Squeaky Wheel Optimization (SWO), and Tabu Search (TS). Chang et al. [7] studied a dynamic allocation model for berth allocation and quay crane assignments. The problem was preliminarily developed based on the rolling-horizon approach. A hybrid parallel GA, which combined parallel genetic algorithm and heuristic algorithm was employed to solve the propose model. Cheong et al. [11] considered the multi-objective optimization aspect of BACAP; indeed, it involves simultaneous optimization of two highly-coupled container terminal operations. Optimization results show that the multi-objective approach offers the port manager flexibility in selecting a desirable solution for implementation. Liang et al [50] introduced the Quay Crane Dynamic Assignment (QCDA) in berth allocation planning problem (BAP) and formulated a multi-objective mathematical model considering each berth for a container ship and the number of quay cranes moves. In order to solve this QCDA in BAP they proposed a multi-objective hybrid GA with a priority-based encoding method. Raa et al. [74]

presented a mixed integer linear programming for the integrated BAP and QCAP. In their research, the authors took into account vessel priorities, preferred berthing, and handling time. Rodriguez et al. [75] proposed a mixed integer programming model with the objective to minimize the total weighted service time for all vessels. A meta-heuristic based on GA is developed to solve this problem. Yang et al. [91] suggested to deal with BACAP by simultaneously solving BAP and QCAP. They formulated a mathematical model which integrates the BAP constraints of Guan et al. [25] and the QCAP constraints of Legato et al. [47]. The objective function of this model is the combination of the objective functions of the BAP and QCAP models. An evolutionary algorithm was developed to find the solution to this coarse combined problem. Rodriguez et al. [76] considered BAP and QCAP as a dynamic and uncertain scheduling problem. The robustness was introduced by means of time buffer that should be maximized to absorb possible incidences or breakdowns. The problem becomes a multi-objective optimization one with two opposite objectives: minimizing the total time of handling the vessel and maximizing robustness.

## 2.5 The Quay Crane Assignment and Scheduling Problem (QCASP)

The solution of QCASP returns the number of quay cranes to assign to each vessel and the best sequence in which tasks should be carried out by these quay cranes. Combining QCAP and QCSP is very natural given the strong relationship between them. The quay crane scheduling problem requires the number of quay cranes assigned to each vessel in order to decide the sequence in which tasks should be done.

Daganzo [16] addressed quay crane scheduling for multiple vessel arrivals. They proposed both an exact and an approximate solution approach with the objective being to minimise the tardiness of all vessels. Peterkofsky and Daganzo [72] developed a B&B algorithm to solve QCSP. Interference between quay cranes has not been considered in both papers. Tavakkoli et al. [83] studied QCASP. They formulated a mixed integer programming to determine the optimal number of quay cranes for every vessel that will arrive at the terminal and at the same time the optimal sequence in which the tasks should be carried out on the vessel. An evolutionary approach (GA) is suggested to solve large scale instances of this type of problem. Unsal et al. [86] extended their constraint programming model for QCSP to deal with the integrated model that QCASP in which QCs are assigned to vessels and scheduled to work on multiple vessels, based on a berthing plan of vessels. Diabat et al. [18] and Fu et al. [20] also proposed a combined model for QCASP and implemented a variant of GA to solve large scale instances of the problem. Their model allows the movement of quay cranes between vessels while the processing of vessels is ongoing. This is achieved by discretizing the time horizon and using a time index. The interference avoiding constraint is represented simply by the position of quay cranes at each short time interval. This increases the number of variables in the model and potentially makes the problem difficult to solve (curse of dimensionality). Note that time discretization makes the model less accurate. The omission of quay crane travelling time between bays/vessels makes the solution of [20] impractical, especially when the frequency of quay crane movement is high. Precedence and simultaneity constraints are also not considered. Note that the results reported in [20] in Table 3 for instance do not make sense. The General Algebraic

Modeling System (GAMS), commercial software for solving optimization problems, results which presumably are exact, cannot possibly be worse that those returned by GA for most of the problem instances considered.

## 2.6 The Berth Allocation, Quay Crane Assignment and Quay Crane Scheduling Problem (BACASP)

Here, all seaside operations are integrated into a single problem. A few researchers tried to combine BAP, QCAP and QCSP at container terminal in order to find a more realistic solution. The solution of this type of model tries to find the optimal berthing time and position (BAP) and at the same time, to determinate the number of quay cranes that will be assigned (QCAP) and the optimal sequence in which to perform the tasks (QCSP) for each vessel that arrives at a container terminal.

Park et al. [70] were the first to consider BAP and QCAP together. An integer programming model is formulated. The authors suggested a two phase solution to solve the mathematical model. In the first phase, the berthing time and position are determined as well as the number of quay cranes for each vessel that arrives in each time segment. The subgradient optimization technique is applied to obtain a near-optimal solution in the first phase. In the second phase, a detailed schedule for each Quay crane is constructed based on the solution found from the first phase. The dynamic programming technique is applied to solve the problem of the second phase. In [1] Ak et al. present a mixed integer linear programming (MILP) model for the seaside operations. A two phase Tabu Search heuristic is implemented to find solutions for large scale instances. The mathematical model is only suitable for low numbers of vessels between 3 and 4 holds each. The travelling time of quay cranes is not considered and there is no penalty if a vessel is moored at an unsuitable position. The simultaneity and precedence constraints are not includeed in the model. Criteria such as the initial position, the travelling time cost and ready time of quay cranes are ignored. Liu et al. [55] studied the seaside operations and use mixed integer linear programming to minimize the maximum relative tardiness of vessel departures. A good idea presented by Liu was that instead of assuming a relationship function to exist between the processing time of a vessel and the number of quay cranes assigned to it, one should introduce a series of parameters $p_{vj}$ each of which refers to the handling time of vessel $j$ when $v$ quay cranes are assigned to it. However, the integration model proposed needs further improvement since in the Berth-Level Model, the berthing times are revised whereas the berthing positions are taken from the tentative berth plan. Meisel et al. [60] proposed a three-phase integration framework using preprocessing and feedback mechanisms. Phase I estimates productivity rates of quay cranes from the vessel stowage plan. In the Phase II, by using Meisel's model [59] the berth allocation problem and the quay crane assignment problem are solved depending on the productivity rates of quay cranes. Phase III schedules the tasks of each vessel. To adjust the solution for BAP, QCAP, and QCSP combined, a feedback loop is needed. In Rodriguez et al. [77], the focus is on BAP and QCAP problems. However, given that the overall problem is a transshipment one, they also have to deal with loadind/unloading. It is therefore of the combined type as discussed in this section. The QC scheduling aspect is handled through a management of vessel holds. They dealt with the overall problem by applying the Greedy Randomized Adaptive Search Procedure (GRASP) which solves the integrated problem in reasonable times. Their aim is to minimize the total elapsed waiting time to serve all these vessels. Turkogullari et al. [84] integrated the seaside operations into a single MILP model and proposed a decomposition algorithm for it. The authors achieved the model and its solution on the back of a list of 8 assumptions amongst which, discretizing the time horizon and making use of a time index in the mathematical model, are perhaps the most important. The anti-interference constraint is represented simply by the position of quay cranes at every small time interval. This increases the number of variables including the integer ones, which potentially, therefore, increases the solution difficulty of the model. Also they assumed that all the quay cranes are available when assigned to the vessels. They computed the processing times of vessels by dividing the number of containers upon each vessel over all the assigned quay cranes to this vessel. The starting quay cranes point is ignored in their model. The precedence and simultaneity constraints were also not considered. They assumed that all tasks (containers) are located in the same bay, therefore, $\lambda$ the interference parameter between quay cranes and the travelling time of quay cranes are also ignored. In their solution, some vessels have large deviations from their desired berthing positions. Moreover, the advocated solution approach requires many parameters some of which are not easy to estimate and may affect the solution if good values are not used. Examples of such parameters are the minimum and maximum number of quay cranes.

## 2.7 Methodology

The real word is varied and complex. Its study often requires a representation or model of it. Models are in three broad categories: **Iconic**, **analogue** and **symbolic**. Iconic models are scaled down or up representations of the real thing. For instance, paper planes are used to study the aerodynamics of real planes; planetariums are simplified representations of solar systems etc... Analogue models try to capture the properties of the real object or phenomenon. For instance, traffic networks can be represented as a network of pipes with water standing for the traffic and the linked up pipes as the road network. Symbolic models use symbols (variables, constants) and relationships between them such as equations and inequalities to represent reality. These are the sort of models we are concerned with here. They are commonly known as Mathematical Models. They are in two overlapping types: descriptive and explanatory. Explanatory models are what we generally want to build although it may not be always possible; they may require starting with a descriptive model phase first. Optimization or mathematical programming models are explanatory models. They typically consist of one or more objective functions and a set of constraints linking a number of variables and constants. When no constraints are involved, the models are known as unconstrained optimisation problems.

Depending on the type of variables and functions involved, optimisation problems can be, continuous, discrete or both (mixed), linear, or nonlinear. For instance, a linear programming problem (LP) involves continuous variables, a linear objective function, and linear constraints. An integer linear programming problem (ILP) is the same as the LP with the added constraints that the variables can only take integer parts. The models we are concerned with here are of the Mixed Integer Linear Programming (MILP) type.

$$max\ \ c^T x + d^T y \tag{2.1}$$

$$s.t \quad Ax + Gy \leq b \tag{2.2}$$

$$x \geq 0, y \geq 0 \quad and\ \ integer \tag{2.3}$$

where $x$ is a vector of continuous variables, $y$ is a vector of integer variables.

## 2.8 Solution approaches

After building a mathematical model, it must be solved to benefit from what it can offer in terms of variable values and insights into their relationship to support decision making. There are a number of approaches or algorithms to solve optimisation problems: exact and approximate or heuristic. Exact approaches return optimum solutions to within the computer accuracy. Approximate approaches do not guarantee optimality. However, they are often preferred because exact methods are often not expected to work for general cases in acceptable computational times depending on the context. For this, and other reasons, heuristics are often the preferred solution approaches.

### 2.8.1 Exact approaches

**The cutting plane algorithm** The cutting plane algorithm has been proposed by Ralph Gomory in the 1950s as a method for solving integer programming and mixed-integer programming problems. The algorithm relies on adding constraints (cuts) to the formulation and seeking to get as close as possible to the convex hull of the solution set of the original problem. To generate a cut, we can choose the row with a non-integer RHS. Selecting the row (variable) that has the largest fraction coefficient (solution) may reduce the number of iterations and the overall solution time. Assume that the selected row is the row with index $i$, $y_{Bi}$ the basic variable, and $\lfloor x \rfloor$ is the integer number which is just below $x \in R$, then:

$y_{Bi} + \sum a_{ij} y_j = b_i$

$y_{Bi} + \sum (\lfloor a_{ij} \rfloor + f_{ij}) y_j = \lfloor b_i \rfloor + f_i$

$y_{Bi} + \sum \lfloor a_{ij} \rfloor y_j + \sum f_{ij} y_j = \lfloor b_i \rfloor + f_i$

$\sum f_{ij} y_j \geq f_i$

$\sum f_{ij} y_j - \delta = f_i$

$-\sum f_{ij} y_j + \delta = -f_i$

Where:

$f_{ij} = a_{ij} - \lfloor a_{ij} \rfloor$ represents the fraction part of $a_{ij}$ and $0 < f_{ij} < 1$.

$f_i = b_i - \lfloor b_i \rfloor$ represents the fraction part of $b_i$ and $0 < f_i < 1$.

$\delta$ is slack variable.

Cuts are added to the current LP relaxation of the MILP. The dual simplex algorithm is then applied to the current LP relaxation, since the current optimum is made primal infeasible by the cut. A new solution is found and the process is repeated until an optimal integer solution is found. Cutting plane algorithms are often impractical because cuts become thinner and thinner as the fractional parts are smaller and smaller meaning that progress is often limited and the convergence is hardly achieved even after a large number of iterations. The ever smaller fractional parts are what often causes convergence difficulties.

**Branch-and-Bound** The method was first proposed by Land [41] and has become the most commonly used tool for solving integer optimization problems. Branch-and-Bound (B&B) is a partial enumeration method which tries to avoid visiting all possible solutions thorugh a tree search. Like the cutting plane method, it starts by solving the LP relaxation of the given problem using the simplex algorithm for instance. If all the basic variables are integers then the optimal solution is found and the algorithm stops. Else, if at least one of the basic variables is fractional, a branching on this variable is implemented leading to two problems. Assuming that the choosing variable from solving the linear programming is the variable $y_i$ and its integer part is equal to $y_{Bi}$, which can be written as follows:

$y_i = \lfloor y_{Bi} \rfloor + f_i$

Where $0 < f_i < 1$. we know that the value of the variable $y_i$ should be integer, then the new two constraints resulting the branching are:

$y_i \leq y_{Bi}$

$y_i \geq y_{Bi} + 1$

Each one of these problems contains the same original problem plus one of the new constraints due to branching.

$$
\begin{array}{llll}
LP1 & & LP2 & \\
max\ c^Tx + d^Ty & & max\ c^Tx + d^Ty & (2.4) \\
s.t \quad Ax + Gy \leq b & & s.t \quad Ax + Gy \leq b & (2.5) \\
\qquad y_i \leq y_{Bi} & & \qquad y_i \geq y_{Bi} + 1 & (2.6) \\
\qquad x \geq 0, y \geq 0 \quad and\ \ integer & & \qquad x \geq 0, y \geq 0 \quad and\ \ integer & (2.7)
\end{array}
$$

In the worst case, all nodes in the search tree will have to be visited before the solution is found. However, often only a handful are visited as the tree is trimmed through the bounding step. Bounding here means that no improvement of the current objective value is possible. It occurs in three situations.

1. optimality: the solution at the node is integer;
2. infeasibility: the problem at the node is infeasible because of a branching constraint;
3. Value dominance: No subsequent solutions from a node can be better than an already known solution.

These rules keep the growth of the search tree in check making B&B a reliable partial enumeration search approach; hence its popularity.

**Benders decomposition** Benders decomposition [2] is a technique in mathematical programming that allows the solution of mixed integer linear programming problems. The technique is named after Jacques F. Benders. The strategy behind Benders decomposition can be summarized as divide and conquer. That is, in Benders decomposition, the variables of the original problem are divided into two subsets so that a first-stage master problem is solved over the first set of variables, and the values for the second set of variables are determined in a second-stage subproblem for a given first-stage solution. If the subproblem determines that the fixed first-stage decisions are in fact infeasible, then so-called Benders cuts are generated and added to the master problem, which is then re-solved until no cuts can be generated. Benders decomposition adds new constraints as it progresses towards a solution.

$$max\ c^T x + d^T y \tag{2.8}$$

$$s.t \quad Ax + Gy \leq b \tag{2.9}$$

$$x \geq 0, y \geq 0 \quad and\ integer \tag{2.10}$$

Here the aim is to fix integer variables $y$ and exploit the continuous component of the problem as an ordinary LP.

Suppose that $y$ have been fixed and thereby been treated as a parameter, the resulting LP is as follows:

$$max\ c^T x \tag{2.11}$$

$$s.t \quad Ax \leq b - Gy \tag{2.12}$$

$$x \geq 0 \tag{2.13}$$

The dual for this problem is then given by:

$$min\ (b - Gy)^T w \tag{2.14}$$

$$s.t \quad A^T w \geq c \tag{2.15}$$

$$w \geq 0 \tag{2.16}$$

Observe that the feasible region of dual problem remains the same whatever value $y$ takes. Therefore by observing the shape of the polyhedron, the optimal solution to the dual with different $y$ values will be directly available at one of its extreme points (if bounded) or along one of the extreme rays (if unbounded). So that we can write MILP as:

$$max\ z \tag{2.17}$$

$$s.t \quad z \leq d^T y + (b - Gy)^T u \quad \forall u \in U \tag{2.18}$$

$$v(b - Gy)^T \geq 0 \quad \forall v \in V \tag{2.19}$$

$$z \in R, y \geq 0 \quad and\ integer \tag{2.20}$$

where:

U = extreme points of the set of u.

V = extreme directions of the set of u.

The drawback of Benders' decomposition is, potentially, the need for all extreme points and extreme rays of the polyhedron of the original problem, containing its solutions. Polyhedra in high dimensions often have an exponential number of vertices and extreme rays, depending on the number of constraints defining them. This means Benders formulation of the problem will, potentially, have a very large number of constraints since each vertex and extreme ray generate a constraint [65, 69].

**Branch-and-Cut** The Branch-and-Cut (B&C) algorithm [68] combines B&B and the cutting plane algorithm, hence the name. It manages a search tree each node of which represents a linear programming subproblem to be processed, i.e. solved and checked for integrality, and perhaps to be analyzed further.

B&C involves running the B&B algorithm and using cutting planes to tighten the linear programming relaxations. Note that if cuts are only used to tighten the initial LP relaxation, the algorithm is called cut and branch. A branch is the creation of two new nodes from a parent node. Typically, a branch occurs when the bounds on a single variable are modified, with the new bounds remaining in effect for that new node and for any of its descendants. A cut is a constraint added to the model. The purpose of adding any cut is to limit the size of the solution domain for the continuous LP problems represented at the nodes, while not eliminating legal integer solutions. The outcome is thus the reduction of the number of branches required to solve the MIP.

The B&C tree is initialized to contain the root node as the only active node. The root node of the tree represents the entire problem, ignoring all of the explicit integrality requirements. Potential cuts are generated for the root node but, in the interest of keeping the problem size reasonable, not all such cuts are applied to the model immediately. If possible, an incumbent solution (that is, the best known solution that satisfies all the integrality requirements) is established at this point for later use in the algorithm.

**Combinatorial Benders' Cuts (CBC)** CBC is used to find exact optimal solutions. It was proposed by Hooker in [29], where he derives the so called Benders' cuts from a minimal set of inconsistencies. Codato and Fischetti [15] extended Hooker's method to solve mixed integer programming with special structure. The approach generates a set of cuts which removes the drawback of the big-M method. CBC is closely related to the Benders' Decomposition [2].

### 2.8.2 Meta-heuristics

Meta-heuristics are higher level heuristics, i.e. they may rely on other heuristics to solve problems. A heuristic is an algorithm based on rules that work but with no guarantee. Their strength lies in not making assumptions on the problems. For instance, they do not rely on gradient information to optimise functions. They are popular because they work on real world problems which are often difficult, i.e. they require an unreasonable computational time for their solution; they are known in the complexity jargon as NP-hard problems [1] For this reason, approximate solution approaches or meta-heuristics are often preferred. Note that the problems we are concerned with throughout this thesis, are of the NP-Hard type. It is therefore relevant that we present here the most popular meta-heuristic approaches for solving them.

**Simulated Annealing (SA)** Simulated annealing is a well established technique for combinatorial optimization problems. The idea is to achieve a goal state without reaching it too fast. In metallurgy, for example, the process of hardening steel requires specially timed heating and cooling to make the iron and carbon atoms settle just right. In mathematical search algorithms, we want to focus on promising solutions without ignoring better solutions we might find later. In other words, we want to reduce error to the global minima without getting stuck in less successful local minima. The algorithm was first introduced by Armen G. Khachaturyan et al. [35,36] and then developed by Scott Kirkpatrick et al. [39].

[1]A problem belongs to the class NP if a solution to it can be verified quickly, i.e. in polynomial time. A problem is NP-hard if the entire class of NP problems polynomially reduce to it [73]. A problem *A* reduce to *B* if it can be constructed for any instance of *A* an equivalent instance of *B*. it is believed that NP-hard problems are intractable, i.e. that there is no efficient algorithm to solve them. To prove this is the most important challenge in computational theory. See [22] for an introduction to complexity theory.

SA is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a meta-heuristic to approximate global optimization in a large search space. It is often used when the search space is discrete (e.g., all tours that visit a given set of cities). For problems where finding an approximate global optimum is more important than finding a precise local optimum in a fixed amount of time, simulated annealing may be preferable to alternatives such as gradient descent.

**Tabu Search** Tabu Search (TS), created by Fred W. Glover [23] is a meta-heuristic search method employing local search methods used in mathematical optimization. Local (neighborhood) searches take a potential solution to a problem and check its immediate neighbors (that is, solutions that are similar except for one or two minor details) in the hope of finding an improved solution. Local search methods have a tendency to become stuck in suboptimal regions or on plateaux where many solutions are equally fit. Tabu search enhances the performance of local search by relaxing its basic rule. First, at each step worsening moves can be accepted if no improving move is available (like when the search is stuck at a strict local minimum). In addition, prohibitions (henceforth the term tabu) are introduced to discourage the search from coming back to previously-visited solutions. The implementation of tabu search uses memory structures that describe the visited solutions or user-provided sets of rules. If a potential solution has been previously visited within a certain short-term period or if it has violated a rule, it is marked as ”tabu” (forbidden) so that the algorithm does not consider that possibility repeatedly. Briefly, the method explores solution spaces by moving at each iteration to the best non-tabu neighbor of the current solution. In order to avoid cycling, solutions that were recently visited are made tabu for a number of iterations [1].

**Particle Swarm Optimization (PSO)** PSO is originally attributed to Kennedy and Shi [34, 78] and was first intended for simulating social behaviour as a stylized representation of the movement of organisms in a bird flock or fish school. The algorithm was simplified and it was observed to be performing optimization. PSO is a meta-heuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. More specifically, PSO does not use the gradient of the problem being optimized, which means PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods. PSO is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position, but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

**Plant Propagation Algorithm** The above listed approaches are very well established and have shown their worth over decades of use on a variety of applications. They are, however, not the only ones. An exhaustive listing of all heuristics and meta-heuristics is beyond the scope of this thesis. However, it will perhaps benefit the potential reader to be aware at least of the type of Nature-Inspired heuristics that are being introduced to handle ever increasing in size and complexity problems. One such heuristic is the Plant Propagation Algorithm or PPA introduced by Salhi and Fraga [79]. It emulates the way plants and in particular the strawberry plant propagate. It starts with a population of plants uniformly randomly planted in the search space. These plants are then ranked according to the values that their positions give to the objective function of the given optimisation problem. The intensity of their propagation is determined by their scores on the objective function. If they score well, this is interpreted as being in a good spot of the land (search space); if on the other hand they score low, then they are considered as being in a bad or poor spot. Those in good spots propagate by generating many short runners leading to new plants; those in bad spots generate few but long runners. Note that the notion of long runner implements the idea of running away from the poor areas or exploring further the search space, while short runners exploit the good spots and refine the search to find local optima. Exploitation and exploration are the two key ingredients in any successful global optimisation.

## 2.9 Genetic Algorithm

In computer science and operations research, the Genetic Algorithm (GA) is a meta-heuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). It was proposed by Holland [28]. Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators. It searches for the optimum by maintaining a population of solutions that are then updated from generation to generation using a number of possible genetic operators such as Crossover, Mutation, and Reproduction. Over successive generations the population evolves towards an optimal solution. GA is a well established meta-heuristic which has been shown to be very effective on combinatorial optimisation problems. GA is particularly effective on difficult problems referred to as being NP-Hard. QCASP has been shown to be NP-hard [22,45,83].

In order to solve large scale problems in acceptable time and to overcome the difficulty of the Branch-and-Cut (B&C) method, a GA is used to find optimal or near optimal solutions. The choice of GA here as the approximate solution approach is dictated by its being well established and reliable. Potential users may adopt it readily. It is an adaptive heuristic method based on natural evolution ideas. It repeatedly modifies a population of solutions, selecting individuals from the current population to be parents which then produce children which form the individuals of the next generation. Over successive generations the population evolves towards an optimal solution. The processes involved in GA are outlined below [33].

1. Generate an initial population: individual solutions (chromosomes) are created randomly to form an initial population.

2. Evaluate the fitness of each individual: choice of parents of new individuals for the new generations is biased toward individuals with good fitness values.

3. Create children: generate new individuals using genetic operators such as crossover, mutation and reproduction.

4. Generate new population: replace the worst individuals in the population with better new ones.

5. Stopping: The process is repeated until stopping criteria are met; these may include the

specified maximum number of generations or time limit, high enough fitness etc...

### 2.9.1 Solution representation: chromosome

The initial and most important step of the GA implementation is the solution representation or chromosome design. GA starts with a randomly generated population of solutions. Each solution is called a chromosome and consists of a sequence of genes.

### 2.9.2 Fitness function

A fitness function is a particular type of objective function that is used to summarise, as a single figure of merit, how close a given design solution is to achieving the set aims. To validate solutions, if anyone of the problem's conditions is violated or any combination are violated, the generated chromosomes are discarded by adding a high penalty to their fitness values.

### 2.9.3 Generating next populations

From the initial randomly generated population, subsequent generations of children, i.e. new populations, must be created. This is achieved by using genetic operators such as crossover, mutation and reproduction (copying of individuals unmodified into subsequent populations).

**Selection process** The selection process picks chromosomes from the current population to be parents to new individuals (children/solutions) of the new population created using one of many genetic operators as listed above. To give priority to the best chromosomes to pass their genes into the next generation, the fitness proportionate selection approach is implemented using a roulette wheel. High fitness individuals/chromosomes/solutions have high probability to be selected to contribute to the next population. In other words they are likely to pass their genes into the next generation.

**Crossover operator** Also called the recombination operator, it is one of the operators of GA used to generate the next population from the present one. It is applied to the chromosomes of a randomly selected couple of individuals (parents). The recombining of their genes results in a couple of new chromosomes (children).

**Mutation operator** Mutation is another operator of GA which contributes to the generation of the next population. To prevent the population getting trapped in a local solution, the mutation operator is used to enable the GA to escape and explore the search space globally by changing one or more genes.

**Reproduction** The reproduction operator consists in copying current individuals as they are into the new population.

### 2.9.4 Stopping criteria

Two stopping criteria are used in the GA: the maximum number of generations, and the the number of generations without any improvement in the best solution found so far; these numbers are pre-set. It is well know that determining good values for a given problem type is difficult. The common practice is to use experimentation to find good default values. We have adopted the same approach. There is a trade-off between increasing the number of generation and the computational time. More generations generally mean a longer computational time.

## 2.10 Summary

In this chapter, we have reviewed the literature concerned with the quayside operations individually or simultaneously dealt with. There are many commercial optimization solvers that can handle these models, such as CPLEX, GUROBI, MINOS, XPRESS - MP, GNU-Solver, NAG Library and LINGO among others. CPLEX and GUROBI are considered the industry standards for optimisation nowadays [56] . However, we used CPLEX as a solver to find exact solution. CPLEX uses Branch-and-Cut search when solving mixed integer programming (MIP) models. CPLEX Optimizer is accessible through independent modeling systems such as AIMMS, AMPL, GAMS, OptimJ and TOMLAB. Also in this chapter, we reviewed some exact as well as meta-heuristics that are used in practice such as GA, TS, SA and PSO. GA has been successfully adopted to find the optimal or near optimal solution for seaside problems compared with other meta-heuristics. GA is widely used to solve these problems.

# Chapter 3

# Robust berth allocation problem

## 3.1 Introduction

Container transportation is at the heart of the import and export of goods. The efficiency with which containers are unloaded/loaded from/onto vessels and are handled/processed, is affected by the time and place at which the vessel is moored along a berth. Minimizing the processing cost of handling a set of vessels by finding the best berthing time and berthing position is the well known Berth Allocation Problem (BAP). A vessel is moored at a good time in order to minimize the total vessel turnaround (or service) time, which is the sum of total handling and waiting times of vessels. If the vessel is not moored at its preferred position, the distance to transfer containers from it to the storage area is not minimal. Therefore, a penalty may have to be incurred by the port as well as the vessel's operator, for failing to handle the ship in time, and causing delay which may have a knock on effect on succeeding operations in this port and/or elsewhere.

The deterministic form of BAP is the one often solved in practice. However, a number of uncertain factors and unexpected events such as the deviation in arrival time and in the operations time of the initial baseline schedule affect BAP. If these uncertainties are ignored when drawing berth allocation plans, last minute scrambling and changes of plans, may ensue. Port managers, therefore, try to protect the initial baseline schedule from the adverse effects of possible disruptions. Hence, there is a need for robust schedules which are not perturbed by variance in key parameters. This is the aim of this study.

There are three types of the berth allocation problem according to Imai [31]:

1-Discrete layout: the quay is divided into a number of segments, called berths. Each one of these segments can only handle one vessel at a time.

2-Continuous layout: the quay in this type is a long segment, there is no partition. Each vessel that arrives to this quay can moor at any position within the boundaries of the quay.

3-Hybrid layout: like the discrete berth, the quay is divided into partitions (berths) but large size vessels can be moored at more than one berth and two or more small vessels can be moored at one berth.

According to Imai [30] there are two distinct cases for the arrival time of vessels:

1- Static arrival: all vessels are waiting at the port and there is no arrival time.

2- Dynamic arrival: the arrival time for each vessel is given. In this case, the vessel cannot be moored before the expected arrival time.

This chapter describes a mixed integer programming model for the continuous berth allocation problem under uncertain conditions, depending on the reputation in punctuality of the operator and the length of processing time of each vessel to find a robust berth allocation plan. In this chapter we formulate an optimization model the objective of which consists of the operations cost related to where the vessel is to be moored and a penalty cost to pay for the potential waiting time. A robust model is achieved by inserting a time buffer between the exact processing time of vessels, to add more flexibility to the final berth allocation plan. Unlike in previous studies where people mainly make the time buffers depend on the shipping line's reputation of punctuality, here we also consider the expected processing time of a vessel when inserting time buffers between vessels. Although a good reputation in punctuality helps the decision maker at the port to estimate the delay in arrival time, the potential delay in processing of vessels depends on how many tasks have to be tackled on the vessel. For example, if there are two vessels that are owned by the same shipping line with one carrying 50 containers to unload and the other 500, then obviously the one with fewer containers (smaller workload; shorter expected processing time) has much lower chances to depart late from the port compared with the one with higher workload. For this reason, a new weight for each vessel is added which represents the product of the proportion of the processing time of a vessel over the sum of the processing time of all vessels under consideration and the number of vessels. In practice, this weight can be seen as a measure of the possibility of having adverse events (faulty machinery such as quay cranes, trucks, yard cranes etc. breaking down) occurring during the processing time of the vessel.

The model suggested here is suitable for robust berth allocation in realistic situations. The contribution of this chapter is two fold:

1. formulation of an efficient mathematical model with more advanced weight parameters for robustness;

2. design and application of a hybrid meta-heuristic based on B&C and GA to improve performance on realistic instances of the problem.

The rest of the chapter is organised as follows. Section 3.2 presents the proposed mathematical model. A numerical example to illustrate the mathematical model is presented in Section 3.3. In Section 3.4, a variant of the genetic algorithm is described for Robust BAP or RBAP; this variant is then combined with B&C to provide a more effective solution approach. Section 3.5 records the comparison results between B&C as implemented in CPLEX, and the hybrid meta-heuristic. Finally, Section 3.6 gives a summary of findings.

## 3.2 Mathematical formulation

The novelty of the model described here is that, unlike previous berth allocation/planning models, as mentioned earlier, its solutions provide robust berth plans. By this, we mean that the solutions mitigate the uncertainty that is often experienced at the level of earlier parts. Note that the solution to the model is in terms of optimum berthing time, berthing position and, the optimum time buffer between the berthing times of vessels. Moreover, we can use this model to solve the traditional BAP by making the value of $\lambda$, the robustness parameter, equal to zero. In the following a number of aspects of this model are given.

### 3.2.1 Assumptions

Although several assumptions have to be satisfied in order to apply the model, they are all realistic. Given a set $V$ of vessels referred to as $v$, $v_i$ or $v_j$ for any vessel, the $i^{th}$ or the $j^{th}$ vessels respectively, we assume that:

1- Every segment of the continuous wharf can handle only one vessel at a time;

2- There is a safety distance between each pair of adjacent vessels;

3- Once the processing of a vessel starts the vessel will only leave after its processing has finished;

4- A vessel can be handled at any place of the wharf. The berthing position depends on its arrival time and the availability of wharf space.

### 3.2.2 Parameters

| | |
|---|---|
| $W$ | Length of the wharf. |
| $A_v$ | Estimated arrival time for vessel $v$. |
| $h_v$ | Estimated operation/processing time to handle vessel $v$. |
| $L_v$ | Length of vessel $v$. |
| $b_v$ | Desired berthing position of vessel $v$; it is determined by the position of yard storage areas allocated to vessel $v$. |
| $C_{1v}$ | Tardiness cost of vessel $v$. |
| $C_{2v}$ | Distance cost of vessel $v$ for mooring away from $b_v$. |
| $IN_v$ | Discrepancy in arrival time of vessel $v$. |
| $PP_v$ | Product of the proportion of the processing time for vessel $v$ over the processing time of all vessels and the number of vessels. |
| $R_v$ | Value of ($IN_v$ + $PP_v$) of vessel $v$, rounded to the nearest integer. |
| $M$ | Arbitrary large positive number. |

### 3.2.3 Binary decision variables

$$\delta_{v_iv_j} = \begin{cases} 1 & \text{if the processing of vessel } v_j \text{ starts later than the finishing time of vessel } v_i. \\ 0 & \text{otherwise} \end{cases}$$

$$\sigma_{v_iv_j} = \begin{cases} 1 & \text{if vessel } v_i \text{ is located nearer one end of the wharf than vessel } v_j. \\ 0 & \text{otherwise.} \end{cases}$$

$$\xi_{v_iv_j} = \begin{cases} 1 & \text{if vessel } v_j \text{ occupies part of the berthing position of vessel } v_i. \\ 0 & \text{otherwise.} \end{cases}$$

$$\zeta_{v_iv_j} = \begin{cases} 1 & \text{if vessel } v_j \text{ occupies part of the berthing position of vessel } v_i \text{ and starts} \\ & \text{ later than } v_i. \\ 0 & \text{otherwise.} \end{cases}$$

### 3.2.4 Continuous decision variables

- $T_v$ Berthing time of vessel $v$.
- $P_v$ Berthing position of vessel $v$.
- $D_v$ Dummy berthing time of vessel $v$. This is a modelling device that allows a smooth solution process.

### 3.2.5 The mathematical model

The explicit model we built for berth allocation is as follows.

$$\min Z = \sum_{v=1}^{V} C_{1v}(T_v + h_v - A_v) + \sum_{v=1}^{V} C_{2v}|P_v - b_v| \tag{3.1}$$

s.t

$$D_{v_i} + h_{v_i} \leq D_{v_j} + M(1 - \delta_{v_i v_j}) \qquad \forall v_i, v_j; v_i \neq v_j \tag{3.2}$$

$$P_{v_i} + L_{v_i} \leq P_{v_j} + M(1 - \sigma_{v_i v_j}) \qquad \forall v_i, v_j; v_i \neq v_j \tag{3.3}$$

$$\sigma_{v_i v_j} + \sigma_{v_j v_i} + \delta_{v_i v_j} + \delta_{v_j v_i} \geq 1 \qquad \forall v_i, v_j; v_i \neq v_j \tag{3.4}$$

$$T_v \geq A_v \qquad \forall v \tag{3.5}$$

$$0 \leq P_v + L_v \leq W \qquad \forall v \tag{3.6}$$

$$D_v \geq T_v \qquad \forall v \tag{3.7}$$

$$\xi_{v_i v_j} = 1 - (\sigma_{v_i v_j} + \sigma_{v_j v_i}) \qquad \forall v_i, v_j; v_i \neq v_j \tag{3.8}$$

$$\zeta_{v_i v_j} \geq \delta_{v_i v_j} + \xi_{v_j v_i} - 1 \qquad \forall v_i, v_j; v_i \neq v_j \tag{3.9}$$

$$T_{v_i} + h_{v_i} + \lambda R_{v_i} \zeta_{v_i v_j} \leq T_{v_j} + M(1 - \zeta_{v_i v_j}) \qquad \forall v_i, v_j; v_i \neq v_j \tag{3.10}$$

$$\delta_{v_i v_j}, \sigma_{v_i v_j}, \zeta_{v_i v_j}, \xi_{v_i v_j} \in \{0, 1\} \qquad \forall v_i, v_j \tag{3.11}$$

$$P_{v_i}, T_{v_i}, D_{v_i} \geq 0 \qquad \forall v_i \tag{3.12}$$

Note that the objective function to minimise is made up of the cost of tardiness which is represented by the term $\sum_{v=1}^{V} C_{1v}(T_v + h_v - A_v)$ and the cost of the vessel being moored at an undesired berthing position represented by the term $\sum_{v=1}^{V} C_{2v}|p_v - b_v|$.

In constraints (3.2), $\delta_{v_i v_j}$ is defined as follows. $\delta_{v_i v_j} = 1$ if the finishing time of vessel $i$ is less than or equal to the dummy berthing time of vessel *j*; 0 if the finishing time of vessel *i* is greater than the dummy berthing time of vessel *j*. Figures 3.1 and 3.2 illustrate how $\delta_{v_iv_j}$ is computed:

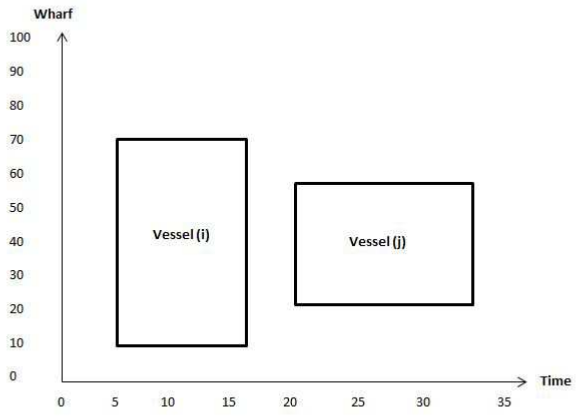


**Figure 3.1:** *Illustration of no overlap in time between vessels i and j*

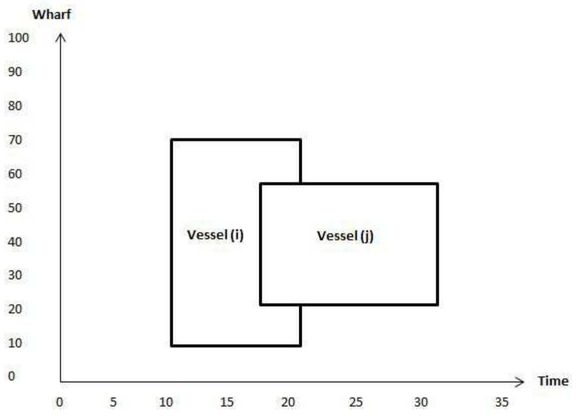


**Figure 3.2:** *Illustration of overlap in time between vessels i and j*

$\delta_{v_iv_j}$ in Figure 3.1 equals 1 because $F_{v_i} \leq T_{v_j}$. Whereas, $\delta_{v_jv_i}$ in the same figure equal 0 because $F_{v_j} > T_{v_i}$. $\delta_{v_iv_j}$ in Figure 3.2 equals 0 because $F_{v_i} > T_{v_j}$ and $\delta_{v_jv_i}$ in the same figure also equals 0 because $F_{v_j} > T_{v_i}$. This means there is an overlap in the time between these two vessels.

In constraints (3.3), $\sigma_{v_iv_j}$ is defined as follows. $\sigma_{v_iv_j} = 1$ if the berthing position of vessel *i* plus the length of vessel *i* is less than or equal to the berthing position of vessel *j*; 0 if the berthing position of vessel *i* plus the length of vessel *i* greater than the berthing position of vessel *j*. Figures 3.3 and 3.4 illustrate how $\sigma_{v_iv_j}$ is computed: $\sigma_{v_iv_j}$ in Figure 3.3 equals 1 because $P_{v_i} + L_{v_i} \leq P_{v_j}$. Whereas, $\sigma_{v_jv_i}$ in the same figure equals 0 because $P_{v_j} + L_{v_j} \geq P_{v_i}$. $\sigma_{v_iv_j}$ in Figure 3.4 equals 0 because $P_{v_i} + L_{v_i} > P_{v_j}$ and $\sigma_{v_jv_i}$ in the same figure equals 0 because $P_{v_j} + L_{v_j} \geq P_{v_i}$. That means there is an overlap in the location between the two vessels.

Constraints (3.4) ensure that overlaps amongst vessels do not occur in both dimensions

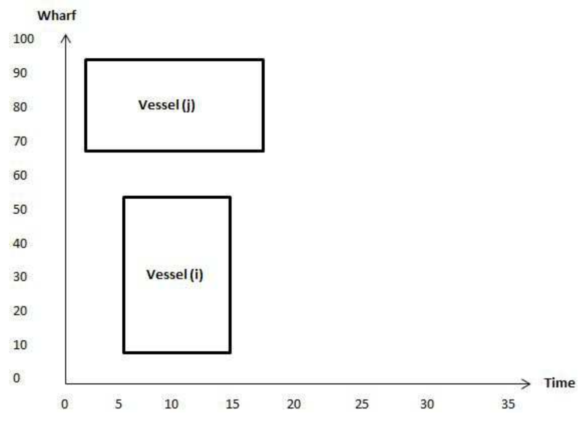


**Figure 3.3:** *Illustration of no overlap in location between vessels i and j*

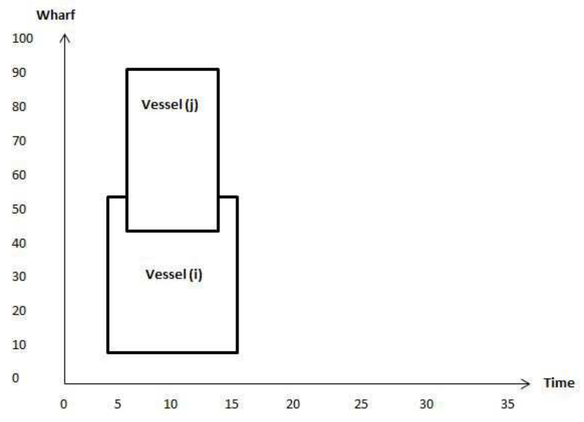


**Figure 3.4:** *Illustration of overlap in location between vessels i and j*

(time and location) depending on $\delta_{v_iv_j}$ and $\sigma_{v_iv_j}$. Constraints (3.5) guarantee that the vessels cannot moor before their arrivals. Constraints (3.6) imply that the berthing position plus the length of the vessel cannot exceed the range of the wharf. Constraints (3.7) guarantee that the dummy berthing times of vessels cannot be less than their berthing times. Constraints (3.8) make variable $\xi_{v_iv_j}$ take value 1 if the vessel $v_j$ occupies part of the berthing position of vessel $v_i$. Constraints (3.9) make variable $\zeta_{v_iv_j}$ take value 1 if vessel $v_j$ occupies part of the berthing position of vessel $v_i$ and starts later than vessel $v_i$. Constraints (3.10) determine the berthing time for each vessel after adding the time buffer between vessels depending on the value of $\zeta_{v_iv_j}$. $R_v$ is the time buffer between the finishing time of vessel $i$ and the berthing time of vessel $j$. The decision maker at the port will select the value of $\lambda$ according to his preferred aspect of the final plan, cost-effective or robust. A small $\lambda$ means a cost-effective berth plan, while a large $\lambda$ means a robust plan. The length of the time buffer between the vessel and its immediately preceding vessels can be computed by multiplying the associated value of the robustness parameter, $\lambda R_{v_i}\zeta_{v_iv_j}$. The constraints set (3.10) work in such a way that, if the value of the integer decision variable $\zeta_{v_iv_j}$ is equal to zero, then we do not need to add any time buffer. Otherwise we need to increase the gap between these two vessels $v_i$ and $v_j$; the value of the time buffer is the value of $\lambda$ times $R_{v_i}$.

The rough size of the above model is as follows:

The number of binary variables is equal to $4V^2 - 4V$.

The number of integer variables is equal to $3V$.

The number of constraints is equal to $6V^2 - 3V$.

## 3.3 Numerical example

Here, we apply B&C as implemented in CPLEX to a small instance of the above model involving six vessels. The rest of the data is given in Table 3.1.

**Table 3.1:** *Input data of example 3.1*

| | | | | | | |
|---|---|---|---|---|---|---|
| Arrival time | 0 | 3 | 13 | 23 | 25 | 30 |
| Estimated processing time | 15 | 20 | 8 | 5 | 10 | 12 |
| Length of vessel | 5 | 3 | 4 | 6 | 4 | 5 |
| Preferred position | 5 | 1 | 4 | 3 | 6 | 0 |
| Tardiness cost | 1 | 1 | 1 | 1 | 1 | 1 |
| Distance cost | 1 | 1 | 1 | 1 | 1 | 1 |
| Instability in arrival | 0.4 | 1.7 | 0.7 | 0.4 | 0.9 | 1.0 |
| proportion of the processing time | 1.3 | 1.7 | 0.7 | 0.4 | 0.9 | 1.0 |

The solution returned by CPLEX consists of the berthing time and position of each vessel with appropriate time buffer between them to give robustness to the plan represented in Figures 3.5 , 3.6 and 3.7. Clearly, the decision maker can use different values of $\lambda$ to affect the property of the final solution sought.

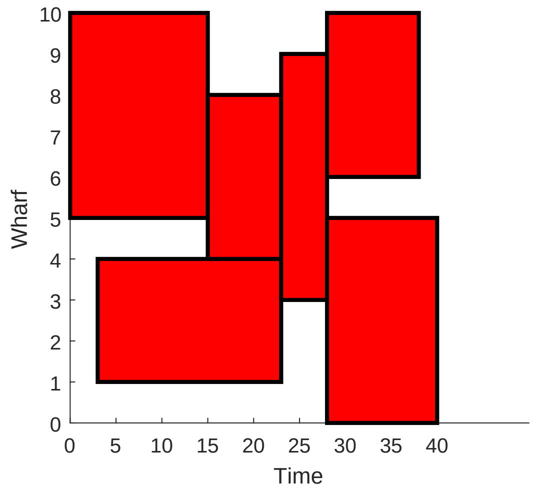


**Figure 3.5:** *Berthing plan when λ=0; Obj.Fun = 75 (0 3 15 23 28 30 5 1 4 3 6 0)*

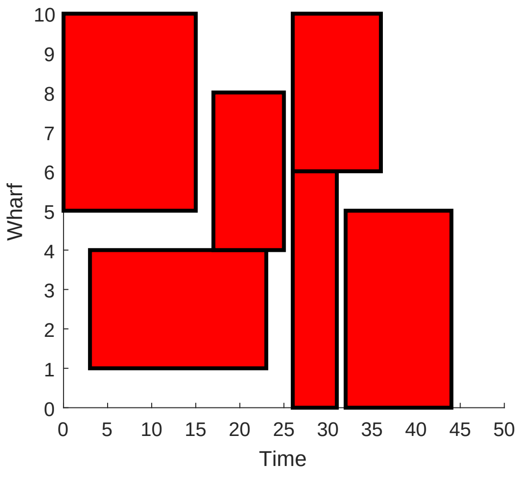


**Figure 3.6:** *Berthing plan when λ=1; Obj.Fun = 83 ( 0 3 17 26 26 32 5 1 4 0 6 0)*

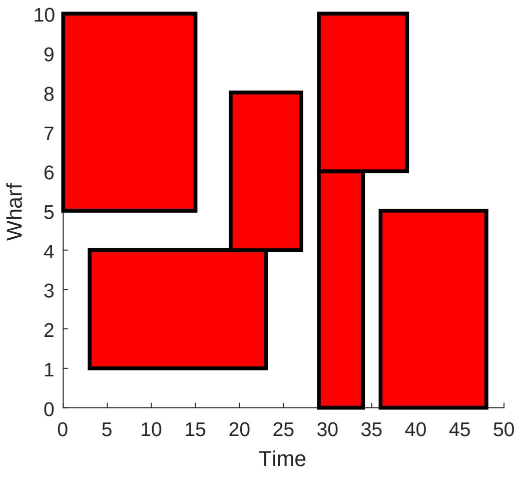


**Figure 3.7:** *Berthing plan when λ=2; Obj.Fun = 95 (0 3 19 29 29 36 5 1 4 0 6 0)*

## 3.4 B&C and GA hybrid algorithm for BAP

### 3.4.1 Solution representation: chromosome

A solution or chromosome is a strand of genes made of two parts of equal lengths. The first part represents berthing times $T_v$ of vessels while the second represents berthing positions $P_v$. The chromosomes are character rather than binary strings. See Figure 3.8 for the chromosome representation of a 3-ship problem. The example shows a solution with ships

1, 2, and 3 being processed in that order at times 42, 37, and 65 along the time axis, and moored at positions 213, 185, and 370 along the quay axis.

| Chromosome | 42 | 37 | 65 | 213 | 185 | 370 |
|---|---|---|---|---|---|---|
| $T_i$&$P_i$ | $T_1$ | $T_2$ | $T_3$ | $P_1$ | $P_2$ | $P_3$ |

**Figure 3.8:** *RBAP chromosome representation*

### 3.4.2 Initial population

The populations are sequences of integers drawn from a discrete uniform distribution on IMIN: IMAX, but within the feasible solution set defined by the constraints (3.5), (3.6) and (3.12). Note that IMIN and IMAX are chosen by the user. For each solution $(T_v, P_v)$, one can compute $C_v$ which represents the handling times plus the time buffer that we need to add depending on the value of $\zeta_{v_i v_j}$, and then compute the completion times $F_v$ given the input parameters $h_{v_i}, R_{v_i}, \lambda_{v_i}$ and the decision variable $\zeta_{v_i v_j}$ as follows.

Handling or processing time: $C_v = h_v + \lambda R_{v_i}$.

Completion or finishing time: $F_v = T_v + C_v$.

### 3.4.3 Fitness function with a penalty term

Each generated random solution is checked against constraints (3.2), (3.3) and (3.4) to see that there is no overlapping of ships in time and place dimensions. A solution that satisfies these constraints is accepted. Otherwise, it is accepted after adding a penalty term to its objective function value $Z$. The penalty term in the fitness function gradually removes infeasible solutions from the next generations. This term is computed as $\gamma_{ij} = A_{ij} \times B_{ij}$

based on the area of overlap in time and space between vessels $i$ and $j$ where

$$A_{ij} := Max(\frac{L_i + L_j}{2} - |\frac{P_i + (P_i + L_i)}{2} - \frac{P_j + (P_j + L_j)}{2}|, 0), \text{and}$$

$$B_{ij} := Max(\frac{C_i + C_j}{2} - \left|\frac{T_i + F_i}{2} - \frac{T_j + F_j}{2}\right|, 0).$$

Note that $P_i + (P_i + Li)$ and $P_j + (P_j + L_j)$ refer to the start of berthing position and their ends.

$A_{ij}$ and $B_{ij}$ are the length and overlapping intervals of ships $i$ and $j$, respectively, and $\gamma_{ij}$ is the extent of those two ships overlapping penalty that is obtained from the multiplication of two recent overlapped quantities in Cartesian time and place space. Maximum feasible error of a solution is accrued when two ships overlap, is

$$\gamma_{ij}^{max} = A_{ij}^{'} \times B_{ij}^{'},$$

where

$$A_{ij}^{'} := \frac{L_i + L_j}{2}, \text{and } B_{ij}^{'} := \frac{C_i + C_j}{2},$$

are respectively the maximum length and overlapped interval of ships $i$ and $j$, and $\gamma_{ij}^{max}$ is the maximum overlapping penalty of these ships. The value of penalty $\gamma_{ij}$ can be normalized by dividing the sum of $\gamma_{ij}$ by the sum of $\gamma_{ij}^{max}$ in the interval [0,1].

The objective of RBAP is to minimize the total vessel turnaround (or service) time, which is the sum of total handling and waiting times of all vessels, i.e., the cost of the tardiness of each vessel can be computed by summing up the berthing time of each vessel, its processing time and its time buffer less the arrival time (first term) plus how much a vessel has deviated from its preferred berth (second term). The objective function used in the GA is the same objective function as that of the mathematical model. Thus, the value of Z can be computed as:

$$Z = \sum_{v=1}^{V} C_{1v}(T_v + C_v - A_v) + \sum_{v=1}^{V} C_{2v}|P_v - b_v|$$

It is clear that the variation in Z occurs in the interval $[0, +\infty)$. By using the exponential function characteristic that $0 < 1/exp(Z) \leq 1$ for each $Z \geq 0$ , the objective function $Z$ can also be normalized in the interval (0,1] as $1/exp(Z)$ . So, the proposed fitness function for this problem can be defined as follows:

$$Fitness = \left(exp\left((Z - \sum_{i=1}^{V} C_v)/\sum_{i=1}^{V} C_v\right)\right)^{-1} - \left(\sum_{i,j\neq i}^{V} \gamma_{ij}/\sum_{i,j\neq i}^{V} \gamma_{ij}^{max}\right)^{0.3}.$$

The above fitness function for each solution consists of two parts; the first part corresponds to the objective function value and the second part corresponds to the impossibility of that response to be penalised. The first part of the fitness function causes the fitness to increase by reducing the objective function value, and the second part causes it to increase by increasing the deviations from the restrictions of non-overlapping in time and place. The power of 0.3 in the second part of the fitness function is the result of experiments to gauge the performance of the model. The two parts are normalized respectively in the intervals (0,1] and [0,1], and the changes in the fitness function fall within the interval $(-1, 1]$.

### 3.4.4 Generating next populations

After generating an initial population and evaluating the fitness of individuals we will use the genetic operators of crossover, mutation, and reproduction to generate a new population. Before making these operators, we first select suitable parents using a selection process see 2.9.

**Crossover operator** The way the recombination is implemented is as follows. Two random integer numbers are chosen from interval $[0, 2|V|]$, where $2|V|$ is the length of a chromosome, using pseudo-random numbers from a uniform distribution, as implemented in Matlab. These two integers indicate two shear points in a chromosome, one low and one high. The genes to the left of the low shear point and those to the right of the high shear point of the first parent are copied into the chromosome of the first child. In the same way, using the same shear points, the chromosome of the second parent contributes to the chromosome of the second child. The genes between the shear points are generated using a random number $\lambda \in[0,1]$ as follows,

$$Ch1 = \lfloor \lambda \times Par1 + (1-\lambda) \times Par2 \rfloor,$$

$$Ch2 = \lfloor \lambda \times Par2 + (1-\lambda) \times Par1 \rfloor,$$

where $Par1$ and $Par2$ are the sections that located between the two integers shear points of the first parent and the second parent, respectively, and $Ch1$ and $Ch2$ are the corresponding middle sections of the first and second child, respectively. It is clear that if $Par1$ and $Par2$ belong to a convex set related to constraints (3.5), (3.6), and (3.12), $Ch1$ and $Ch2$ will be in this set too. This is because if $x$ is the convex combination of two integer numbers $y$ and $z$, with $z \geq y$, then $y \leq \lfloor x \rfloor \leq z$. Constraints (3.5), (3.6), and (3.12) are then checked for feasibility to confirm the new chromosomes are feasible solutions. For illustration purposes, please see Figures 3.9 and 3.10.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Parent1 | 7 | 12 | 25 | **50** | **33** | **70** | **20** | 95 |
| Offspring1 | 7 | 12 | 25 | **49** | **38** | **72** | **28** | 95 |
| Parent2 | 9 | 15 | 22 | **48** | **62** | **80** | **60** | 84 |

**Figure 3.9:** *RBAP offspring1*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Parent1 | 7 | 12 | 25 | **50** | **33** | **70** | **20** | 95 |
| Offspring2 | 9 | 15 | 22 | **48** | **56** | **78** | **52** | 84 |
| Parent2 | 9 | 15 | 22 | **48** | **62** | **80** | **60** | 84 |

**Figure 3.10:** *RBAP offspring2*

**Mutation operator** The mutation presented in this chapter is as follows. If the selected gene is related to the berthing position, $P_v$, (defined in section 3.4), a random integer number is selected in the interval $[0, W - L_i]$, to replace the gene. If it is related to the berthing time, $T_v$, (also defined above), a random integer number is selected in the interval $[A_i, LN]$, where $LN$ is a large number, to replace it. The new generated chromosome will satisfy conditions (3.5), (3.6) and (3.12). Figure 3.11 illustrates the mutation operator.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 12 | 25 | 50 | 33 | 70 | **20** | 95 |
| 7 | 12 | 25 | 50 | 33 | 70 | **50** | 95 |

**Figure 3.11:** *RBAP mutation operator*

## 3.5 Computational experiments

Ten instances of the mathematical model of RBAP with different numbers of vessels have been solved using B&C and a hybrid meta-heuristic which combines both B&C and GA. All instances have randomly generated arrival times, expected processing times, lengths of vessels and preferred berth places.

B&C can only solve exactly small scale instances of RBAP and requires prohibitive time for large scale instances. GA, on the other hand, provides solutions for all instances, although for some instances, the quality of the approximate solutions may not be good enough. This is because of the nature of the search algorithm and the size of the search spaces of the problems. The statement is based on our limited experimentation and also what is in the literature. It was not warranted to carry out extensive experimentation with GA. However, hybridising it with B&C, i.e. developing a hybrid meta-heuristic which combines both B&C and GA is worthwhile. We have, therefore, implemented such a hybrid. The hybridisation scheme is coarse; it calls first B&C as implemented in CPLEX for short periods of time; one minute of CPU time is allocated to B&C, to be precise, before GA is invoked with the final solution provided by CPLEX. This seems to act as a a pointer to where good solutions lie. In Table 3.2, the execution time of the hybrid is the aggregated execution times of B&C and GA.

The B&C and GA together, coded in Matlab, managed to solve all 10 instances. The GA parameters for population size, probability of crossover, probability of mutation and the maximum number of generations are 200, 0.85, 0.05 and 1000, respectively. These parameters have been set by experimentation and other experiences that can be found in the literature. Moreover, 20 trials were conducted to solve each instance using GA to mitigate the randomness of the algorithm.

The hybrid has been run on a PC with Intel Core i5 and 3.20 GHz CPU with 8 GB RAM running Windows 7 Operating System. Note that CPLEX runs have been aborted after one hour of execution time. All computational results are recorded in the self explanatory Table 3.2.

In the instances considered, the number of constraints, the number of decision variables and the B&C computational time grow exponentially with the increase in the number of vessels. However, the CPU time required by B&C+GA does not grow fast with the increase in the problem size. The average objective function values and the standard deviations of B&C+GA are recorded in Table 3.2.

B&C managed to solve only few instances in reasonable times as shown in Table 3.2 column 6. We terminate CPLEX as there is no improvement after one hour of run time. Columns 7 and 8 of Table 3.2 represent the obtained objective function and CPU time using CPLEX if we run the model the same time that hybrid consumed. The hybrid algorithm, however, found solutions for all instances in short CPU times (see column 10 of Table 3.2).

The gap between B&C and the hybrid meta-heuristic solutions increases with the increase in the size of the problem (number of vessels) which translates into a large search space. Note that the gap is with CPLEX after 1 hour. In other words, as the problem size increases, the hybrid algorithm finds less and less accurate solutions unless the time increases substantially. CPLEX, on the other hand, may not even return a solution good or bad in acceptable times for large scale instances. There is therefore a tradeoff between the performances of both approaches which is captured by the gaps between the solutions returned by the two algorithms. Hybridisation reduces this gap.

**Table 3.2:** *Computational results for RBAP*

| No. | Problem Size | | $\lambda$ | CPLEX | | CPLEX2 | | B&C+GA | | | | | Gap(%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Constraints | Variables | | Obj.Val | CPU(hh:mm:ss) | Obj.Val | CPU(s) | 60 seconds | Best Obj.Val | Mean | St.Dev | Ave.CPU(s) | |
| 20 | 2360 | 1640 | 1 | 270* | 00:00:16 | 270 | 16 | 270 | 270 | 270 | 0.00 | 131 | 0.00 |
| | | | 2 | 281* | 00:00:55 | 281 | 55 | 281 | 281 | 281 | 0.00 | | 0.00 |
| 25 | 3700 | 2550 | 1 | 389* | 00:07:40 | 390 | 136 | 395 | 392 | 392 | 0.00 | 136 | 0.77 |
| | | | 2 | 408 | 01:00:00 | 412 | 136 | 421 | 394 | 395.5 | 2.20 | | -3.43 |
| 30 | 5340 | 3660 | 1 | 467 | 01:00:00 | 470 | 142 | 470 | 470 | 470 | 0.00 | 142 | 0.64 |
| | | | 2 | 486 | 01:00:00 | 527 | 142 | 562 | 504 | 522.8 | 7.98 | | 3.70 |
| 35 | 7280 | 4970 | 1 | 584 | 01:00:00 | 633 | 149 | 672 | 624 | 632.3 | 3.89 | 149 | 6.85 |
| | | | 2 | 624 | 01:00:00 | 768 | 149 | 771 | 742 | 765.6 | 10.01 | | 18.91 |
| 40 | 9520 | 6480 | 1 | 831 | 01:00:00 | 1143 | 164 | 1191 | 1041 | 1059 | 9.89 | 164 | 25.27 |
| | | | 2 | 1046 | 01:00:00 | 1228 | 164 | 1299 | 1153 | 1212 | 28.67 | | 10.23 |

* Optimal solution.

## 3.6 Summary

Berth allocation is one of the most important operations in container terminals. Determining the optimal berthing time and the best berthing position for vessels arriving at container terminals is essential for the efficient running of the these terminals. Here, a new mathematical model of the mixed integer programming type that addresses robust berth allocation is proposed. Its solutions help mitigate the uncertainty in arrival times and handling times of vessels. Instances of this model have been solved with an exact method namely Branch-and-Cut as implemented in CPLEX, an approximate approach namely GA and a hybrid of both which benefits from the exact nature of the former and the robustness of the latter. The numerical results show that the hybrid meta-heuristic B&C+GA is superior to both B&C and GA in that it finds solutions to all problems in acceptable times and accuracy. B&C+GA is characterised by its coarse hybridisation nature.

# Chapter 4

# Quay crane assignment and quay crane scheduling problem

## 4.1 Introduction

After the berth allocation problem is solved, assigning quay cranes to vessels becomes the next challenge at container terminals. This is the quay cranes assignment problem or QCAP. Once the quay cranes are assigned to each vessel, it is essential to find the optimal sequence in which to perform tasks. This is the scheduling problem known as the quay crane scheduling problem or QCSP. Choosing the best sequence to perform all the tasks on each vessel is a very important operation to minimize the handling time of every vessel.

These two operations have been solved either individually or integrated [16, 20, 83]. Solving them individually makes them more tractable in size, but the results obtained might be suboptimal as input parameters, which are outputs of the other problem, have to be approximated based on experience. Suboptimal solutions will translate into loss of revenue.

In Meisel [59] it has been suggested that "..., the QC-to-Vessel assignment is made up by single QC-hours. This assignment is allowed to change during the handling process. Time variable crane assignments are very common in practice but did not receive attention by research until then." And "In practice, the QC-to-Vessel assignment can change during the handling process of a vessel".

In this chapter, a new optimization model to represent QCAP and QCSP as QCASP is introduced. Planning simultaneously, a significant improvement of this model is that we do not force the number of quay cranes allocated to a vessel to be fixed during the whole processing period of the vessel. If necessary, a quay crane may move from one vessel to another before the processing of this vessel has finished. In addition, quay cranes do not always start their work at the same time and from the same point (initial location of quay cranes). For this reason, the initial position for each quay crane is taken into account in order to compute the exact time of quay crane travelling from their initial position to the assigned task also between any two tasks located on the same vessel or on different vessels. Also the available time (ready time) for each quay crane is considered.

There are some conditions that should be considered to solve these two problems simultaneously. After finding the berthing time and the berthing position of each vessel arriving at a container terminal by solving the BAP, they are used as inputs to the quay crane assignment problem to determine the best number of quay cranes to allocate to each vessel. The starting time of a quay crane on a vessel should be greater than or equal to its berthing time. It should also be greater than or equal to the ready crane time which means the earliest available time of the quay crane. Moreover, the starting time of quay crane should be greater than or equal to the completion time of the quay crane before it moves to other vessels. For this reason the quay crane assignment depends on the output of quay crane scheduling and berth allocation for a set of vessels. In general, after determining the number of quay cranes for each vessel, the quay crane scheduling problem arises to choose the best sequence of the tasks that will be performed by these quay cranes. In the proposed model, we do not determine the number of quay cranes for each vessel, the number of quay cranes will change depending on the what is the optimal solution. The contribution of this chapter is two fold:

1. to formulate a mathematical model that combines QCAP and QCSP in a one aggregated model (QCASP) allowing quay cranes to move between vessels while they are still being processed. In other words, it allows the number of quay cranes allocated to any vessel to change during the handling of the vessel;

2. to solve realistic instances of the problem using an adapted variant of the Genetic Algorithm (GA).

The proposed mathematical model is given in Section 4.2 and two particular instances of the problem are presented to illustrate the proposed model. Section 4.3 provides an illustration of the proposed mathematical model. An implementation of GA to find solutions to QCASP is given in section 4.4. Numerical comparisons between CPLEX and GA are given in section 4.5. Section 4.6 summarises the chapter.

## 4.2 Mathematical formulation

The formulation assumes a container terminal with a continuous berth. The solutions are the optimum number of quay cranes to be assigned to each docked vessel and their schedules to carry out all necessary moves in an optimum way. Note that the number of quay cranes assigned to a vessel at the beginning of the operation may not be the same as at the end because our model allows quay cranes to move between vessels. This modification reflects a more realistic situation in practical operations and allows the best usage of all quay cranes, and thus should result in a more efficient operations plan for the port. Also, the interference between quay cranes that move on the same rail at container terminal are avoided. Comparing with the traditional way where these two problems were considered individually, the combined model as proposed does not require us to estimate the processing time when allocating quay cranes to vessels. Therefore in general it allows a more accurate solution.

In this mathematical model, the factors which are faced by the decision makers in the real world such as the travelling time of a quay crane between two holds on the same vessel and between two holds on the different vessels have been taken into account in the modelling. Also, interferences between quay cranes are avoided by introducing non-interference constraints which consider both the potential interference between cranes of doing tasks on the same vessel and those doing works on different vessels. Note that we do not force the number of quay cranes allocated to a vessel to be fixed during the whole processing period of the vessel. If necessary, a quay crane may move from one vessel to another before the processing of this vessel has finished. This is more flexible than using a fixed number of quay cranes to handle a vessel; it has the potential to give better working plans and to some extent, to mitigate uncertainty linked to the performance of quay cranes. In addition, quay cranes do not always start their work from the same point and at the same time . For this reason, the initial position for each quay crane is taken into account in order to compute the exact travelling time of a quay crane and the available time for each quay crane. Precedence and simultaneity constraints are taken into account as well.

### 4.2.1 Assumptions

Consider a continuous berth container terminal with fixed length and berth allocation already decided. Now assume that:

1- The berthing position and berthing time of vessels are given as inputs;

2- Each vessel is divided longitudinally into bays; all bays have the same length. Thus, the length of a vessel is given in terms of the number of its bays;

3- The safety distance between each pair of adjacent quay cranes depends on the width of a bay;

4- Once a quay crane starts processing a task, it can leave only after it has finished it;

5- Quay cranes are on the same rail and thus cannot cross over each other;

6- Some tasks must be performed before others and there are some tasks which cannot be performed simultaneously.

### 4.2.2 Indices

| | |
|---|---|
| $V$ | Number of vessels $v(i, j = 1, 2, ..., V)$. |
| $B$ | Number of tasks on the vessels $b(i, j = 1, 2, ..., B)$. |

| | |
|---|---|
| $Q$ | Number of quay cranes $q(i, j = 1, 2, ..., Q)$. |

### 4.2.3 Parameters

| | |
|---|---|
| $p_b^v$ | Time required to perform task $b$ on vessel $v$. |
| $l_b^v$ | Location of task $b$ on vessel $v$ expressed by the ship bay number on vessel $v$. |
| $r_q$ | Earliest available time of the $q^{th}$ quay crane. |
| $I_0^q$ | Initial location of quay crane $q$ which is relative to the ship-bay number. |
| $t_{b_i b_j}^{qv}$ | Travel time of the $q^{th}$ quay crane from the location $l_{b_i}^v$ of task $b_i$ to the location $l_{b_j}^v$ of task $b_j$. $t_{b_0 b_j}^{qv}$ represents the travel time from the initial position $I_0^q$ of the $q^{th}$ quay crane to the location $l_{b_j}^v$ of the task $b_j$ on vessel $v$. Note that the value of $t$ represents the travelling time between two adjacent bays. |
| $T_v$ | Berthing time of vessel $v$. |
| $P_v$ | Berthing position of vessel $v$. |
| $d_v$ | Requested departure time for vessel $v$. |
| $W_v$ | Tardiness cost of vessel $v$ per time unit. |
| $R_v$ | Earliness income of vessel $v$ per time unit . |
| $\Psi$ | Set of pairs of tasks that cannot be performed simultaneously. When tasks $b_i$ and $b_j$ cannot be performed simultaneously, then $(b_i, b_j) \in \Psi$. |
| $\Phi$ | Set of ordered pairs of tasks for which there is a precedence relationship. When task $b_i$ must precede task $b_j$, then we have $(b_i, b_j) \in \Phi$. |
| $M$ | Arbitrary large positive number. |

### 4.2.4 Binary decision variables

$$X^{qv}_{b_i b_j} = \begin{cases} 1 & \text{if the } q^{th} \text{ quay crane performs task } b_j \text{ immediately after performing task } b_i \text{ on} \\ & \text{vessel } v. \\ 0 & \text{otherwise} \end{cases}$$

Tasks $b_0$ and $b_{B_v+1}$ are considered as the dummy initial and final states of each quay crane, respectively. Thus, when task $b_j$ is the first task of the $q^{th}$ quay crane on vessel $v$ then $X^{qv}_{b_0 b_j} = 1$. Similarly, when task $b_j$ is the last task of the $q^{th}$ quay crane on vessel $v$ then $X^{qv}_{b_j b_{B_v+1}} = 1$.

$$Z^{v}_{b_i b_j} = \begin{cases} 1 & \text{if task } b_j \text{ starts later than the finish time of task } b_i \text{ on vessel } v. \\ 0 & \text{otherwise.} \end{cases}$$

$$Y^{q}_{v_i v_j} = \begin{cases} 1 & \text{if the } q^{th} \text{ quay crane is assigned to vessel } v_j \text{ immediately after finishing its task} \\ & \text{on vessel } v_i. \\ 0 & \text{otherwise.} \end{cases}$$

$$\beta^{v_i v_j}_{b_i b_j} = \begin{cases} 1 & \text{if the task } b_j \text{ on vessel } v_j \text{ starts later than the finish time } D \text{ of task } b_i \text{ on vessel } v_i \\ 0 & \text{otherwise.} \end{cases}$$

$$\alpha^{v_i v_j}_{b_i b_j} = \begin{cases} 1 & \text{if the task } b_i \text{ on vessel } v_i \text{ is located below the task } b_j \text{ on vessel } v_j. \\ 0 & \text{otherwise.} \end{cases}$$

### 4.2.5 Continuous decision variables

| | |
|---|---|
| $E_v$ | Earliness of vessel $v$. |
| $A_v$ | Tardiness of vessel $v$. |
| $S_{qv}$ | Starting time of $q^{th}$ quay crane on vessel $v$. |
| $D_{b_i}^v$ | Completion time of task $b_i$ on vessel $v$. |
| $C_{qv}$ | Completion time of $q^{th}$ quay crane on vessel $v$. |
| $F_v$ | Finishing (departure) time of vessel $v$. |

### 4.2.6 The mathematical model

$$\min Z = \sum_{v=1}^{V} W_v A_v - \sum_{v=1}^{V} R_v E_v \tag{4.1}$$

s.t

$$d_v - F_v = E_v - A_v \qquad \forall v \tag{4.2}$$

$$\sum_{v_j=1}^{V} Y_{v_0 v_j}^q = 1 \qquad \forall q \tag{4.3}$$

$$\sum_{v_i=1}^{V} Y_{v_i(V+1)}^q = 1 \qquad \forall q \tag{4.4}$$

$$\sum_{v_j=1}^{V+1} Y_{v v_j}^q - \sum_{v_j=0}^{V} Y_{v_j v}^q = 0 \qquad \forall v, q \tag{4.5}$$

$$\sum_{v_i=0}^{V} \sum_{q=1}^{Q} Y_{v_i v_j}^q \geq 1 \qquad \forall v_j \tag{4.6}$$

$$S_{qv} \geq r_q - M(1 - Y_{v_0 v}^q) \qquad \forall v, q \tag{4.7}$$

$$S_{qv} \geq T_v - M(1 - \sum_{v_j=1}^{V+1} Y_{v v_j}^q) \qquad \forall v, q \tag{4.8}$$

$$S_{qv_j} \geq C_{qv_i} - M(1 - Y^{q}_{v_i v_j}) \qquad \forall v_i, v_j, q \tag{4.9}$$

$$\sum_{b_j=1}^{B_v} X^{qv}_{b_0 b_j} = \sum_{v_i=0}^{V} Y^{q}_{v_i v} \qquad \forall v, q \tag{4.10}$$

$$\sum_{b_j=1}^{B_v} X^{qv}_{b_j b_{B_v+1}} = \sum_{v_i=0}^{V} Y^{q}_{v_i v} \qquad \forall v, q \tag{4.11}$$

$$\sum_{b_j=1}^{B_v+1} X^{qv}_{b b_j} - \sum_{b_j=0}^{B_v} X^{qv}_{b_j b} = 0 \qquad \forall b, v, q \tag{4.12}$$

$$\sum_{q=1}^{Q} \sum_{b_i=0}^{B_v} X^{qv}_{b_i b_j} = 1 \qquad \forall b_j, v \tag{4.13}$$

$$\sum_{b_i=0}^{B_v} \sum_{b_j=1}^{B_v+1} X^{qv}_{b_i b_j} \leq M \sum_{v_i=0}^{V} Y^{q}_{v_i v} \qquad \forall v, q \tag{4.14}$$

$$D^{v}_{b_i} + p^{v}_{b_i} + t^{qv}_{b_i b_j} - D^{v}_{b_j} \leq M(1 - X^{qv}_{b_i b_j}) \qquad \forall b_i, b_j, v, q \tag{4.15}$$

$$S_{qv} + p^{v}_{b_j} + t^{qv}_{b_0 b_j} - D^{v}_{b_j} \leq M(1 - X^{qv}_{b_0 b_j}) \qquad \forall b_j, v, q \tag{4.16}$$

$$D^{v}_{b_j} - C_{qv} \leq M(1 - X^{qv}_{b_j b_{B_v+1}}) \qquad \forall b_j, v, q \tag{4.17}$$

$$C_{qv} - F_v \leq M(1 - \sum_{v_j=1}^{V+1} Y^{q}_{v v_j}) \qquad \forall v, q \tag{4.18}$$

$$D^{v}_{b_i} + p^{v}_{b_j} \leq D^{v}_{b_j} \qquad \forall (b_i, b_j) \in \Phi_v, v \tag{4.19}$$

$$D^{v}_{b_i} - D^{v}_{b_j} + p^{v}_{b_j} \leq M(1 - Z^{v}_{b_i b_j}) \qquad \forall b_i, b_j, v \tag{4.20}$$

$$Z^{v}_{b_i b_j} + Z^{v}_{b_j b_i} = 1 \qquad \forall (b_i, b_j) \in \Psi_v, v \tag{4.21}$$

$$\sum_{\theta=0}^{Q} \sum_{\kappa=0}^{B_v} X^{\theta v}_{\kappa b_j} - \sum_{\theta=0}^{Q} \sum_{\kappa=0}^{B_v} X^{\theta v}_{\kappa b_i} \leq M(Z^{v}_{b_i b_j} + Z^{v}_{b_j b_i}) \qquad \forall b_i, b_j, v, q; l_{b_i} < l_{b_j} \tag{4.22}$$

$$D^{v_i}_{b_i} - D^{v_j}_{b_j} + p^{v_j}_{b_j} \leq M(1 - \beta^{v_i v_j}_{b_i b_j}) \qquad \forall b_i, b_j, v_i, v_j \tag{4.23}$$

$$P_{v_i} + l^{v}_{b_i} \leq P_{v_j} + l^{v}_{b_j} + M(1 - \alpha^{v_i v_j}_{b_i b_j}) \qquad \forall b_i, b_j, v_i, v_j \tag{4.24}$$

$$\beta^{v_i v_j}_{b_i b_j} + \beta^{v_j v_i}_{b_j b_i} + \alpha^{v_j v_i}_{b_j b_i} \geq \sum_{\kappa=0}^{B_v} X^{q_i v_i}_{\kappa b_i} + \sum_{\kappa=0}^{B_v} X^{q_j v_j}_{\kappa b_j} - 1 \qquad \forall b_i, b_j, v_i, v_j, q_i, q_j; v_j \neq v_i; q_i < q_j \tag{4.25}$$

$$X^{qv}_{b_ib_j}, Z^{v}_{b_ib_j}, Y^{q}_{v_iv_j}, \alpha^{v_iv_j}_{b_ib_j}, \beta^{v_iv_j}_{b_ib_j} \in \{0,1\} \qquad \forall b_i, b_j, v_i, v_j, v, q \qquad (4.26)$$

$$C_{qv}, F_v, D^{v}_{b_j}, P_v, T_v \geq 0 \qquad \forall b_j, v, q \qquad (4.27)$$

In the objective function (4.1), the first term $\sum_{v=1}^{V} W_v A_v$ represents the tardiness cost if the departure time of a vessel is greater than its due time. The second term $\sum_{v=1}^{V} R_v E_v$ represents the earliness income if the finishing time of a vessel is less than its due time. Note that in practice this reward for earliness may be zero. Constraints (4.2) calculate the earliness or tardiness of a vessel depending on the difference between its due time and its finishing time.

The constraints (4.3)-(4.6) represent the main conditions for QCAP. However, constraints (4.3) and (4.4) respectively select the first and the last ships for each quay crane. Constraints (4.5) guarantee that ships are processed in a well-defined sequence. Constraints (4.6) guarantee that each vessel be handled by at least one quay crane. This set of constraints is not really necessary since one can imagine that a large number of quay cranes is available while only a few vessels are to be handled. However, this situation is unlikely and in general only a few quay cranes are available to handle a large number of vessels. Therefore, we do not want any of the quay cranes to be idle. Hence the need for these constraints.

The constraints (4.7)-(4.9) determine the starting time of quay cranes. Constraints (4.7) force the starting time of the earliest vessel that is to be done by the $q^{th}$ quay crane to be after the ready time of the $q^{th}$ quay crane. Note that vessel $v_0$ is a dummy vessel from which the working sequence starts. Constraints (4.8) say that the starting time of the $q^{th}$ quay crane on the vessel $v$ is no earlier than the berthing time of vessel $v$ if the $q^{th}$ quay crane is assigned to serve this vessel. Constraints (4.9) ensure that the starting time of the $q^{th}$ quay crane on vessel $v_j$ is no earlier than the finishing time of its predecessor vessel $v_i$.

Constraints (4.10) ensure that if a quay crane is assigned to a vessel, then it will start its processing from one of the tasks on that vessel. Constraints (4.11) ensure that if a quay crane is assigned to a vessel, then it will finish its processing with one of the tasks on that vessel. Constraints (4.12) show a flow balance ensuring that tasks are performed in a well-defined sequence on every vessel. Constraints (4.13) ensure that every task on each vessel must be handled by exactly one quay crane. Constraints (4.14) ensure that tasks on a vessel are handled by a quay crane only if this quay crane is allocated to that vessel. Constraints (4.15) simultaneously determine the completion time for each task and eliminate sub-tours; sub-tours here are the looping on tasks which have already been done. To illustrate, let Task 1, Task 2, and Task 3, be carried out in this order. A sub-tour would be to do Task 1, Task 2, Task 3, and Task 2 again, for instance. Constraints (4.15) remove this possibility. Constraints (4.16) determine the quay crane starting time on vessel $v$ and the completion time of the same quay crane is computed by constraints (4.17). Constraints (4.18) determine the finishing time of each vessel.

When required, constraints (4.19) force task $b_i$ to be completed before the starting of task $b_j$ for all the task pairs $(b_i, b_j) \in \Phi$. Constraints (4.20) define $Z^v_{b_i b_j}$ such that $Z^v_{b_i b_j} = 1$ when the operation of task $b_j$ on vessel $v$ starts after the completion of task $b_i$ on the same vessel. Constraints (4.21) ensure that the pair of tasks that are members of the set $\Psi$ will not be handled simultaneously.

By constraints (4.22), interference between quay cranes is avoided. Suppose that tasks $b_i$ and $b_j$ are performed simultaneously and $l_i < l_j$, this means that $Z^v_{b_i b_j} + Z^v_{b_j b_i} = 0$. Note that both quay cranes and tasks are ordered in an increasing order of their relative location in the direction of increasing ship-bay number. Suppose that, for $q_1 < q_2$, quay crane $q_1$ performs task $b_j$ and quay crane $q_2$ performs task $b_i$. Then, interference between quay cranes $q_1$ and $q_2$ results in $\sum_{\theta=1}^{q_1}\sum_{\kappa=0}^{B_v} X_{\kappa b_j}^{\theta v} - \sum_{\theta=1}^{q_1}\sum_{\kappa=0}^{B_v} X_{\kappa b_i}^{\theta v} = 1$, where $\kappa$ refers to the set of all tasks. This violates constraints (4.22), since we have $Z_{b_i b_j}^{v} + Z_{b_j b_i}^{v} = 0$ as mentioned earlier.

Constraints (4.23)-(4.25), which are introduced in this study for the first time, avoid interference amongst different tasks on different vessels, and enable quay cranes to move freely amongst vessels. In constraints (4.23), $\beta_{b_i b_j}^{v_i v_j}$ is defined as follows. $\beta_{b_i b_j}^{v_i v_j} = 1$ if the finishing time $D$ of task $b_i$ on vessel $v_i$ plus the processing time of task $b_j$ on vessel $v_j$ is less than or equal to the finishing time of task $b_j$ on vessel $v_j$; 0 if the finishing time of task $b_i$ on vessel $v_i$ plus the processing time of task $b_j$ on vessel $v_j$ is greater than the finishing time $D$ of task $b_j$ on vessel $v_j$. Figures 4.1 and 4.2 illustrate how the value of $\beta_{b_i b_j}^{v_i v_j}$ is computed.

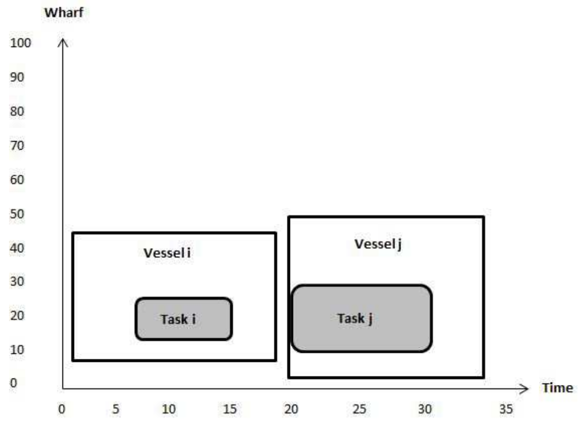


**Figure 4.1:** *No time overlap between task $b_i$ on vessel $v_i$ and task $b_j$ on vessel $v_j$*

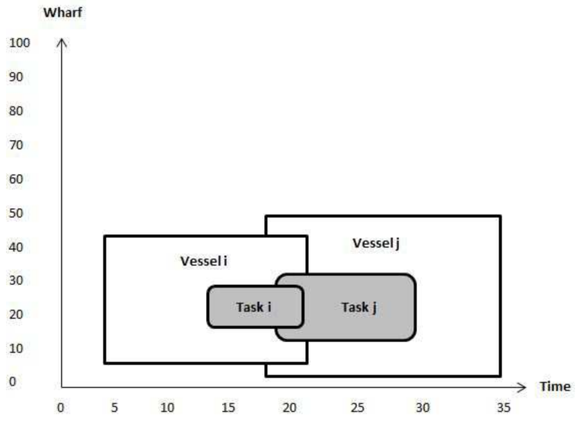


**Figure 4.2:** *Time overlap between task $b_i$ on vessel $v_i$ and task $b_j$ on vessel $v_j$*

The value of $\beta_{b_i b_j}^{v_i v_j}$ in Figure 4.1 can be 1 because $D_{b_i}^{v_i} + p_{b_j}^{v_j} \leq D_{b_j}^{v_j}$, whereas the value of $\beta_{b_j b_i}^{v_j v_i}$ in the same figure equals 0 because $D_{b_j}^{v_j} + p_{b_i}^{v_i} > D_{b_i}^{v_i}$. The value of $\beta_{b_i b_j}^{v_i v_j}$ in Figure 4.2 equals 0 because $D_{b_i}^{v_i} + p_{b_j}^{v_j} > D_{b_j}^{v_j}$ and the value of $\beta_{b_j b_i}^{v_j v_i}$ in the same figure equals 0 because $D_{b_j}^{v_j} + p_{b_i}^{v_i} > D_{b_i}^{v_i}$. This means there is overlap in the time between these two tasks on these two vessels.

In constraints (4.24), $\alpha^{v_i v_j}_{b_i b_j}$ is defined as follows. $\alpha^{v_i v_j}_{b_i b_j} = 1$ if the berthing position $P$ of vessel $v_i$ plus the location $l$ of task $b_i$ on that vessel is less than or equal to the berthing position $P$ of vessel $v_j$ plus the location $l$ of task $b_j$ on that vessel; 0 if the berthing position of vessel $v_i$ plus the location of task $b_i$ on that vessel is greater than the berthing position of vessel $v_j$ plus the location of task $b_j$ on that vessel. Figures 4.3 and 4.4 illustrate how the value of $\alpha^{v_i v_j}_{b_i b_j}$ is computed.

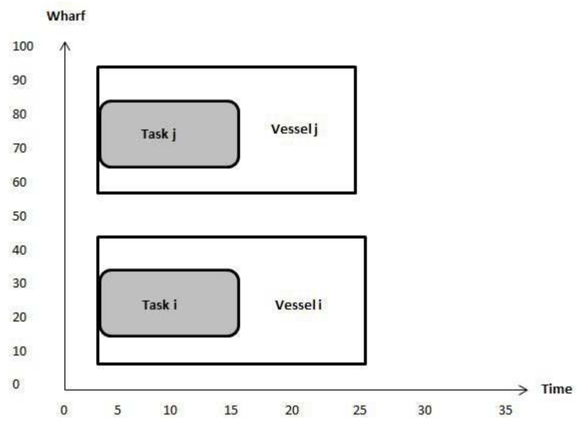


**Figure 4.3:** *No location overlap between task $b_i$ on vessel $v_i$ and task $b_j$ on vessel $v_j$*

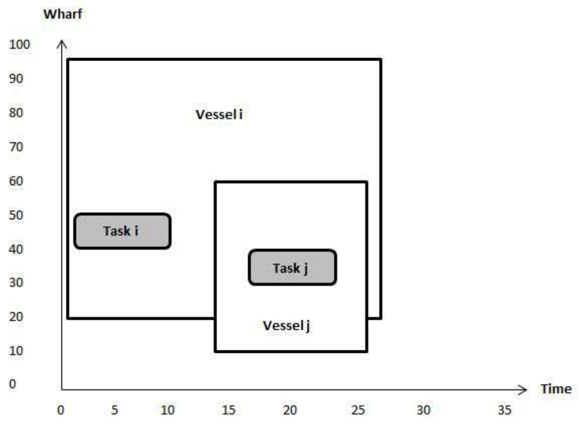


**Figure 4.4:** *Location overlap between task $b_i$ on vessel $v_i$ and task $b_j$ on vessel $v_j$*

The value of $\alpha^{v_i v_j}_{b_i b_j}$ in Figure 4.3 can be 1 because $P_{v_i} + l^v_{b_i} \leq P_{v_j} + l^v_{b_j}$. The value of $\alpha^{v_i v_j}_{b_i b_j}$ in Figure 4.4 equals 0, because $P_{v_i} + l^v_{b_i} > P_{v_j} + l^v_{b_j}$. This means there is overlap in the position between these two tasks on these two vessels.

Constraints (4.25) prevent the interference between quay cranes handling two different vessels depending on the values of $\beta^{v_i v_j}_{b_i b_j}$ and $\alpha^{v_i v_j}_{b_i b_j}$, respectively.

The size of the above model is as follows:

The number of binary variables is equal to $VB(2VB - B - 1) + VQ(B^2 + 2B + V + 2)$.

The number of integer variables is equal to $3V + VB + 2VQ$.

The number of constraints is equal to

$$VBQ(VBQ - BQ - VB + 3B + 1) + VB(B + 2V - 2) + VQ(V + 6) + 2B^2 - 2B + 2V + 2Q.$$

## 4.3 Numerical examples

To illustrate the case of better plans, consider the situation in which two quay cranes are available to handle two vessels with data as given in Table 4.1, and each vessel has two tasks to process.

**Table 4.1:** *Input data of Example 4.1*

| | | |
|---|---|---|
| Ready (crane) | 0 | 2 |
| Initial location (crane) | 11 | 16 |
| Berthing time | 0 | 0 |
| Processing time of tasks ,vessel 1 | 85 | 29 |
| Processing time of tasks ,vessel 2 | 18 | 33 |
| Location task, vessel 1 | 1 | 2 |
| Location task, vessel 2 | 1 | 2 |
| Expected departure time of vessel | 85 | 33 |
| Berthing position | 10 | 15 |
| Tardiness cost (per unit time) | 5 | 1 |
| Earliness income (per unit time) | 1 | 1 |

In the previous models where a fixed number of quay cranes is allocated to every vessel during the whole processing period, the optimal working plan is described in Figure 4.5 with the objective value being equal to 171 (150 units of tardiness belong to the first vessel and 21 units of tardiness belong to the second vessel). Even though quay crane 2 finished its work on vessel 2 at 54 (2+18+1+33), it is not allowed to move to vessel 1 according to the constraints in the previous mathematical models. This wastes the working time of this quay crane which will result into a sub-optimal solution. In contrast, in our model, a variable number of quay cranes is used during the processing period, which allows the second quay crane to move to vessel 1 to perform other tasks as showing in Figure 4.6. As a result, vessel 1's finishing time is completed at 87 time units earlier than in the previous plan with an objective value equals to 31 (10 units of tardiness belong to the first vessel and 21 units of tardiness belong to the second vessel), due to making the best use of both quay cranes. Note that we only allow the quay crane to move after it has finished its allocated work on vessel 2.

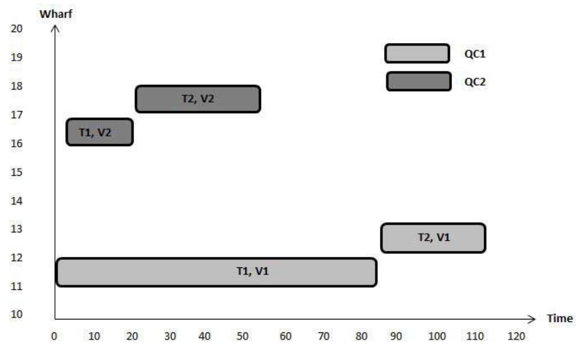


**Figure 4.5:** *Model solution for fixed number of QC's of example 4.1*

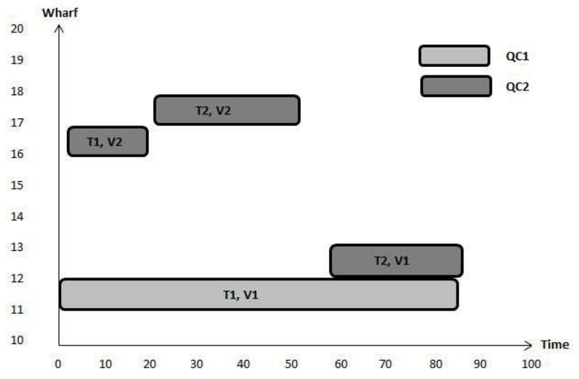


**Figure 4.6:** *Suggested model solution of example 4.1*

Our model also allows quay cranes to share tasks on the same vessels to which they are allocated. Consider the input data for two vessels arriving at a container terminal as given in Table 4.2.

**Table 4.2:** *Input data of Example 4.2*

| | | |
|---|---|---|
| Ready (crane) | 0 | 0 |
| Initial location (crane) | 22 | 24 |
| Berthing time | 0 | 0 |
| Processing time of tasks,vessel 1 | 28 | 22 |
| Processing time of tasks,vessel 2 | 39 | 34 |
| Location task, vessel 1 | 1 | 2 |
| Location task, vessel 2 | 1 | 2 |
| Expected departure time of vessel | 28 | 39 |
| Berthing position | 20 | 25 |
| Tardiness cost (per unit time) | 1 | 1 |
| Earliness income (per unit time) | 1 | 1 |

In Example 4.2, the objective value returned by previous models is 60 time units (23 units of tardiness belong to the first vessel and 37 units of tardiness belong to the second vessel), (see Figure 4.7). Since we allow quay cranes to move between vessels if no interference constraints are violated, the solution of our model returns an objective value of only 35 time units (1 unit of tardiness belong to the first vessel and 34 units of tardiness belong to the second vessel), as can be seen in Figure 4.8.

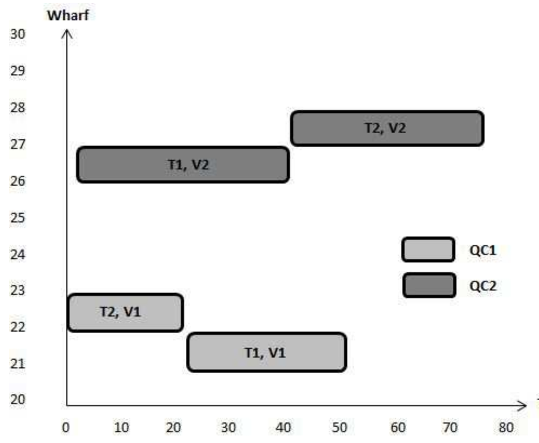


**Figure 4.7:** *Model solution for fixed number of QC's of example 4.2*

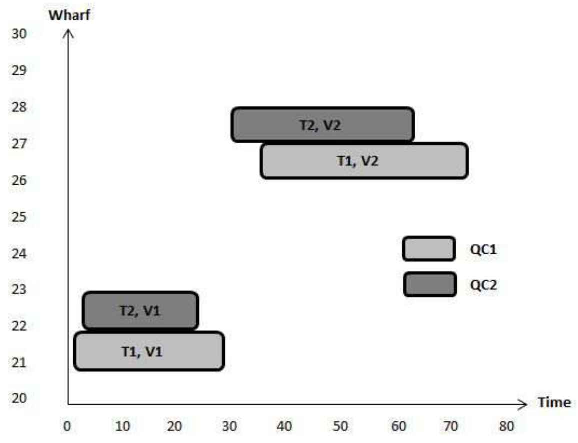


**Figure 4.8:** *Suggested model solution of example 4.2*

## 4.4 Application of GA to QCASP

### 4.4.1 Solution representation: chromosome

GA starts with a randomly generated population of solutions. Here, representing a sequence of holds (tasks) for all docked vessels. The value of a gene is randomly picked from the index set of all holds; it cannot, therefore, be duplicated, i.e. each gene is unique. Each chromosome consists of $v \times b$ genes, where $v$ represents the number of vessels and $b$ the number of tasks on each vessel. A simple chromosome for the case of two vessels, each with three tasks, is illustrated in Figure 4.9. Here, genes (1,2,3) represent the tasks on the first vessel and genes (4,5,6) represent those on the second vessel.

| | | | | | |
|---|---|---|---|---|---|
| 1 | 4 | 3 | 6 | 5 | 2 |

**Figure 4.9:** *QCASP chromosome representation*

Based on the sequence of tasks for all vessels represented by the chromosome, a quay crane schedule can be constructed using the following steps that are the extension of the procedure proposed by Lee et al. [45] and used for each vessel separately. Here, however, we assume that the berth allocation plan (berthing time and berthing position of each vessel arriving at the container terminal) as well the initial position and ready time of each quay crane at the beginning of scheduling are known.

## Quay crane scheduling procedure [45]:

Begin

Step 1: Based on the current position of each quay crane, determine which quay cranes can handle the first unassigned task in the chromosome without interference with other quay cranes. If only one quay crane is available, this task is assigned to it and it is deleted from the chromosome; the position and the completion time of the assigned quay crane are updated. The completion time of task is also computed. If two quay cranes are available, go to Step 2.

Step 2: Compare the completion times of the two available quay cranes, and assign this task to the quay crane with earlier completion time. This task is then deleted from the chromosome, and both the position and the completion time of the assigned quay crane are updated. Also the completion time of task is computed. If their completion times are equal, go to Step 3.

Step 3: Compare the distance between this task and these two available quay cranes, and assign the task to the quay crane with the shorter distance. Then, this task is deleted from the chromosome, and both the position and the completion time of the assigned quay crane are updated. The completion time of task is also computed. If their distances are equal, go to Step 4.

Step 4: Assign this task to the quay crane with the smaller number. Then, this task is deleted from the chromosome, and both the position and the completion time of the assigned quay crane are updated. Also the completion time of task is computed.

Step 5: Steps 1–4 are repeated until all the tasks in the chromosome are assigned.

Stop

### 4.4.2 Solution validation

To validate chromosomes/solutions, three important situations must be considered. The first one is the precedence relationship between tasks. For instance, some of the bays of a given vessel need to be unloaded and loaded. The discharging of containers from a bay must precede the loading of this bay. For this reason the generated chromosome should be checked to see if it satisfies this condition, i.e. constraints (4.19). The second situation is the non-simultaneity of some tasks, i.e. constraints (4.21) must also be satisfied. Finally, the interference between quay cranes is avoided by introducing non-interference constraints which consider both the potential interference of tasks on the same vessel constraints (4.22) and those on different vessels constraints (4.25). If any one of constraints (4.19), (4.21), (4.22), (4.25) is violated or any combination are violated, the generated chromosomes are discarded by putting a high penalty to their fitness values.

### 4.4.3 Evaluation of fitness

The objective of QCASP is to minimise the tardiness and to maximise the earliness of vessels. The completion time of each quay crane can be computed by summing up the processing times of all the tasks that have been performed by this quay crane plus the travel time it takes to move from one hold to another. The tardiness of each vessel can be computed by subtracting the finishing time from the expected departure time. The finishing time represents the maximum processing time of the vessel required by the quay cranes assigned to it. The objective function used by the GA in MATLAB is the same objective function as that of the mathematical model. Thus, the fitness value of a chromosome is calculated by Equation (4.28).

$$Fitness(chromosome) = \frac{1}{\sum_{v=1}^{V} W_v A_v - \sum_{v=1}^{V} R_v E_v} \tag{4.28}$$

Note that there is no difference between the two formulae of the fitness function, i.e. $\sum_{v=1}^{V} W_v A_v - \sum_{v=1}^{V} R_v E_v$ and $\frac{1}{\sum_{v=1}^{V} W_v A_v - \sum_{v=1}^{V} R_v E_v}$. For a minimisation problem, which is our case, in the first formula we sort the solutions/chromosomes in ascending order resulting in the best solutions being at the top; in the second formula, the solutions are sorted in a descending order putting again the best solutions at the top.

### 4.4.4 Generating next populations

From the initial randomly generated population, subsequent generations of children, i.e. new populations, must be created. This is achieved by using genetic operators such as crossover, mutation and reproduction (copying of individuals unmodified into subsequent populations).

**Crossover operator** To produce a new chromosome (offspring) the 'Order Crossover' of Cheng and Gen [9] is used. Order crossover is a permutation-based crossover. It works as follows. A subsequence of consecutive alleles from parent 1 is selected and used to partially make the offspring; the remaining alleles to complete the creation of the offspring are chosen from parent 2 avoiding any repetitions. The same procedure is then applied starting from parent 2 to make the second offspring. The crossover operator always creates two offspring. Figures 4.10 and 4.11 illustrate the order crossover.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parent1 | 7 | 12 | 5 | 3 | 6 | 1 | 10 | 8 | 11 | 2 | 4 | 9 |
| Offspring1 | **7** | **12** | **5** | **3** | 8 | 2 | 6 | 4 | 11 | 9 | 10 | 1 |
| Parent2 | 8 | 2 | 5 | 6 | 7 | 4 | 11 | 9 | 12 | 3 | 10 | 1 |

**Figure 4.10:** *QCASP offspring 1*

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parent1 | 7 | 12 | 5 | 3 | 6 | 1 | 10 | 8 | 11 | 2 | 4 | 9 |
| Offspring2 | **8** | **2** | **5** | **6** | 7 | 12 | 3 | 1 | 10 | 11 | 4 | 9 |
| Parent2 | 8 | 2 | 5 | 6 | 7 | 4 | 11 | 9 | 12 | 3 | 10 | 1 |

**Figure 4.11:** *QCASP offspring 2*

**Mutation operator** In this algorithm, two genes from the chromosome are randomly selected and then swapped with each other. Figure 4.12 illustrates the mutation operator.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 8 | 4 | **3** | 6 | 5 | 2 | 7 | **9** | 1 |
| 8 | 4 | **9** | 6 | 5 | 2 | 7 | **3** | 1 |

**Figure 4.12:** *QCASP mutation operator*

## 4.5 Computational experiments

Twenty instances of the above mathematical model of QCASP with different numbers of vessels, tasks, and quay cranes have been solved using CPLEX and GA. They are recorded in Tables 4.3 and 4.4. All instances have randomly generated processing time of tasks for each hold from the uniform distribution $U(10,50)$.

CPLEX solved problems 1 to 10. These are relatively small in size. They were solved in acceptable times, although 28, 68, 33 and 46 hours were required for problems 6, 7, 8 and 9, respectively. These times are hardly acceptable in the context of container terminal operations. The rest of the problems, 11 to 20, which are the realistic instances, could not be solved with CPLEX in acceptable times ($\geq$ 100 hours of CPU time). In some cases they could not be solved at all due to the limitations of the computing platform used. Note that the B&C implemented in CPLEX does not take any problem specific cuts. At least, we are not aware that the package such a facility. We have submitted out problems as a non-expert wood, i.e with no preprocessing or choice of parameters to make the execution faster or slower.

GA, coded in Matlab, managed to solve all 20 instances. For the small size problems of Table 4.3, the GA parameters of population size, rate of crossover, rate of mutation and the maximum number of generations are set as 150, 0.2, 0.1 and 500, respectively. In the case of the large size instances of Table 4.4, population size, crossover and mutation rates and the maximum number of generations are set as 300, 0.25, 0.2 and 1000, respectively. These parameters have been set by experimentation and other experiences that can be found in the literature. Moreover, 20 trials were conducted to solve each instance using GA to mitigate the randomness of the algorithm.

All experiments have been performed on a PC with Intel Core i5 and 3.20 GHz CPU with 8 GB RAM running Windows 7 Operating System. The 20 problems and their corresponding results are presented in Tables 4.3 and 4.4, containing small problems 1 through 10 and longer ones consisting problems 11 through to 20, respectively.

### 4.5.1 Test results

On the small size problems of Table 4.3, the number of constraints, the number of decision variables and CPLEX computational time grow exponentially with the instance number of vessels, tasks and quay cranes. Note, however, that in column 13 of Table 4.3 showing the CPU time required by GA to solve these problems, this time does not change much with the increase in the problem size. It is also important to note that in most cases, GA obtains the optimal or near optimal solution within 500 generations. The average gap between CPLEX and GA returned objective values is about 0.4%.

**Table 4.3:** *Computational results for small scale instances of QCASP*

| No. | Problem Information | | | Problem Size | | | CPLEX | | GA | | | | Gap(%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Vessels | Tasks | Q.Cranes | Constraints | Dec.Vars | Int.Vars | Obj.Val | CPU(hh:mm:ss) | Best Obj.Val | Mean | St.Dev | Ave.CPU(s) | |
| 1 | 2 | 3 | 5 | 538 | 270 | 243 | 70 | 00:00:05 | 70 | 70 | 0.0 | 15 | 0.0 |
| 2 | 4 | 2 | 5 | 1030 | 444 | 389 | 242 | 00:06:11 | 244 | 244 | 0.0 | 14 | 0.8 |
| 3 | 3 | 3 | 5 | 1141 | 474 | 431 | 405 | 00:40:54 | 405 | 412.5 | 8.3 | 13 | 0.0 |
| 4 | 2 | 5 | 5 | 1244 | 566 | 535 | 105 | 04:41:12 | 105 | 105 | 0.0 | 18 | 0.0 |
| 5 | 4 | 2 | 8 | 2152 | 636 | 560 | 156 | 00:34:34 | 156 | 156 | 0.0 | 14 | 0.0 |
| 6 | 4 | 3 | 4 | 1412 | 632 | 580 | 865 | 28:59:44 | 884 | 904.6 | 28.7 | 17 | 2.1 |
| 7 | 3 | 4 | 6 | 2472 | 807 | 756 | 1264 | 68:06:22 | 1268 | 1268 | 0.0 | 14 | 0.3 |
| 8 | 3 | 5 | 5 | 2776 | 1014 | 965 | 1035 | 33:56:21 | 1035 | 1038.5 | 20.9 | 15 | 0.0 |
| 9 | 3 | 5 | 8 | 6013 | 1392 | 1328 | 770 | 46:09:14 | 780 | 780 | 0.0 | 15 | 1.2 |
| 10 | 2 | 8 | 8 | 5972 | 1766 | 1720 | 320 | 06:05:31 | 320 | 320 | 0.0 | 18 | 0.0 |

$$Gap = \frac{GAObj.Fun. - CPLEXObj.Fun.}{CPLEXObj.Fun.} * 100$$

CPLEX did not solve some of the larger size instances due to the limitation of the computing platform used, shown in Table 4.4, and marked with a star in column 8 due to the limitation of the computing platform used. GA, however, finds the optimal or near optimal solutions for all the instances in reasonable CPU times (see column 13 of Table 4.4). Note that CPLEX runs have been terminated after three hours of execution time.

**Table 4.4:** *Computational results for large scale instances of QCASP*

| No. | Problem Information | | | Problem Size | | | CPLEX | | GA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Vessels | Tasks | Q.Cranes | Constraints | Dec.Vars | Int.Vars | Obj.Val | CPU(hh:mm:ss) | Best Obj.Val | Mean | St.Dev | Ave.CPU(s) |
| 11 | 5 | 10 | 8 | 67966 | 9675 | 9538 | 2275 | 03:00:00 | 667 | 670 | 2.5 | 93 |
| 12 | 5 | 8 | 12 | 94942 | 8235 | 8072 | 975 | 03:00:00 | 542 | 549.5 | 6.6 | 82 |
| 13 | 4 | 16 | 10 | 92060 | 11084 | 10954 | 1777 | 03:00:00 | 1030 | 1073.5 | 25.9 | 115 |
| 14 | 6 | 8 | 10 | 98336 | 9642 | 9466 | 3731 | 03:00:00 | 651 | 669.8 | 11.4 | 95 |
| 15 | 5 | 12 | 8 | 97426 | 13575 | 13428 | 2878 | 03:00:00 | 884 | 912.2 | 13.1 | 111 |
| 16 | 4 | 16 | 8 | 107096 | 16652 | 16520 | 4302 | 03:00:00 | 1112 | 1130.8 | 8.9 | 113 |
| 17 | 4 | 12 | 12 | 130400 | 12492 | 12348 | 2440 | 03:00:00 | 690 | 710.8 | 17.6 | 89 |
| 18 | 4 | 16 | 12 | 230784 | 21388 | 21228 | * | 03:00:00 | 1026 | 1071.1 | 34.4 | 115 |
| 19 | 5 | 16 | 10 | 263700 | 26385 | 26200 | * | 03:00:00 | 1379 | 1417.8 | 20.1 | 137 |
| 20 | 6 | 12 | 12 | 313236 | 22338 | 22116 | * | 03:00:00 | 1045 | 1080.6 | 22.4 | 124 |

* No output is generated

## 4.6 Summary

This chapter describes QCASP, a mathematical formulation of the combined problem of Quay Crane Assignment and Quay Crane Scheduling. It is a mixed integer programming model which allows quay cranes to move between two holds of the same ship and between two holds on different vessels. The time it takes for these cranes to move between holds is taken into account in the optimisation process. Interference between quay cranes is avoided by introducing non-interference constraints which take into account both the potential interference of tasks on the same vessel and those on different vessels. The Branch-and-Cut algorithm as implemented in CPLEX 12.6 has been used to find the optimal solutions of relatively small instances of QCSAP. It cannot cope with larger instances of the problem which are of practical size. GA, however, coped well with all problems. It required almost the same CPU time for all problems of small size and CPU times of the same magnitude for the larger instances. Moreover, on most of the 17 instances that CPLEX managed to solve, GA also found the optimum. Overall the average discrepancy between the objective function values of the solutions found by GA is 0.4. This shows that GA, while substantially more efficient than B&C, it is also quite robust on most instances considered as this average discrepancy shows. The combined model presented here is obviously the way forward as it is more likely to provide better solutions than those found by solving QCAP or QCSP operations problems individually. It is also a substantial improvement on combined variants which do not allow for quay cranes movement between vessels, as found in the literature.

# Chapter 5

# Berth allocation, quay crane assignment and quay crane scheduling problem

## 5.1 Introduction

This chapter is an attempt to integrate all three seaside operations problems into a single model and solve it as such. BAP, QCAP and QCSP have to be considered almost always explicitly in the model container port. As what we have already mentioned in the previous chapters, they have been solved individually. However, given their tight relationship with each other, see Figure 5.1, it is obvious that finding solutions to the individual problems is unlikely to result in an overall optimal solution. Hence the need to solve them in one single integrated model.

In Figure 5.1, the outputs of the berth allocation are the berthing time $T_v$ and the berthing position $P_v$. These outputs of the berthing plan will be used as inputs to the quay cranes assignment problem to determine the number of quay cranes for each vessel. The

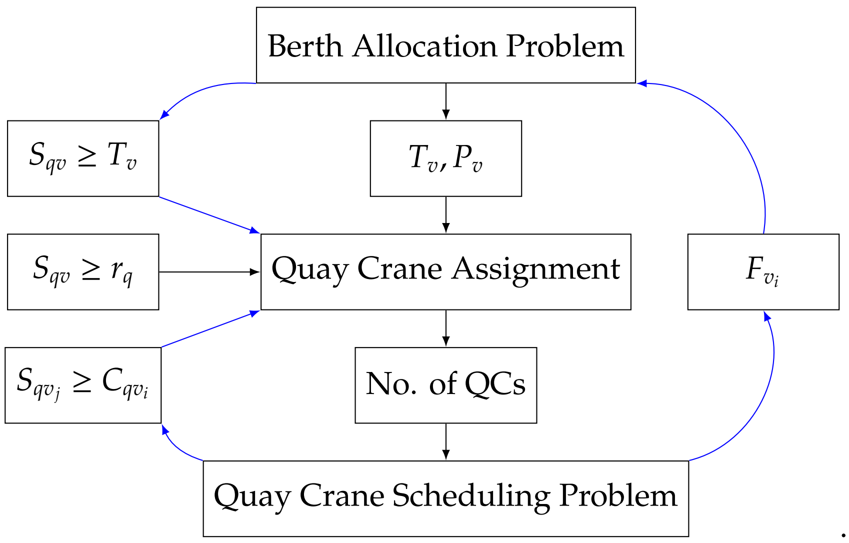


**Figure 5.1:** *Illustration of the relationship among the quayside operations*

starting time of the a quay crane on a vessel $S_{qv}$ should be greater than or equal to its berthing time $T_v$. It should also be greater than or equal to the ready crane time $r_q$ which means the earliest available time of quay crane. After determining the number of quay cranes for each vessel, the solution of the quay crane scheduling problem will return to choose the best sequence in which tasks will be performed by the quay cranes which are assigned to it. Note that the starting time of handling the vessel $S_{qv_j}$ should be greater than or equal to the finishing time $F_{v_i}$ if vessel $v_j$ is moored at the same berthing position as that of vessel $v_i$. The berthing position which will be allocated to the new arrival vessel should be emptied before the new vessel starts processing. As a result, the berth plan depends on the output of the quay crane scheduling problem. The aims of this chapter are:

1. to formulate a mathematical model that combines BAP, QCAP and QCSP in one aggregate formulation (BACASP). In this model, features which are very important to compute an exact solution are considered;

2. to solve realistic instances of the problem using an adapted variant of GA.

The chapter is organized as follows: The proposed mathematical model is given in Section 5.2. Section 5.3 provides an illustration of the proposed mathematical model . In Section 5.4, a solution approach, namely a meta-heuristic based on GA is presented. Section 5.5 describes the computational experience of CPLEX and GA. The summary is given in Section 5.6.

## 5.2 Mathematical formulation

This section describes a mixed integer programming model which combines the three seaside port operations namely berth allocation, quay cranes assignment, and quay crane scheduling which, traditionally, are considered piecemeal. This combined model is then solved to find the best location and the optimal time for berthing for each vessel that arrives at the container terminal. When the vessel is moored at its preferred position, the distance to transfer the containers from the vessel to the storage area is minimized, thus saving on the overall processing time of the vessel. An important characteristic of this model is that the number of QC's assigned to any vessel is not fixed since these QC's are allowed to move between vessels. This, allows a full usage of every quay crane. The solution also includes the optimal sequence in which to perform every task on the vessel; this is the so called quay crane scheduling problem or QCSP.

The model itself has features which are not commonly represented in existing models. Travelling times of a quay crane between two holds on the same ship and between two holds on different vessels have also been considered. Quay crane interference avoidance is explicitly represented when the quay cranes operate on the same vessel and when quay cranes are on different vessels. Another advantage of this model is that it allows a quay crane that has become idle (finished its work) to move from its current vessel to another even if overall the current vessel to which it has been allocated is still being processed. In practice, the quay cranes do not always start their works from the same point and at the same time. For this reason a quay crane ready time has been considered in the mathematical model. The initial position for each quay crane is taken into account in order to compute the exact time of quay crane travelling. The precedence and simultaneity constraints are taken in to consideration as well.

### 5.2.1 Assumptions

Consider a continuous container terminal with fixed length where vessels can moor at a preferable place when possible. Now assume that:

1- Each vessel is divided longitudinally into bays; each bay has the same length. Thus, the length of vessels is in number of bays (bay lengths);

2- The safety distance between each pair of adjacent quay cranes depends on the width of a bay;

3- Each segment of the continuous wharf can handle one vessel at a time;

4- Once a quay crane starts processing a task, it leaves only when it has finished the workload of this bay;

5- Any vessel can be processed in any space of the wharf depending on their arrival time and the available terminal;

6- Quay cranes are on the same rail and thus cannot cross over each other;

7- Some tasks must be performed before others and there are some tasks that cannot be performed simultaneously.

### 5.2.2 Indices

| | |
|---|---|
| $V$ | Number of vessels $v(i, j = 1, 2, ..., V)$. |
| $B$ | Number of tasks on the vessels $b(i, j = 1, 2, ..., B)$. |
| $Q$ | Number of quay cranes $q(i, j = 1, 2, ..., Q)$. |

### 5.2.3 Parameters

| | |
|---|---|
| $p_b^v$ | Time required to perform task $b$ on vessel $v$. |
| $l_b^v$ | Location of task $b$ on vessel $v$ expressed by the ship bay number on vessel $v$. |
| $r_q$ | Earliest available time of the $q^{th}$ quay crane. |
| $l_0^q$ | Initial location of the $q^{th}$ quay crane which is relative to the ship-bay number. |
| $t_{b_ib_j}^{qv}$ | Travel time of the $q^{th}$ quay crane from the location $l_{b_i}^v$ of task $b_i$ to the location $l_{b_j}^v$ of task $b_j$ on vessel $v$. $t_{b_0b_j}^{qv}$ represents the travel time from the initial position $l_q^v$ of the $q^{th}$ quay crane to the location $l_{b_j}^v$ of the task $b_j$ on vessel $v$. Note that the value of $t$ represents the travelling time between two adjacent bays. |
| $a_v$ | Estimated arrival time for vessel $v$. |
| $d_v$ | Requested departure time for vessel $v$. |
| $\hat{P}_v$ | Preferred berth position of vessel $v$. It is determined by the position of yard storage areas allocated to vessel $v$. $\hat{P}_v$ reflects that the berth position has the shortest distance to the allocated yard storage area for vessel $v$. |
| $U_v$ | Distance cost of vessel $v$. If vessel $v$ moors at $\hat{P}_v$, the transportation cost is the lowest based on the distance cost due to the vessel being moored at a place |

|  |  |
|---|---|
|  | with distance deviation . |
| $L_v$ | Length of vessel $v$. |
| $W$ | Length of the wharf. |
| $W_v$ | Tardiness cost of vessel $v$ per unit time. |
| $R_v$ | Earliness income of vessel $v$ per unit time. |
| $\Psi$ | Set of pairs of tasks that cannot be performed simultaneously; when tasks $b_i$ and $b_j$ cannot be performed simultaneously $((b_i, b_j) \in \Psi)$. |
| $\Phi$ | Set of ordered pairs of tasks for which there is a precedence relationship; when task $b_i$ must precede task $b_j$ $((b_i, b_j) \in \Phi)$. |
| $M$ | Arbitrary large positive number. |

### 5.2.4 Binary decision variables

$$X^{qv}_{b_i b_j} = \begin{cases} 1 & \text{if the } q^{th} \text{ quay crane performing task } b_j \text{ immediately after performing task } b_i \text{ on} \\ & \text{vessel } v. \\ 0 & \text{otherwise.} \end{cases}$$

Tasks $b_0$ and $b_{B_v+1}$ are considered as the dummy initial and final states of each quay crane, respectively. Thus, when task $b_j$ is the first task of the $q^{th}$ quay crane then $X^{qv}_{b_0 b_j} = 1$. In addition, when task $b_j$ is the last task of the $q^{th}$ quay crane then $X^{qv}_{b_j b_{B_v+1}} = 1$.

$$Z^{v}_{b_ib_j} = \begin{cases} 1 & \text{if task } b_j \text{ starts later than the finishing of task } b_i \text{ on vessel } v. \\ 0 & \text{otherwise.} \end{cases}$$

$$Y^{q}_{v_iv_j} = \begin{cases} 1 & \text{if the } q^{th} \text{ quay crane is assigned to vessel } v_j \text{ right after finishing its tasks on} \\ & \text{vessel } v_i. \\ 0 & \text{otherwise.} \end{cases}$$

$$\delta_{v_iv_j} = \begin{cases} 1 & \text{if the processing of vessel } v_j \text{ starts later than the finishing of vessel } v_i. \\ 0 & \text{otherwise.} \end{cases}$$

$$\sigma_{v_iv_j} = \begin{cases} 1 & \text{if vessel } v_i \text{ is located nearer one end of the wharf than vessel } v_j. \\ 0 & \text{otherwise.} \end{cases}$$

$$\beta^{v_iv_j}_{b_ib_j} = \begin{cases} 1 & \text{if task } b_j \text{ on vessel } v_j \text{ starts later than the finish time } D \text{ of task } b_i \text{ on vessel } v_i. \\ 0 & \text{otherwise.} \end{cases}$$

$$\alpha^{v_iv_j}_{b_ib_j} = \begin{cases} 1 & \text{if task } b_i \text{ on vessel } v_i \text{ is located below task } b_j \text{ on vessel } v_j. \\ 0 & \text{otherwise.} \end{cases}$$

### 5.2.5 Continuous decision variables

| | |
|---|---|
| $T_v$ | Berthing time of vessel $v$. |
| $P_v$ | Berthing position of vessel $v$. |
| $A_v$ | Tardiness of vessel $v$. |

| | |
|---|---|
| $E_v$ | Earliness of vessel $v$. |
| $C_{qv}$ | Completion time of the $q^{th}$ quay crane on vessel $v$. |
| $F_v$ | Finishing time of vessel $v$. |
| $D^v_{b_i}$ | Completion time of task $b_i$ on vessel $v$. |
| $S_{qv}$ | Starting time of the $q^{th}$ quay crane on vessel $v$. |

### 5.2.6 The mathematical model

$$\min Z = \sum_{v=1}^{V} W_v A_v - \sum_{v=1}^{V} R_v E_v + \sum_{v=1}^{V} U_v |P_v - \hat{P}_v| \tag{5.1}$$

s.t

$$d_v - F_v = E_v - A_v \qquad \forall v \tag{5.2}$$

$$F_{v_i} \leq T_{v_j} + M(1 - \delta_{v_i v_j}) \qquad \forall v_i, v_j \tag{5.3}$$

$$P_{v_i} + L_{v_i} \leq P_{v_j} + M(1 - \sigma_{v_i v_j}) \qquad \forall v_i, v_j \tag{5.4}$$

$$\sigma_{v_i v_j} + \sigma_{v_j v_i} + \delta_{v_i v_j} + \delta_{v_j v_i} \geq 1 \qquad \forall v_i, v_j \tag{5.5}$$

$$a_v \leq T_v \qquad \forall v \tag{5.6}$$

$$P_v + L_v \leq W \qquad \forall v \tag{5.7}$$

$$\sum_{v_j=1}^{V} Y^q_{v_0 v_j} = 1 \qquad \forall q \tag{5.8}$$

$$\sum_{v_i=1}^{V} Y^q_{v_i(V+1)} = 1 \qquad \forall q \tag{5.9}$$

$$\sum_{v_j=1}^{V+1} Y^q_{v v_j} - \sum_{v_j=0}^{V} Y^q_{v_j v} = 0 \qquad \forall v, q \tag{5.10}$$

$$\sum_{v_i=0}^{V} \sum_{q=1}^{Q} Y^q_{v_i v_j} \geq 1 \qquad \forall v_j \tag{5.11}$$

$$S_{qv} \geq r_q - M(1 - Y^q_{v_0 v}) \qquad \forall v, q \tag{5.12}$$

$$S_{qv} \geq T_v - M(1 - \sum_{v_j=1}^{V+1} Y^q_{v v_j}) \qquad \forall v, q \tag{5.13}$$

$$S_{qv_j} \geq C_{qv_i} - M(1 - Y^q_{v_i v_j}) \qquad \forall v_i, v_j, q \tag{5.14}$$

$$\sum_{b_j=1}^{B_v} X^{qv}_{b_0 b_j} = \sum_{v_i=0}^{V} Y^q_{v_i v} \qquad \forall v, q \tag{5.15}$$

$$\sum_{b_j=1}^{B_v} X^{qv}_{b_j b_{B_v+1}} = \sum_{v_i=0}^{V} Y^q_{v_i v} \qquad \forall v, q \tag{5.16}$$

$$\sum_{b_j=1}^{B_v+1} X^{qv}_{b b_j} - \sum_{b_j=1}^{B_v} X^{qv}_{b_j b} = 0 \qquad \forall b, v, q \tag{5.17}$$

$$\sum_{q=1}^{Q} \sum_{b_i=0}^{B_v} X^{qv}_{b_i b_j} = 1 \qquad \forall b_j, v \tag{5.18}$$

$$\sum_{b_i=0}^{B_v} \sum_{b_j=1}^{B_v+1} X^{qv}_{b_i b_j} \leq M \sum_{v_i=0}^{V} Y^q_{v_i v} \qquad \forall v, q \tag{5.19}$$

$$D^v_{b_i} + t^{qv}_{b_i b_j} + p^v_{b_i} - D^v_{b_j} \leq M(1 - X^{qv}_{b_i b_j}) \qquad \forall b_i, b_j, v, q \tag{5.20}$$

$$S_{qv} + D^v_{b_j} + t^{qv}_{b_0 b_j} - p^v_{b_j} \leq M(1 - X^{qv}_{b_0 b_j}) \qquad \forall b_j, v, q \tag{5.21}$$

$$D^v_{b_j} - C_{qv} \leq M(1 - X^{qv}_{b_j b_{B_v+1}}) \qquad \forall b_j, v, q \tag{5.22}$$

$$C_{qv} - F_v \leq M(1 - \sum_{v_j=1}^{V+1} Y^q_{v v_j}) \qquad \forall v, q \tag{5.23}$$

$$D^v_{b_i} + p^v_{b_j} \leq D^v_{b_j} \qquad \forall (b_i, b_j) \in \Phi_v, v \tag{5.24}$$

$$D^v_{b_i} - D^v_{b_j} + p^v_{b_j} \leq M(1 - Z^v_{b_i b_j}) \qquad \forall b_i, b_j, v \tag{5.25}$$

$$Z^v_{b_i b_j} + Z^v_{b_j b_i} = 1 \qquad \forall (b_i, b_j) \in \Psi_v, v \tag{5.26}$$

$$\sum_{\theta=0}^{Q} \sum_{\kappa=0}^{B_v} X^{\theta v}_{\kappa b_j} - \sum_{\theta=0}^{Q} \sum_{\kappa=0}^{B_v} X^{\theta v}_{\kappa b_i} \leq M(Z^v_{b_i b_j} + Z^v_{b_j b_i}) \qquad \forall b_i, b_j; l_{b_i} < l_{b_j}; \forall v, q \tag{5.27}$$

$$D^{v_i}_{b_i} - D^{v_j}_{b_j} + p^{v_j}_{b_j} \leq M(1 - \beta^{v_i v_j}_{b_i b_j}) \qquad \forall b_i, b_j, v_i, v_j \tag{5.28}$$

$$P_{v_i} + l^v_{b_i} \leq P_{v_j} + l^v_{b_j} + M(1 - \alpha^{v_iv_j}_{b_ib_j}) \qquad \forall b_i, b_j, v_i, v_j \tag{5.29}$$

$$\beta^{v_iv_j}_{b_ib_j} + \beta^{v_jv_i}_{b_jb_i} + \alpha^{v_jv_i}_{b_jb_i} \geq \sum_{\kappa=0}^{B_v} X^{q_iv_i}_{\kappa b_i} + \sum_{\kappa=0}^{B_v} X^{q_jv_j}_{\kappa b_j} - 1 \qquad \forall b_i, b_j, v_i, v_j, q_i, q_j; v_j \neq v_i; q_i < q_j \tag{5.30}$$

$$X^{qv}_{b_ib_j}, Z^v_{b_ib_j}, Y^q_{v_iv_j}, \delta_{v_iv_j}, \sigma_{v_iv_j}, \alpha^{v_iv_j}_{b_ib_j}, \beta^{v_iv_j}_{b_ib_j} \in \{0, 1\} \qquad \forall b_i, b_j, v_i, v_j, v, q \tag{5.31}$$

$$C_{qv}, F_v, D^v_{b_j}, P_v, T_v \geq 0 \qquad \forall b_j, v, q \tag{5.32}$$

In the objective function (5.1), the first term $\sum_{v=1}^{V} W_vA_v$ represents the tardiness cost if the departure time of the vessel is later than its due time. The second term $\sum_{v=1}^{V} R_vE_v$ represents the income from earliness if the finishing time of vessel is earlier than its due time. The last term $\sum_{v=1}^{V} U_v|P_v - \hat{P}_v|$ represents the cost incurred if the vessel is moored at an undesired berthing position. Constraints (5.2) determine if the vessel has earliness or tardiness depending on the difference between the departure (due) time of the vessel and the finishing time of this vessel.

In constraints (5.3), $\delta_{v_iv_j}$ is defined as follows. $\delta_{v_iv_j} = 1$ if the finishing time of vessel *i* is less than or equal to the berthing time of vessel *j*; 0 if the finishing time of vessel *i* is greater than the berthing time of vessel *j*. See Figures 3.1 and 3.2 for illustration.

In constraints (5.4), $\sigma_{v_iv_j}$ is defined as follows. $\sigma_{v_iv_j} = 1$ if the berthing position of vessel *i* plus the length of vessel *i* is less than or equal to the berthing position of vessel *j*; 0 if the berthing position of vessel *i* plus the length of vessel *i* is greater than the berthing position of vessel *j*. See Figures 3.3 and 3.4 for illustration.

Constraints (5.5) ensure that the overlaps among vessels do not exist in the two dimensions (time and location) depending on the values of $\delta_{v_iv_j}$ and $\sigma_{v_iv_j}$. Constraints (5.6) state that the vessels cannot moor before their arrivals. Constraint (5.7) implies that the berthing position plus the length of vessel cannot exceed the range of the wharf.

Constraints (5.8) and (5.9) respectively select the first and the last ships for each quay crane. Constraints (5.10) guarantee that ships are processed in a well-defined sequence. Constraints (5.11) force every quay crane available at the terminal container to handle at least one ship. This set of constraints is not really necessary since one can imagine that a large number of quay cranes is available while only a few vessels are to be handled. However, this situation is unlikely and in general only a few quay cranes are available to handle a large number of vessels. Therefore, we do not want any of the quay cranes to be idle. Hence the need for these constraints.

Constraints (5.12) ensure that the starting time of the earliest vessel that is to be done by the $q^{th}$ quay crane should be after the ready time of the quay crane depending on the value of $Y^{q}_{v_0 v_j}$. Note that the vessel $v_o$ is a fake vessel. Constraints (5.13) ensure that the starting time of the $q^{th}$ quay crane on the vessel $v$ is greater than or equal to the berthing time if the $q^{th}$ quay crane is assigned to the vessel. Constraints (5.14) ensure that the starting time of the $q^{th}$ quay crane on vessel $v_i$ should be no earlier than the finishing time of its predecessor vessel $v_i$.

Constraints (5.15) ensure that if a quay crane is allocated to a vessel, then it will start its processing from one task on that vessel. Constraints (5.16) ensure that if a quay crane is allocated to a vessel, then it will finish its processing on one task on that vessel. Constraints (5.17) show a flow balance ensuring that tasks are performed in a well-defined sequence. Constraints (5.18) ensure that every task on each vessel must be completed by exactly one quay crane. Constraints (5.19) ensure that if a quay crane is not assigned to a vessel, the tasks on this vessel will not be performed by this quay crane. Constraints (5.20) simultaneously determine the completion time for each task and eliminates sub-tours; sub-tours here are the looping on tasks which have already been done. To illustrate, let Task 1, Task 2, and Task 3, be carried out in this order. A sub-tour would be to do Task 1, Task 2, Task 3, and Task 2 again, for instance. Constraints (5.21) define the quay crane operation starting time. The completion time of each quay crane is computed by constraints (5.22). Constraints (5.23) determine the finishing time of each vessel.

When required, constraints (5.24) force task $i$ to be completed before task $j$ for all the tasks which are in the set $\Phi$. Constraints (5.25) define $Z_{ij}^{v}$ such that $Z_{ij}^{v} = 1$ when the operation of task $j$ on vessel $v$ starts after the operation for task $i$ completed; and 0 otherwise. Constraints (5.26) ensure that the pair of tasks that are members of the set $\Psi$ will not be handled simultaneously.

Constraints (5.27) prevent interference between quay cranes. Suppose that tasks $i$ and $j$ are performed simultaneously and $l_i < l_j$. This means that $Z_{ij}^{v} + Z_{ji}^{v} = 0$. Note that both quay cranes and tasks are ordered in an increasing order of their relative location in the direction of increasing ship-bay number. Suppose that, for $q_1 < q_2$, quay crane $q_1$ performs tasks $j$ and quay crane $q_2$ performs task $i$. Then, interference between quay cranes $q_1$ and $q_2$ results. However, in such a case, $\sum_{\theta=1}^{q_1} \sum_{\kappa=0}^{B_v} X_{\kappa b_j}^{\theta v} - \sum_{\theta=1}^{q_1} \sum_{\kappa=0}^{B_v} X_{\kappa b_i}^{\theta v} = 1$, it cannot be allowed because of constraints (5.27), and then we have $Z_{ij}^{v} + Z_{ji}^{v} = 0$.

In constraints (5.28), $\beta_{b_i b_j}^{v_i v_j}$ is defined as follows. $\beta_{b_i b_j}^{v_i v_j} = 1$ if the finishing time $D$ of task $b_i$ on vessel $v_i$ plus the processing time of task $b_j$ on vessel $v_j$ is less than or equal to the finishing time $D$ of task $b_j$ on vessel $v_j$; 0 if the finishing time of task $b_i$ on vessel $v_i$ plus the processing time of task $b_j$ on vessel $v_j$ is greater than the finishing time of task $b_j$ on vessel $v_j$. See Figures 4.1 and 4.2 for illustration.

In constraints (5.29), $\alpha^{v_i v_j}_{b_i b_j}$ is defined as follows. $\alpha^{v_i v_j}_{b_i b_j} = 1$ if the berthing position $P$ of vessel $v_i$ plus the location $l$ of task $b_i$ on that vessel is less than or equal to the berthing position $P$ of vessel $v_j$ plus the location $l$ of task $b_j$ on that vessel; 0 if the berthing position of vessel $v_i$ plus the location of task $b_i$ on that vessel is greater than the berthing position of vessel $v_j$ plus the location of task $b_j$ on that vessel. See Figures 4.3 and 4.4 for illustration.

Constraint (5.30) prevents the interference between the quay cranes handling two different vessels depending on the value of $\beta^{v_i v_j}_{b_i b_j}$ and $\alpha^{v_i v_j}_{b_i b_j}$.

The size of the above model is as follows:

The number of binary variables is equal to $VB(2VB-B-1)+VQ(B^2+2B+V+2)+2V(V-1)$,

The number of integer variables is equal to $5V + VB + 2VQ$,

The number of constraints is equal to

$VBQ(VBQ - BQ - VB + 3B + 1) + VB(B + 2V - 2) + VQ(V + 6) + 3V^2 + 2B^2 - 2B + V + 2Q$.

## 5.3 Numerical examples

To illustrate the case of better plans, consider the situation in which two quay cranes are available to handle two vessels with data as given in Table 5.1, and each vessel has two tasks to process. Note that without considering the berth plan the vessels can be moored at any possible place at quayside. In this case the cost of handling these vessels may increase depending on how far away the vessels are berthing from their preferred berthing position.

When a fixed number of quay cranes is allocated to every vessel during the whole processing period, the optimal working plan is described in Figure 5.2. Even though quay

**Table 5.1:** *Input data of Example 5.1*

| | | |
|---|---|---|
| Ready (crane) | 0 | 0 |
| Initial location (crane) | 11 | 14 |
| Processing time of tasks ,vessel 1 | 85 | 27 |
| Processing time of tasks ,vessel 2 | 18 | 33 |
| Location task, vessel 1 | 1 | 3 |
| Location task, vessel 2 | 1 | 4 |
| Expected departure time of vessel | 85 | 33 |
| Arrival time | 0 | 0 |
| Preferred berthing position | 10 | 15 |
| Tardiness cost (per unit time) | 5 | 1 |
| Earliness income (per unit time) | 1 | 1 |

crane 2 finished its work on vessel 2 at (2+18+3+33)=56, it is not allowed to move away from vessel 2. This wastes the effective working time of this quay crane which will result into a sub-optimal solution. In contrast, in our model, a variable number of quay cranes is used during the processing period, which allows quay crane 2 to move from vessel 2 to perform other tasks as shown in Figure 5.3. As a result, the finishing time of vessel 1 will be 89 time units which is earlier than in the previous plan i.e. 114 time units, due to making the best use of both quay cranes. Note that we only allow the quay crane to move after it has finished its work on vessel 2.

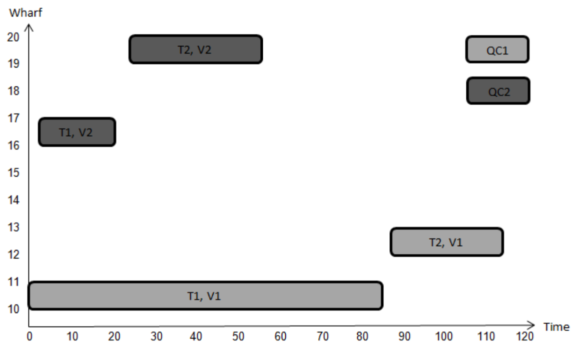


**Figure 5.2:** *Model solution for fixed number of QC's for example 5.1*

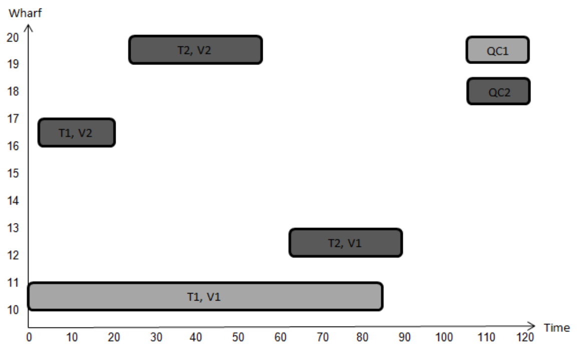


**Figure 5.3:** *Suggested model solution for example 5.1*

Our model also allows quay cranes to share tasks on the same vessel to which they are allocated. Consider the input data for two vessels arriving at a container terminal as given in Table 5.2.

**Table 5.2:** *Input data of Example 5.2*

| | | |
|---|---|---|
| Ready (crane) | 0 | 0 |
| Initial location (crane) | 22 | 24 |
| Processing time of tasks,vessel 1 | 28 | 22 |
| Processing time of tasks,vessel 2 | 39 | 34 |
| Location task, vessel 1 | 1 | 2 |
| Location task, vessel 2 | 1 | 2 |
| Expected departure time of vessel | 28 | 39 |
| Arrival time | 0 | 0 |
| Preferred berthing position | 20 | 25 |
| Tardiness cost (per unit time) | 1 | 1 |
| Earliness income (per unit time) | 1 | 1 |

In Example 5.2, the finishing time for vessel 1 is 51 and the finishing time for vessel 2 is 76 (see Figure 5.4). Since we allow quay cranes to move between vessels if no interference constraints are violated, the solution to our model is the finishing time for vessel 1 which is equal to 29, and the finishing time for vessel 2 which is equal to 73, as can be seen in Figure 5.5.

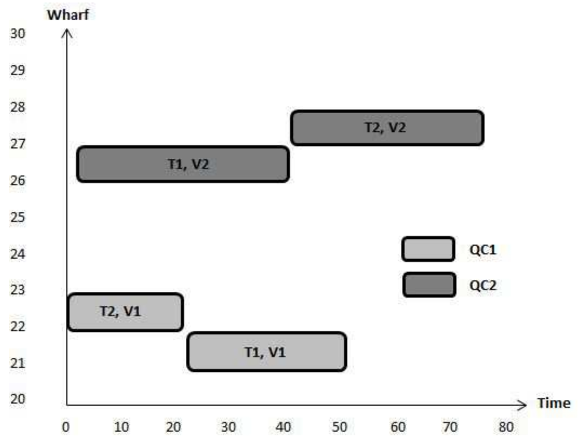


**Figure 5.4:** *Model solution for fixed number of QC's for example 5.2*

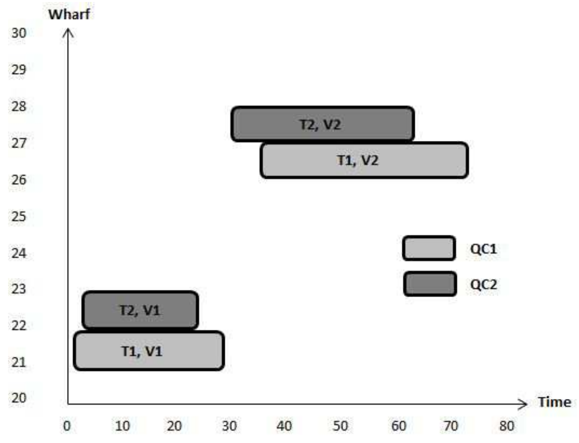


**Figure 5.5:** *Suggested model solution for example 5.2*

## 5.4 Application of GA to BACASP

### 5.4.1 Solution representation: chromosome

Here, a solution or chromosome is a strand of genes made of three parts. The first part represents berthing times ($T_v$) of vessels while the second represents berthing positions ($P_v$). The populations for berthing time and position are generated randomly but within the feasible solution set defined by the constraints (5.6), (5.7) and (5.32) for each solution $((T_v),(P_v))$. The third part here represents a sequence of holds (tasks) for all docked vessels. The value of a gene is randomly picked from the index set of all holds; it cannot, therefore, be duplicated, i.e. each gene is unique. Each chromosome (third part) consists of $v \times b$ genes, where $v$ represents the number of vessels and $b$ the number of tasks on each vessel. The chromosomes are character rather than binary strings.

A simple chromosome for the case of three vessels, each with two tasks, is illustrated in Figure 5.6. Here, the genes (1-3) represent the berthing time for the three vessels, the genes (4-6) represent the berthing position for the three vessels, and the genes (7-12) represent the sequence of performing the tasks on these three vessels.

| Chromosome | 42 | 37 | 65 | 213 | 185 | 370 | 1 | 4 | 3 | 6 | 5 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T_i$&$P_i$ | $T_1$ | $T_2$ | $T_3$ | $P_1$ | $P_2$ | $P_3$ | Sequence of tasks | | | | | |

**Figure 5.6:** *BACASP chromosome representation*

Based on the sequence of tasks for all vessels represented by the chromosome, a quay crane schedule can be constructed using the steps that are the extension of the procedure proposed by Lee et al. [45] used for each vessel separately. See quay crane scheduling procedure 4.4.1 for illustration purposes.

### 5.4.2 Solution validation

In addition to the three conditions that are presented in section 4.4.2, there is another condition that should be considered when solving this type of problem. Each generated random solution is checked against constraints (5.3)-(5.5) to see that there is no overlapping of ships in time and space (berthing position). A solution that satisfies these constraints is accepted. Otherwise it is accepted after addition of a penalty term to its objective function value $Z$. The objective function value and the penalty term when used form the fitness of that solution. The penalty term in the fitness function gradually removes infeasible solutions from subsequent generations. If any one of constraints (5.3)-(5.5), (5.24), (5.26), (5.27), (5.30) is violated or any combination are violated, the generated chromosomes are discarded by putting a high penalty to their fitness values.

### 5.4.3 Evaluation of fitness

The objective of the BACASP is to minimise the tardiness of vessels, to maximise the earliness and to find the optimum berthing time and position for each vessel that arrives at the container terminal. The completion time of each quay crane can be computed by summing up the processing time of all the tasks that have been performed by this quay crane plus the travel time which it takes to move from one hold to another. The tardiness of each vessel can be computed by subtracting the finishing time from the expected departure time and the earliness for each vessel can be computed by subtracting the expected departure time from the finishing time if the vessel finished its handling before its expected departure time. The finishing time of vessel represents the maximum processing time of the vessel required by the quay cranes assigned to it. In addition, there is the berthing cost which is due to how far the vessel is moored away from its preferring berthing position. The fitness function used by the GA in MATLAB is the same as the objective function of the mathematical model. Thus, the fitness value of a chromosome is calculated by Equation (5.33).

$$Fitness(chromosome) = \frac{1}{\sum_{v=1}^{V} W_v A_v - \sum_{v=1}^{V} R_v E_v + \sum_{v=1}^{V} U_v |P_v - \hat{P}_v|} \tag{5.33}$$

### 5.4.4 Generating next populations

From an initial population and the fitness function value of each individual, we will use the genetic operators of crossover, mutation, and reproduction to generate a new population. Before applying the genetic operator, we must select suitable parents using a selection process.

**Crossover operator** The way the recombination is implemented is as follows. If the specified genes are related to the variables $T_i$ and $P_i$, the crossover that is implemented in RBAP (please see 3.4.4) will be applied . If the specified genes are related to the third part of a chromosome (sequence tasks), the 'Order Crossover' of Cheng and Gen [9], (section 4.4.4), is implemented. For illustration purposes, please see Figures 5.7 and 5.8.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parent1 | 7 | 12 | 25 | **50** | **33** | **70** | **20** | 95 | 7 | 12 | 5 | 3 | 6 | 1 | 10 | 8 | 11 | 2 | 4 | 9 |
| Offspring1 | 7 | 12 | 25 | **49** | **38** | **72** | **28** | 95 | **7** | **12** | **5** | **3** | 8 | 2 | 6 | 4 | 11 | 9 | 10 | 1 |
| Parent2 | 9 | 15 | 22 | **48** | **62** | **80** | **60** | 84 | 8 | 2 | 5 | 6 | 7 | 4 | 11 | 9 | 12 | 3 | 10 | 1 |

**Figure 5.7:** *BACASP offspring 1*

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parent1 | 7 | 12 | 25 | **50** | **33** | **70** | **20** | 95 | 7 | 12 | 5 | 3 | 6 | 1 | 10 | 8 | 11 | 2 | 4 | 9 |
| Offspring2 | 9 | 15 | 22 | **48** | **56** | **78** | **52** | 84 | **8** | **2** | **5** | **6** | 7 | 12 | 3 | 1 | 10 | 11 | 4 | 9 |
| Parent2 | 9 | 15 | 22 | **48** | **62** | **80** | **60** | 84 | 8 | 2 | 5 | 6 | 7 | 4 | 11 | 9 | 12 | 3 | 10 | 1 |

**Figure 5.8:** *BACASP offspring 2*

**Mutation operator** The mutation presented here is that, if the specified gene is related to the variables $T_i$, a random integer number in $[a_i, LN]$, where $LN$ is a large number, and if it is related to the variables $P_i$ , a random integer number in interval $[0, W - L_i]$, it replaces the previous value of that gene. In this case, every new chromosome will satisfy the conditions of (5.6), (5.7) and (5.32). If it is related to the third part of a chromosome, two genes from the chromosome are randomly selected and then swapped. Figure 5.9 illustrates the mutation operator.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | **12** | 25 | 50 | 33 | 70 | **20** | 95 | 8 | 4 | **3** | 6 | 2 | 7 | **9** | 1 |
| 7 | **15** | 25 | 50 | 33 | 70 | **50** | 95 | 8 | 4 | **9** | 6 | 2 | 7 | **3** | 1 |

**Figure 5.9:** *BACASP mutation operator*

## 5.5 Computational experiments

Twenty instances of the above mathematical model of BACASP with different numbers of vessels, tasks, and quay cranes have been solved using Branch-and-Cut (B&C) as implemented in CPLEX, and GA. All instances have randomly generated processing times of tasks for each hold from the uniform distribution $U(10,50)$. All experiments have been performed on a PC with Intel Core i5 and 3.20 GHz CPU with 8 GB RAM running Windows 7 Operating System. The 20 problems and their corresponding results are presented in Table 5.3, containing small problems 1 through 10 and Table 5.4, containing large problems 11 through 20. Unlike CPLEX, GA is coded in Matlab and solves all 20 instances.

In the instances considered, the number of constraints, the number of decision variables and the B&C computational time grow exponentially with the increase in the number of vessels, tasks and quay cranes. However, the CPU time required by GA does not grow that fast with the increase in the problem size. CPLEX did not solve some of the larger size instances, shown in Table 5.4. GA, on the other hand, finds the optimal or near optimal solutions for all the instances in reasonable CPU times (see column 13 of Tables 5.3 and 5.4).

For the small size problems of Table 5.3, the GA parameters of population size, rate of crossover, rate of mutation and the maximum number of generations are set as 200, 0.5, 0.4 and 500, respectively. These parameters have been set by experimentation and other experiences that can be found in the literature. Moreover, 20 trials were conducted to solve each instance using GA to mitigate the randomness of the algorithm. CPLEX solved problems 1 to 10 which are relatively small in size. Six problems were solved in acceptable times, but problems 6, 7, 9 and 10 required 24, 22, 24 and 60 hours, respectively. Note that the solution of instances 6, 9 and 10 stopped automatically because there is not enough memory. These times are hardly acceptable in the context of container terminal operations. The average objective function values of the solutions returned by GA and their standard deviations are recorded in columns 11 and 12 of Table 5.3. B&C managed to solve only few of the small size instances. The average gap between CPLEX and GA returned objective values is about 0.3.

**Table 5.3:** *Computational results for small scale instances of BACASP*

| No. | Problem Information | | | Problem Size | | | CPLEX | | GA | | | | Gap(%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Vessels | Tasks | Q.Cranes | Constraints | Dec.Vars | Int.Vars | Obj.Val | CPU(hh:mm:ss) | Best Obj.Val | Mean | St.Dev | CPU(s) | |
| 1 | 2 | 2 | 2 | 120 | 90 | 66 | 67* | 00:00:01 | 67 | 67 | 0.0 | 37 | 0.0 |
| 2 | 2 | 4 | 4 | 616 | 322 | 288 | 58* | 00:02:21 | 58 | 58 | 0.0 | 41 | 0.0 |
| 3 | 2 | 4 | 5 | 822 | 374 | 337 | 470* | 00:03:18 | 470 | 478.5 | 3.6 | 40 | 0.0 |
| 4 | 4 | 2 | 5 | 1038 | 444 | 373 | 1250* | 00:05:33 | 1251 | 1251 | 0.0 | 42 | 0.08 |
| 5 | 3 | 3 | 5 | 1123 | 453 | 398 | 795* | 00:41:25 | 805 | 805 | 0.0 | 41 | 1.25 |
| 6 | 3 | 4 | 4 | 1325 | 597 | 544 | 275 | 24:57:28 | 258 | 269 | 8.8 | 43 | – |
| 7 | 4 | 3 | 4 | 1412 | 624 | 556 | 1615* | 22:23:14 | 1636 | 1636 | 0.0 | 43 | 1.3 |
| 8 | 4 | 2 | 8 | 2136 | 612 | 520 | 1165* | 00:01:12 | 1166 | 1171.2 | 4.3 | 40 | 0.08 |
| 9 | 3 | 6 | 4 | 2741 | 1161 | 1102 | 771 | 24:20:17 | 755 | 777 | 14.1 | 49 | – |
| 10 | 3 | 5 | 8 | 5920 | 1296 | 1220 | 815 | 60:12:54 | 780 | 819.5 | 23.4 | 48 | – |

* Optimal solution. $Gap = \frac{GAObj.Fun. - CPLEXObj.Fun.}{CPLEXObj.Fun.} * 100$

The rest of the problems, 11 to 20, which are real world instances in Table 5.4, could not be solved with CPLEX in acceptable times (> 100 hours of CPU time). In some cases they could not be solved at all due to the limitations of the computing platform used see the cases 15, 17-20. However, we terminated all these instances after 3 hours. In the case of the large size instances of Table 5.4, population size, crossover and mutation rates, and the maximum number of generations are set as 500, 0.5, 0.4, and 1000, respectively. These parameters have been set by experimentation and other experiences that can be found in the literature. Moreover, 20 trials were conducted to solve each instance using GA to mitigate the randomness of the algorithm. The average objective function values of the solutions returned by GA and their standard deviations are recorded in columns 11 and 12 of Table 5.4.

**Table 5.4:** *Computational results for large scale instances of BACASP*

| No. | Problem Information | | | Problem Size | | | CPLEX | | GA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Vessels | Tasks | Q.Cranes | Constraints | Dec.Vars | Int.Vars | Obj.Val | CPU(hh:mm:ss) | Best Obj.Val | Mean | St.Dev | CPU(s) |
| 11 | 4 | 8 | 6 | 16212 | 3764 | 3662 | 476 | 03:00:00 | 263 | 300.6 | 19.4 | 280 |
| 12 | 5 | 5 | 5 | 7620 | 2175 | 2070 | 1220 | 03:00:00 | 1202 | 1237.4 | 40.4 | 244 |
| 13 | 4 | 10 | 8 | 42072 | 6628 | 6504 | 698 | 03:00:00 | 358 | 453.05 | 38.60 | 363 |
| 14 | 4 | 10 | 10 | 63876 | 7572 | 7434 | 1803 | 03:00:00 | 157 | 180 | 12.2 | 342 |
| 15 | 5 | 8 | 10 | 66535 | 7005 | 6840 | * | 03:00:00 | 138 | 165.4 | 13.3 | 337 |
| 16 | 6 | 6 | 12 | 79188 | 6222 | 6012 | 259 | 03:00:00 | 26 | 35.05 | 7.57 | 296 |
| 17 | 6 | 8 | 10 | 97964 | 9246 | 9046 | * | 03:00:00 | 196 | 226.9 | 22.87 | 356 |
| 18 | 4 | 12 | 12 | 129872 | 11956 | 11796 | * | 03:00:00 | 107 | 131.3 | 11.3 | 382 |
| 19 | 6 | 10 | 12 | 217488 | 15342 | 15108 | * | 03:00:00 | 340 | 378.2 | 18.2 | 436 |
| 20 | 5 | 16 | 12 | 374689 | 28455 | 28232 | * | 03:00:00 | 529 | 572 | 22.2 | 535 |

* No output is generated.

We have experimented on relatively limited size cases with few vessels, quay cranes and tasks. The experimental result shows that the proposed mathematical model is capable of finding the optimal solution for small scale of instances. However, as can be seen, instances of the model grow to large sizes with hundreds of constraints and integer variables. This means that exact solution is computationally expensive. For instance, the problems 6, 7, 9 and 10 in Table 5.3, require over 20 hours of CPU time with CPLEX. Truly practical instances are beyond CPLEX, running on a PC. GA, however, coped well with all problems. It required almost the same CPU time for all problems of small size and CPU times of the same magnitude for the larger instances. Overall the average discrepancy between the objective function values of the solutions found by both algorithms is small. This shows that GA, while vastly more efficient than B&C, is also quite robust on most instances considered as this average discrepancy shows.

## 5.6 Summary

This chapter describes an integrated mathematical model (BACASP) which combines all seaside operations that arise at container terminals. This model is desirable because solving the problems individually and even when combined pairwise, may lead to suboptimal solutions. BACASP is a mixed-integer programming model which is solved using B&C as implemented in CPLEX 12.6. The main contribution of this work is this single extended model, which has features which are not commonly represented in existing models. Quay crane interference avoidance for instance is explicitly represented as constraints (5.27) when the quay cranes operate on the same vessel and as constraints (5.30) when quay cranes are on different vessels. Another advantage of this model is that it allows a quay crane that has become idle (finished its work) to move from its current vessel to another even if overall the current vessel to which it has been allocated is still being processed. Travelling times of a quay crane between two holds on the same ship and between two holds on different vessels have also been considered. The quay crane starting time is considered. In practice, quay cranes do not always start their work at the same time. For this reason quay crane ready times have been considered in the mathematical model. The initial position for each quay crane is taken into account in order to compute the exact time of quay crane traveling. The precedence and simultaneity constraints are taken in consideration as well.

# Chapter 6

# Conclusions and future work

This chapter is in two parts. The first part presents the main conclusions whereas the second part highlights future work and directions for worthwhile research.

## 6.1 Main conclusions

Berth allocation (BAP) is one of the most important operations in container terminals. Determining the optimal berthing time and the best berthing position for vessels arriving at container terminals is essential for the efficient running of the terminals. Quay cranes are the most important equipment used at container terminals; they are very expensive to build, and very difficult to operate. Determining the number of quay cranes for each vessel (QCAP) is an essential operation at container terminals. Finding the optimum sequence in which to perform the tasks (QCSP) to unload/load containers for all vessels moored at the container terminal is the key to forcing the quay cranes to finish their work in optimal time. Scheduling quay cranes by hand is time consuming and potentially inefficient. This thesis aims at improving on that. In it, we propose three mathematical models representing these operations. An effective meta-heuristic for solving these problems is also proposed. Our work provides an insight into how to improve the performance level of the quayside operations in a typical container terminal. It also contributes to filling some of the gaps in the literature that appeared due to recent trends and changes in maritime logistics.

In Chapter 2, a comprehensive literature review of studies of operations at container terminals (seaside operations) is presented. The literature is divided according to the problem that it dealt with, i.e. BAP, QCSP, BACAP, QCASP and BACASP.

In Chapter 3, a new mathematical model of the mixed integer programming type that addresses RBAP is proposed. Its solution helps mitigate the uncertainty in arrival times and handling times of vessels. In addition to the reputation in punctuality, a new weight is proposed as another criterion to estimate the deviation of the handling time for each vessel from the expected finishing time. A time buffer is inserted between two vessels depending on the value of the reputation in punctuality and the proportion of the processing time of each vessel. Instances of this model have been solved with an exact method namely B&C as implemented in CPLEX, and a hybridisation algorithm based on B&C and GA is used to find optimal or near optimal solutions in reasonable times. The numerical results show that the hybridisation solution approach is superior to B&C in that it finds solutions to all problems in acceptable times.

In Chapter 4, a mixed integer programming with non-interference constraints among quay cranes has been introduced to solve QCASP. The B&C method is used to find the exact solution for small scale instances using CPLEX. Note that the number of constraints and decision variables increases exponentially when the number of vessels, tasks, and quay cranes increases. B&C cannot cope with larger instances of the problem which are of practical size. We therefore proposed a modified GA to solve large scale problems and find optimal or near optimal solutions in a reasonable time. This coped well with all instances. It required almost the same CPU time for problems of small size and for the larger instances.

In this mathematical model, the factors which are always facing the decision maker in the real world problems such as the travelling time of a quay crane between two holds on the same vessel and between two holds on different vessels have been taken into account in the optimisation process. Also, interference among quay cranes is avoided by introducing non-interference constraints which consider both the potential interference between tasks on the same vessel and those on different vessels. Another advantage in this model is to allow a quay crane that finished its work to move from one vessel to another even if the vessel handling is not finished. In addition the quay cranes do not always start their work from the same point and at the same time. For this reason, the initial position for each quay crane is taken into account in order to compute the exact time of quay crane travelling. The precedence and simultaneity constraints are taken in consideration as well.

CPLEX managed to solve most of the 10 small instances but it failed to find solutions to the large instances. GA, however, found the optimum or near optimal solution for all instances in reasonable time. Overall the average discrepancy between the objective function values of the solutions found by both algorithms is small. This shows that GA, while substantially more efficient than B&C, is also quite robust on most instances considered as this average discrepancy shows.

In Chapter 5, an integrated mathematical model to simultaneously solve the seaside operations BACASP, that arise at container terminals, is introduced. This approach is desirable because solving them individually and even when combined pairwise, may lead to suboptimal solutions. The problem is formulated as a mixed-integer programming (MIP) model. The B&C algorithm as implemented in CPLEX 12.6 has been used to find the exact solutions of relatively small instances of BACASP. It could not achieve optimality in reasonable times. It fails to solve large instances. We proposed a modified GA to solve large scale instances of the problem and find the optimal or near optimal solutions in reasonable times.

The combined model presented here is obviously the way forward as it is more likely to provide better solutions than those found by solving seaside operations problems individually. It is also a substantial improvement on combined variants which do not consider all available information.

The model itself has features which are not commonly represented in existing models. Travelling times of a quay crane between two holds on the same ship and between two holds on different vessels have also been considered. Quay crane interference avoidance is explicitly represented as constraints (5.27) when the quay cranes operate on the same vessel and as constraints (5.30) when quay cranes are on different vessels. Another advantage of this model is that it allows a quay crane that has become idle to move from its current vessel to another even if overall the current vessel to which it has been allocated is still being processed. In practice, quay cranes do not always start their works at the same time. For this reason quay cranes ready times have been considered in the mathematical model. The initial position for each quay crane is taken into account in order to compute the exact time of quay crane travelling. The precedence and simultaneity constraints are taken in consideration as well.

We have experimented on relatively limited size cases with only few vessels, tasks and quay cranes. The experimental results show that the proposed mathematical model when solved is capable of providing the optimal solution for small scale instances. However, GA coped well with all problems. It required almost the same CPU time for all problems of small size and these of larger instances. Overall the average discrepancy between the objective function values of the solutions found by both algorithms is small. This shows that GA, while vastly more efficient than B&C, it is also quite robust on most instances considered as this average discrepancy shows.

We have also applied CBC on some instances. The runtime is much longer than direct solving. The number of iterations is still too high. The master problem is solved faster than directly solving the complete problem, but the number of iterations needed increases with the problem size; default solving by CPLEX is faster.

## 6.2 Suggestions for further work

The size of the models provided in these thesis and elsewhere seems to grow exponentially in the number of constraints and decision variables with the increase in the number of vessels, tasks, and quay cranes. With this in mind, finding good solutions in acceptable times will become more and more difficult. Future work, therefore, will be concerned with the design and implementation of efficient solution approaches. Finer and seamless hybridisation of exact and approximate approach to reduce time overheads and improve solution quality is a possible area of investigation. Their evaluation through comparisons with other meta-heuristics as have been introduced recently [6, 19, 34, 79, 80, 87, 88], constitutes also a worthwhile endeavour.

Seaside and landside operations are tightly interconnected despite the fact that common practice is to deal with them individually. We have tried to alleviate this shortcoming by integrating some operations into single models and deal with them as such. Further integration to include seaside and landside operations is therefore another worthwhile area of research.

Here, it is assumed that information available to the decision maker is known deterministically. In reality, however, this assumption is unrealistic because of gaps in information and its unreliability and fluctuation. This is the curse of uncertainty. Moreover, it is inevitable since the concerned systems are complex and dynamic. This, therefore, calls upon the use of stochastic programming and new approaches such as robust programming to cope with the problem of uncertainty in the decision making process at the level of container ports.

# Bibliography

[1] A. Ak. *Berth and quay crane scheduling: problems, models and solution methods*. PhD thesis, Schoolof Industrial and Systems Engineering, Georgia Institute of Technology, 2008.

[2] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962.

[3] C. Bierwirth and F. Meisel. A fast heuristic for quay crane scheduling with interference constraints. *Journal of Scheduling*, 12(4):345–360, 2009.

[4] C. Bierwirth and F. Meisel. A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 202(3):615–627, 2010.

[5] C. Bierwirth and F. Meisel. A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 244(3):675–689, 2015.

[6] C. Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4):353–373, 2005.

[7] D. Chang, Z. Jiang, W. Yan, and J. He. Integrating berth allocation and quay crane assignments. *Transportation Research Part E: Logistics and Transportation Review*, 46(6):975–990, 2010.

[8] J. H. Chen, D.-H. Lee, and M. Goh. An effective mathematical formulation for the unidirectional cluster-based quay crane scheduling problem. *European Journal of Operational Research*, 232(1):198–208, 2014.

[9] R. Cheng and M. Gen. Parallel machine scheduling problems using memetic algorithms. *Computers & Industrial Engineering*, 33(3-4):761–764, 1997.

[10] C. Cheong, K. Tan, D. Liu, and C. Lin. Multi-objective and prioritized berth allocation in container ports. *Annals of Operations Research*, 180(1):63–103, 2010.

[11] C. Y. Cheong, M. S. Habibullah, R. S. M. Goh, and X. Fu. Multi-objective optimization of large scale berth allocation and quay crane assignment problems. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 669–676. IEEE, 2010.

[12] N. Chestney. Global seaborne trade seen doubling by 2030. http://www.reuters.com/article/marine-trade-idUSL5N0CV1PQ20130408, 2013. [Online; accessed 06-December-2016].

[13] S. Chung and F. T. Chan. A workload balancing genetic algorithm for the quay crane scheduling problem. *International Journal of Production Research*, 51(16):4820–4834, 2013.

[14] S. Chung and K. L. Choy. A modified genetic algorithm for quay crane scheduling operations. *Expert Systems with Applications*, 39(4):4213–4221, 2012.

[15] G. Codato and M. Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766, 2006.

[16] C. F. Daganzo. The crane scheduling problem. *Transportation Research Part B: Methodological*, 23(3):159–175, 1989.

[17] R. M. de Oliveira, G. R. Mauri, and L. A. N. Lorena. Clustering search heuristic for solving a continuous berth allocation problem. In *Evolutionary Computation in Combinatorial Optimization*, pages 49–62. Springer, 2012.

[18] A. Diabat and E. Theodorou. An integrated quay crane assignment and scheduling problem. *Computers & Industrial Engineering*, 73:115–123, 2014.

[19] I. Fister, X.-S. Yang, and J. Brest. A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation*, 13:34–46, 2013.

[20] Y.-M. Fu, A. Diabat, and I.-T. Tsai. A multi-vessel quay crane assignment and scheduling problem: Formulation and heuristic solution approach. *Expert Systems with Applications*, 41(15):6959–6965, 2014.

[21] S. S. Ganji, A. Babazadeh, and N. Arabshahi. Analysis of the continuous berth allocation problem in container ports using a genetic algorithm. *Journal of marine science and technology*, 15(4):408–416, 2010.

[22] M. R. Garey and D. S. Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[23] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[24] K.-S. Goh and A. Lim. Combining various algorithms to solve the ship berthing problem. In *Tools with Artificial Intelligence, 2000. ICTAI 2000. Proceedings. 12th IEEE International Conference on*, pages 370–375. IEEE, 2000.

[25] Y. Guan and R. K. Cheung. The berth allocation problem: models and solution methods. *OR Spectrum*, 26(1):75–92, 2004.

[26] P. Guo, W. Cheng, and Y. Wang. A modified generalized extremal optimization algorithm for the quay crane scheduling problem with interference constraints. *Engineering Optimization*, 46(10):1411–1429, 2014.

[27] C. J. HANG. *Models and new methods for the quayside operations in port container terminals*. PhD thesis, DEPARTMENT OF CIVIL & ENVIRONMENTAL ENGINEERING, NATIONAL UNIVERSITY OF SINGAPORE, 2011.

[28] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* (2nd ed. in 1992). Cambridge: MIT Press, 1975.

[29] J. Hooker. *Logic-based methods for optimization: combining optimization and constraint satisfaction*. John Wiley & Sons, 2000.

[30] A. Imai, E. Nishimura, and S. Papadimitriou. The dynamic berth allocation problem for a container port. *Transportation Research Part B: Methodological*, 35(4):401–417, 2001.

[31] A. Imai, X. Sun, E. Nishimura, and S. Papadimitriou. Berth allocation in a container port: using a continuous location space approach. *Transportation Research Part B: Methodological*, 39(3):199–221, 2005.

[32] D. E. Joslin and D. P. Clements. Squeaky wheel optimization. *Journal of Artificial Intelligence Research*, 10:353–373, 1999.

[33] N. Kaveshgar, N. Huynh, and S. K. Rahimian. An efficient genetic algorithm for solving the quay crane scheduling problem. *Expert Systems with Applications*, 39(18):13108–13117, 2012.

[34] J. Kennedy. Particle swarm optimization. in: Encyclopedia of machine learning. pages 760–766. Springer, 2011.

[35] A. Khachaturyan, S. Semenovskaya, and B. Vainstein. A statistical-thermodynamic approach to determination of structure amplitude phases. *Sov. Phys. Crystallogr*, 24:519–524, 1979.

[36] A. Khachaturyan, S. Semenovsovskaya, and B. Vainshtein. The thermodynamic approach to the structure analysis of crystals. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 37(5):742–754, 1981.

[37] K. H. Kim and K. C. Moon. Berth scheduling by simulated annealing. *Transportation Research Part B: Methodological*, 37(6):541–560, 2003.

[38] K. H. Kim and Y.-M. Park. A crane scheduling method for port container terminals. *European Journal of operational research*, 156(3):752–768, 2004.

[39] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[40] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[41] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960.

[42] D.-H. Lee, J. H. Chen, and J. X. Cao. The continuous berth allocation problem: A greedy randomized adaptive search solution. *Transportation Research Part E: Logistics and Transportation Review*, 46(6):1017–1029, 2010.

[43] D.-H. Lee, H. Qiu Wang, and L. Miao. Quay crane scheduling with handling priority in port container terminals. *Engineering Optimization*, 40(2):179–189, 2008b.

[44] D.-H. Lee, H. Q. Wang, and L. Miao. An approximation algorithm for quay crane scheduling with non-interference constraints in port container terminals. *Tristan VI, Phuket, June*, pages 10–15, 2007.

[45] D.-H. Lee, H. Q. Wang, and L. Miao. Quay crane scheduling with non-interference constraints in port container terminals. *Transportation Research Part E: Logistics and Transportation Review*, 44(1):124–135, 2008.

[46] Y. Lee and C.-Y. Chen. An optimization heuristic for the berth scheduling problem. *European Journal of Operational Research*, 196(2):500–508, 2009.

[47] P. Legato, D. Gullì, and R. Trunfio. The quay crane deployment problem at a maritime container terminal. In *Submitted to the 22th European Conference on Modelling and Simulation*, 2008.

[48] P. Legato and R. Trunfio. A local branching-based algorithm for the quay crane scheduling problem under unidirectional schedules. *4OR*, 12(2):123–156, 2014.

[49] C.-l. Li, X. Cai, and C.-y. Lee. Scheduling with multiple-job-on-one-processor pattern. *IIE transactions*, 30(5):433–445, 1998.

[50] C. Liang, J. Guo, and Y. Yang. Multi-objective hybrid genetic algorithm for quay crane dynamic assignment in berth allocation planning. *Journal of Intelligent Manufacturing*, 22(3):471–479, 2011.

[51] A. Lim. The berth planning problem. *Operations research letters*, 22(2):105–110, 1998.

[52] A. Lim. An effective ship berthing algorithm. In *IJCAI*, pages 594–599, 1999.

[53] A. Lim, B. Rodrigues, F. Xiao, and Y. Zhu. Crane scheduling using tabu search. In *Tools with Artificial Intelligence, 2002.(ICTAI 2002). Proceedings. 14th IEEE International Conference on*, pages 146–153. IEEE, 2002.

[54] A. Lim, B. Rodrigues, F. Xiao, and Y. Zhu. Crane scheduling with spatial constraints. *Naval Research Logistics (NRL)*, 51(3):386–406, 2004.

[55] J. Liu, Y.-w. Wan, and L. Wang. Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. *Naval Research Logistics (NRL)*, 53(1):60–74, 2006.

[56] B. Meindl and M. Templ. Analysis of commercial and free and open source solvers for linear optimization problems. *Eurostat and Statistics Netherlands within the project ESSnet on common tools and harmonised methodology for SDC in the ESS*, 2012.

[57] F. Meisel. The quay crane scheduling problem with time windows. *Naval Research Logistics (NRL)*, 58(7):619–636, 2011.

[58] F. Meisel and C. Bierwirth. Integration of berth allocation and crane assignment to improve the resource utilization at a seaport container terminal. In *Operations Research Proceedings 2005*, pages 105–110. Springer, 2006.

[59] F. Meisel and C. Bierwirth. Heuristics for the integration of crane productivity in the berth allocation problem. *Transportation Research Part E: Logistics and Transportation Review*, 45(1):196–209, 2009.

[60] F. Meisel and C. Bierwirth. A framework for integrated berth allocation and crane operations planning in seaport container terminals. *Transportation Science*, 47(2):131–147, 2013.

[61] L. Moccia, J.-F. Cordeau, M. Gaudioso, and G. Laporte. A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. *Naval Research Logistics (NRL)*, 53(1):45–59, 2006.

[62] M. F. Monaco and M. Sammarra. Quay crane scheduling with time windows, one-way and spatial constraints. *International Journal of Shipping and Transport Logistics*, 3(4):454–474, 2011.

[63] K. Moon. A mathematical model and a heuristic algorithm for berth planning. *Brain Korea*, 21, 2000.

[64] R. Moorthy and C.-P. Teo. Berth management in container terminal: the template design problem. *OR spectrum*, 28(4):495–518, 2006.

[65] G. L. Nemhauser and L. A. Wolsey. Integer programming and combinatorial optimization. *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, 20:8–12, 1988.

[66] W. Ng and K. Mak. Quay crane scheduling in container terminals. *Engineering Optimization*, 38(6):723–737, 2006.

[67] S. Nguyen, M. Zhang, M. Johnston, and K. Chen Tan. Hybrid evolutionary computation methods for quay crane scheduling problems. *Computers & Operations Research*, 40(8):2083–2093, 2013.

[68] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

[69] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982.

[70] Y.-M. Park and K. H. Kim. A scheduling method for berth and quay cranes. *OR Spectrum*, (25):1–23, 2003.

[71] M. Parker and J. Ryan. Finding the minimum weight iis cover of an infeasible system of linear inequalities. *Annals of Mathematics and Artificial Intelligence*, 17(1):107–126, 1996.

[72] R. I. Peterkofsky and C. F. Daganzo. A branch and bound solution method for the crane scheduling problem. *Transportation Research Part B: Methodological*, 24(3):159–172, 1990.

[73] M. Pinedo. Scheduling: theory, algorithms and systems, 1995.

[74] B. Raa, W. Dullaert, and R. Van Schaeren. An enriched model for the integrated berth allocation and quay crane assignment problem. *Expert Systems with Applications*, 38(11):14136–14147, 2011.

[75] M. Rodriguez-Molins, F. Barber, M. R. Sierra, J. Puente, and M. A. Salido. A genetic algorithm for berth allocation and quay crane assignment. In *Advances in Artificial Intelligence–IBERAMIA 2012*, pages 601–610. Springer, 2012.

[76] M. Rodriguez-Molins, L. Ingolotti, F. Barber, M. A. Salido, M. R. Sierra, and J. Puente. A genetic algorithm for robust berth allocation and quay crane assignment. *Progress in Artificial Intelligence*, 2(4):177–192, 2014.

[77] M. Rodriguez-Molins, M. A. Salido, and F. Barber. A grasp-based metaheuristic for the berth allocation problem and the quay crane assignment problem by managing vessel cargo holds. *Applied intelligence*, 40(2):273–290, 2014.

[78] C. E. Russell and Y. Shi. Particle swarm optimization: developments, applications and resources. In *evolutionary computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 81–86. IEEE, 2001.

[79] A. Salhi and E. S. Fraga. Nature-inspired optimisation approaches and the new plant propagation algorithm. In *The ICeMATH2011*, pages K2–1–K2–8, 2011.

[80] A. Salhi and J. A. Vazquez-Rodríguez. Tailoring hyper-heuristics to specific instances of a scheduling problem using affinity and competence functions. *Memetic Computing*, 6(2):77–84, 2014.

[81] M. Sammarra, J.-F. Cordeau, G. Laporte, and M. F. Monaco. A tabu search heuristic for the quay crane scheduling problem. *Journal of Scheduling*, 10(4-5):327–336, 2007.

[82] D. Steenken, S. Voß, and R. Stahlbock. Container terminal operation and operations research-a classification and literature review. *OR spectrum*, 26(1):3–49, 2004.

[83] R. Tavakkoli-Moghaddam, A. Makui, S. Salahi, M. Bazzazi, and F. Taheri. An efficient algorithm for solving a new mathematical model for a quay crane scheduling problem in container ports. *Computers & Industrial Engineering*, 56(1):241–248, 2009.

[84] Y. B. Türkoğulları, Z. C. Taşkın, N. Aras, and İ. K. Altınel. Optimal berth allocation, time-variant quay crane assignment and scheduling with crane setups in container terminals. *European Journal of Operational Research*, 254(3):985–1001, 2016.

[85] UNCTAD. *Review of maritime transport*. New York & Geneva: United Nations, 2015.

[86] O. Unsal and C. Oguz. Constraint programming approach to quay crane scheduling problem. *Transportation Research Part E: Logistics and Transportation Review*, 59:108–122, 2013.

[87] J. A. Vazquez-Rodríguez and A. Salhi. Hybrid evolutionary methods for the solution of complex scheduling problems. *Advances in Artificial Intelligence*, pages 17–28, 2006.

[88] J. A. Vazquez-Rodríguez and A. Salhi. A synergy exploiting evolutionary approach to complex scheduling problems. *Computer Aided Methods in Optimal Design and Operations, Series on Computers and Operations Research, World Scientific Publishing Co. Pvt. Ltd*, pages 59–68, 2006.

[89] F. Wang and A. Lim. A stochastic beam search for the berth allocation problem. *Decision Support Systems*, 42(4):2186–2196, 2007.

[90] Y. Xu, Q. Chen, and X. Quan. Robust berth scheduling with uncertain vessel delay and handling time. *Annals of Operations Research*, 192(1):123–140, 2012.

[91] C. Yang, X. Wang, and Z. Li. An optimization approach for coupling problem of berth allocation and quay crane assignment in container terminal. *Computers & Industrial Engineering*, 63(1):243–253, 2012.

[92] L. Zhen and D.-F. Chang. A bi-objective model for robust berth allocation scheduling. *Computers & Industrial Engineering*, 63(1):262–273, 2012.

[93] L. Zhen, L. H. Lee, and E. P. Chew. A decision model for berth allocation under uncertainty. *European Journal of Operational Research*, 212(1):54–68, 2011.

[94] Y. Zhu and A. Lim. Crane scheduling with non-crossing constraint. *Journal of the Operational Research Society*, 57(12):1464–1471, 2006.

# Publications

1- Alsoufi, G., Yang, X., Salhi, A. A combinatorial Benders' cuts approach to the seaside operations problem in container ports. Presented in the 11th metahueristics international conference, Agadir, Morocco, June 7-10, 2015.

2- Alsoufi, G., Yang, X. and Salhi, A. Robust Berth Allocation Using a Hybrid Approach Combining Branch-and-Cut and the Genetic Algorithm. In International Workshop on Hybrid Metaheuristics, pp187-201, 2016. Springer International Publishing.

3- Alsoufi, G., Yang, X. and Salhi, A. Combined Quay Crane Assignment and Quay Crane Scheduling with Crane Inter-Vessel Movement and Non-Interference Constraints. Journal of the Operational Research Society, 1-12, 2017.

4- Alsoufi, G., Yang, X. and Salhi, A. An Evolutionary Approach to a Combined Mixed Integer Programming Model of Seaside Operations as Arise in Container Ports. Accepted in Journal of Annals of Operations Research, 2017.