VNIVERSITAT DE VALÈNCIA

Facultat de Ciències Matemàtiques

Departament d'Estadística i Investigació Operativa

Programa de doctorat en Estadística i Optimització

DOCTORAL THESIS

Models and algorithms for berth allocation problems in port terminals

Juan Francisco Correcher Valls

Supervised by

Dr. Ramón Álvarez-Valdés Olaguíbel

April 2017

Abstract

Seaports play a key role in maritime commerce and the global market economy. Goods of different kinds are carried in specialized vessels whose handling requires ad hoc port facilities. Port terminals comprise the quays, infrastructures, and services dedicated to handling the inbound and outbound cargo carried on vessels. Increasing seaborne trade and ever-greater competition between port terminals to attract more traffic have prompted new studies aimed at improving their quality of service while reducing costs. Most terminals implement operational planning to achieve more efficient usage of resources, and this poses new combinatorial optimization problems which have attracted increasing attention from the Operations Research community. One of the most important problems confronted at the quayside is the efficient allocation of quay space to the vessels calling at the terminal over time, also known as the *Berth Allocation Problem*. A closely related problem arising in terminals that specialize in container handling concerns the efficient assignment of quay cranes to vessels, which, together with quay space planning, leads to the *Berth Allocation and Quay Crane Assignment Problem*. These problems are known to be especially hard to solve, and therefore require designing methods capable of attaining good solutions in reasonable computation times.

This thesis studies different variants of these problems considering well-known and new real-world aspects, such as terminals with multiple quays or irregular layouts. Mathematical programming and metaheuristics techniques are extensively used to devise tailored solution methods. In particular, new integer linear models and heuristic algorithms are developed to deal with problem instances of a broad range of sizes representing real situations. These methods are evaluated and compared with other state-of-the-art proposals through various computational experiments on different benchmark sets of instances. The results obtained show that the integer models proposed lead to optimal solutions on small instances in short computation times, while the heuristic algorithms obtain good solutions to both small and large instances. Therefore, this study proves to be an effective contribution to the efforts aimed at improving port efficiency and provides useful insights to better tackle similar combinatorial optimization problems.

to my parents

Acknowledgements

This thesis is the result of my research carried out in the *Departament d'Estadística i Investigació Operativa* at the *Universitat de València* (Spain), from the end of 2013 to the end of 2016. Many people have contributed directly or indirectly to the fulfilment of this work.

First of all, the help and mentorship of Ramón Álvarez-Valdés, my supervisor, have been invaluable, and I cannot thank him enough. He has been always open to discussing any detail and has generously promoted many activities to deepen my research and my knowledge of the field.

I would also like to express my gratitude to the members of the department, especially to José Manuel Tamarit, for his collaboration and his sagacious advice on certain optimization tools; José Bermúdez, for his statistical suggestions; Maite León, for her guidance in my first teaching experiences at the university; and Ángel Corberán, for kindly welcoming me into the department.

Nor can I forget my office mates: Blanca, Abel, and Hèctor, who have accompanied me during these years, and the other *predocs* in the faculty, especially Juanjo, Sheldon, Enric and Adina, with whom I shared good times, much needed to endure the difficulties of the daily work at the research coalface.

I am also grateful for the kindness and hospitality of Greet Vanden Berghe and all the members of the Combinatorial Optimisation and Decision Support group at KU Leuven in Ghent (Belgium). They treated me as if I were another member of the group.

I would like to thank the Spanish society, for financing this investigation through several grants and projects of the Conselleria d'Educació, Formació i Ocupació of the Generalitat Valenciana and the Spanish Ministerio de Economía y Competitividad.

I also thank my friends, who have always encouraged me and have always been ready to make the most of my visits to Alicante.

Finally, I have no words to express my gratitude to my family, especially to my parents, who have given me everything. Without their care and support this research could have not been accomplished.

Contents

List of Figures

List of Tables

List of Pseudocodes

List of Models

List of Acronyms

ACA	Analytic Constructive Algorithm
ANOVA	Analysis of variance
BACAP	Berth Allocation and Quay Crane Assignment Problem
BACASP	Berth Allocation and Specific Quay Crane Assignment Problem
BAP	Berth Allocation Problem
BRKGA	Biased Random-Key Genetic Algorithm
CPA	Cutting Plane Algorithm
DWT	Deadweight tonnes
ECA	Exploratory Constructive Algorithm
FCFS	First-Come, First-Served
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
LS	Local Search
MILP	Mixed Integer Linear Program
OR	Operations Research
QC	Quay Crane
QCAP	Quay Crane Assignment Problem
QCSP	Quay Crane Scheduling Problem
SBS	Stochastic Beam Search
TEU	Twenty-foot Equivalent Unit
TS	Tabu Search
UNCTAD	United Nations Conference on Trade and Development

Chapter 1

Introduction

Homo sapiens has always striven to make good decisions. Throughout history, humans have proved capable of planning future actions and performing them systematically to achieve various goals. The first manifestations of this executive planning are lost in the mists of time, but traces from the past reveal our age-old interest in the development of our practical capabilities. Since the Paleolithic era, we have conceived myriads of tools and methods to change the environment with the aim of improving our living conditions. And this has not been a merely individual effort, but a social endeavour.

Thousands of years ago, the domestication of some plants and animals made it possible for certain human groups to settle permanently, and as a result their population and density increased. The ensuing new conditions and needs demanded new efforts to deal with them. The construction of infrastructures and the management of crops, warehouses and other resources posed problems such as the recording of information and the correct combination of measures (see Harris and Johnson, 2007, Graeber, 2012). Furthermore, agriculture required precise predictions to determine calendars based on the movement of celestial bodies. As a response to these problems, in some of those settlements writing and arithmetic were conceived and developed. These instruments brought about new forms of symbolic representation and abstraction, which in turn enhanced our ability to organize people and changing our surroundings. They also contributed to improving our rudimentary astronomy and geometry and prompted the development of new tools and techniques (see Solís and Sellés, 2013).

Over centuries, various sciences were developed to better explain and predict the world, and many of them found valuable support in mathematics for abstracting, modelling and communicating different aspects of reality. They served all kinds of human purposes, ranging from saving people from a premature death by constructing sewage systems to killing people in wars by using siege machines. Meanwhile, theories and methods in astronomy, physics and many technical disciplines were perfected to become more precise and effective in their application to reality (see Solís and Sellés, 2013, Kuhn, 1992).

From the sixteenth century of the Common Era the expansion of states around the world and emerging capitalism demanded new organizing methods based on the application of calculation to aggregated data. Thus was born political arithmetic, the precur-

sor of statistics and political economy, as a support for managerial decision-making (see Naredo, 2015). Additionally, new mechanical inventions were conceived and progressively introduced into workshops to increase productivity. During the nineteenth century, the boom of industrial production and the expansion of markets forced capitalist companies to become more competitive on pain of bankruptcy. Engineers were in demand to achieve the best utilization of factory resources by devising new machines, thereby automating processes that until then had been carried out by human beings (see Polanyi, 2001). Fossil energy sources, such as coal and oil, were increasingly exploited to serve industrialization and also fostered an unprecedented rise in transportation. Steamships freed navigation from the vagaries of wind and improved upon the capacity of previous vessels, while railways connected vast areas in the hinterlands crossing the continents (see Stopford, 2009, ch. 1). The new alternatives gave rise to new opportunities for companies, and consequently posed new managerial problems (see Meersman and de Voorde, 2010). States, for their part, continued competing with each other for land, natural resources, and populations, and therefore they also had to confront new challenges in organizing large masses of people, especially growing armies composed of many specialized units with diverse demands and characteristics.

During the first half of the twentieth century, business and military managers prompted the emergence of sciences such as logistics and operations research. Mathematicians, engineers, and other scientists devised formal methods to plan and optimize processes such as the exploitation, organization and transportation of resources. Simultaneously, computer science, a new discipline based on mathematical logic, laid the foundations of automated computing and problem solving and made it possible to construct programmable computers. Such sciences underwent considerable development during the second half of the century, fostered by states, mass consumption and growing international trade (see Hillier and Lieberman, 2015, ch. 1). Commerce between regions in different continents experienced an upsurge with the development of diesel-driven cargo vessels, which contributed to maritime commerce by improving freight transport efficiency. Moreover, the standardization of containers for the transportation of a range of cargo brought about the construction of specialized vessels and the development of ad hoc facilities in ports worldwide. Consequently, these infrastructures posed new operational planning problems, which began to be analysed using the new management sciences (see Heaver, 2012).

Maritime commerce continues to grow in the twenty-first century and the operations research community is increasingly interested in the mathematical analysis and formulation of the problems encountered therein (Woo et al., 2011). Many of these problems concern the optimization of planning decisions in ports so that scarce resources are used efficiently. Thus researchers tackle these problems by developing models and computational methods capable of obtaining good or even optimal solutions. Pursuing this line, the present study is a modest contribution to the efforts aimed at achieving the most efficient use of the resources available, in particular, at the quayside of seaports.

1.1 Scope and objectives

The purpose of this doctoral thesis is to analyse various optimization problems concerning operational berth planning in port terminals and to propose new methods for solving them. The main resource managed in this process is the quay space, which must be used efficiently by the terminal in order to provide a high quality service in processing calling vessels. In container terminals, quay cranes also have to be assigned efficiently to moored vessels, as they determine vessels' handling time and thus affect the vessel schedule. Both resources, quay space and quay cranes, give rise to different combinatorial optimization problems.

In particular, this thesis will address various versions of the *Berth Allocation Problem* and the *Berth Allocation and Quay Crane Assignment Problem*. These problems concern the assignment of berthing time and position (and quay cranes) to each vessel expected to arrive within a given time horizon. The objective is to minimize the stay of vessels at the terminal, since this is a determining factor in its competitiveness. In formulating these problems, new and existing constraints arising in real-world cases, which give rise to interesting new variants, will be taken into consideration.

The main objective is to provide methods capable of solving realistic instances of these problems. Specifically, new mathematical programming models and exact methods will be proposed to obtain optimal solutions, whenever possible, and new heuristic algorithms will be developed to obtain high quality solutions in short computation times. These methods will be evaluated through extensive computational experiments and compared with existing proposals in the literature.

The approach is aimed at helping decision makers in port terminals to better perform quayside operational planning, thereby contributing to the state of the art of operations research at ports. Furthermore, regarding the mathematical dimension, it will provide new insights and search strategies which will be useful for better confronting similar combinatorial optimization problems.

1.2 Outline

This thesis is divided into nine chapters:

This chapter introduces the research, defines its scope and objectives and describes the structure of the thesis.

Chapter 2 examines the context and motivation in depth. It first outlines the fundamentals of maritime freight transport and the increasing demand for higher port efficiency. Next it describes the main areas and operational planning problems encountered at container terminals. It then focuses on the problems tackled in this investigation and their variants. Finally, it outlines the research methods employed in the field to address these problems, especially those applied in this thesis.

Chapter 3 describes the investigation conducted on a novel continuous Berth Allocation Problem in which several quays can be taken into account. First, a literature review is provided to clarify the precedents. Then the problem is described and formulated by means of a new mixed integer linear model. A Genetic Algorithm is also

developed to obtain good solutions to the problem. Finally, these methods are evaluated and compared with other proposals in the literature through extensive computational experiments.

Chapter 4 addresses the integration of the continuous Berth Allocation Problem and the Quay Crane Assignment Problem. After a literature review, this combined problem is formulated by means of a new mixed integer linear model. Then several families of valid inequalities are proposed to reinforce the formulation. The performance of this approach is assessed and compared with other state-of-the-art approaches in various computational experiments.

Chapter 5 tackles an extended version of the problem studied in the previous chapter. In this variant, instead of a number of cranes, a set of specific cranes is assigned to each vessel, taking into account additional real-world constraints. To solve this problem, a mixed integer linear model and a cutting plane algorithm are proposed and compared with other existing exact approaches through several experiments.

Chapter 6 proposes a Biased Random-Key Genetic Algorithm with Memetic characteristics and several Local Search procedures to solve large instances of the problem tackled in Chapter 4. The parameters of these algorithms are adjusted by means of extensive experiments and statistical analysis. Finally, the selected configuration of the Genetic Algorithm is compared with the exact approaches to evaluate its performance.

Chapter 7 extends these heuristic methods to solve the problem addressed in Chapter 5. Analogous experiments and comparisons are conducted to assess their quality on small and large instances.

Chapter 8 tackles a novel hybrid Berth Allocation Problem arising in terminals with irregular quays. In this problem, new restrictions on vessel mooring and departure imposed by the relations between the berths (such as adjacency, oppositional and blocking relations) are taken into consideration. After the literature review, a mixed integer linear model, easily adaptable to address a wide range of problems, is proposed. Several experiments are conducted on real-world and randomly generated instances to evaluate this approach.

Finally, Chapter 9 summarizes the contributions of the investigation presented in this thesis and proposes further research directions. Additional information about the projects, the derived works and other issues related to the communication of these results is also provided.

Chapter 2

Preliminaries

This chapter examines the context and motivation of this research in depth. It first outlines the fundamentals of maritime freight transport and its current trends. Next it focuses on the increasing demand of higher port efficiency and productivity, which is motivated by the rise in seaborne trade. Operations Research is then presented as a scientific field dedicated to the study and improvement of efficiency in many processes, including port management. Next, the main areas and operational planning problems of a container terminal are described. An overview of the problems addressed in this investigation and their variants is then provided. Finally, this chapter outlines the research methods and techniques employed in the field to tackle these problems, especially those applied in this study.

2.1 Maritime Transport

Maritime transport plays a crucial role in the global economy. It is the backbone of international supply chains and trade networks. Every day, thousands of vessels cross the seas bound for ports around the world. This is possible only by efficient coordination of multiple agents: shippers, port operators, freight forwarders, and carriers, who are responsible for different services. Briefly, the *shipper* is the agent who wants to send one or more goods and so bears the freight cost, the *port operator* handles those goods in the port, the *freight forwarder* organizes the shipments for one or more shippers by contacting one or more carriers, and the *carrier* transports the goods on the ships it operates (see Stopford, 2009, ch. 3). The connection between the seaport and the *hinterland*, that is, the inland region which is the origin and destination of the goods, is effected by other carriers via waterways, using small ships such as barges and feeders, and by *land-based carriers*, who carry the goods by rail or road to the recipient (Figure 2.1).

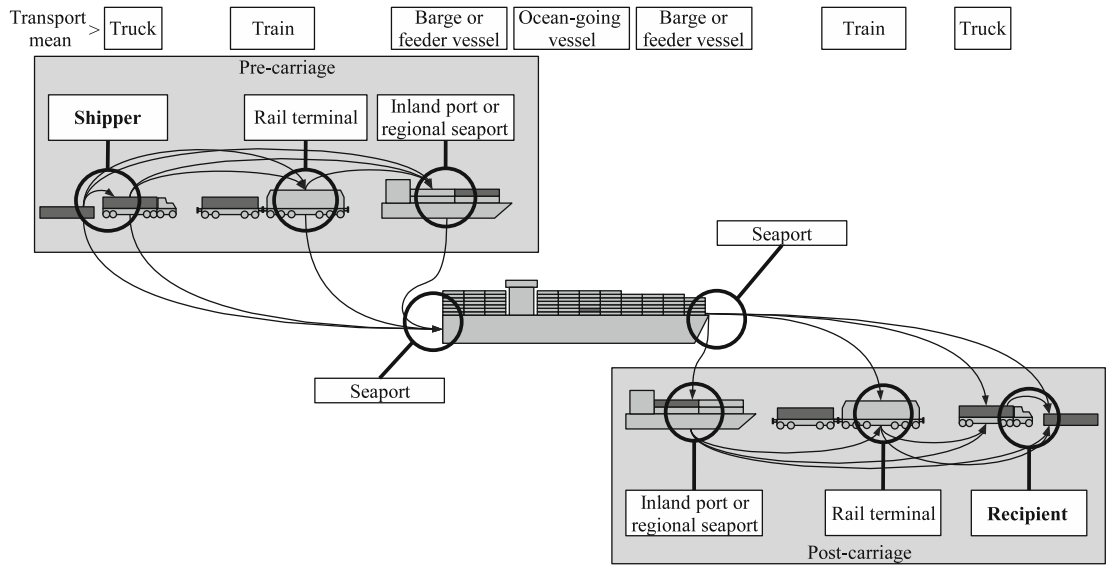


Figure 2.1: Transport chain (Meisel, 2009, p. 8).

Maritime transport is the main mode of international freight transport. Estimations reported by the United Nations Conference on Trade and Development (UNCTAD, 2016, p. 6) indicate that in 2015 seaborne trade accounted for 80% of total world merchandise trade in terms of weight, clearly surpassing overland and air transport. In terms of value, its share is estimated as lying between 55% and 66%. Indeed, maritime transport is very efficient in energy consumption per tonne-km and is able to greatly benefit from economies of scale, thereby reducing unit costs and producing lower CO_2 emissions compared to other modes of transport (UNCTAD, 2012, p. 129). The book by Stopford (2009) and the books published by Grammenos (2010) and Talley (2012) provide comprehensive introductions to maritime economics.

2.1.1 Ships

Goods are transported in different specialized vessels according to their type. The UNCTAD distinguishes up to four main categories of commercial vessels: oil tankers, bulk carriers, container ships and general cargo ships. Among the vessels not included in these categories there are chemical tankers, gas tankers, offshore ships, ferry and passenger ships, and other unclassified ships. Oil tankers and bulk carriers usually transport just one kind or very few kinds of cargo in a single shipment, while container ships are able to carry a large variety of goods packed into containers, which are piled up inside the vessel.

The transport of goods using standardized containers emerged in the 1960s as a means of improving efficiency in worldwide logistics. Increasing global trade, especially between North America, Japan, and Europe, required fast, reliable and secure freight transport, which at that time was still heterogeneous and labour-intensive (see Stopford,

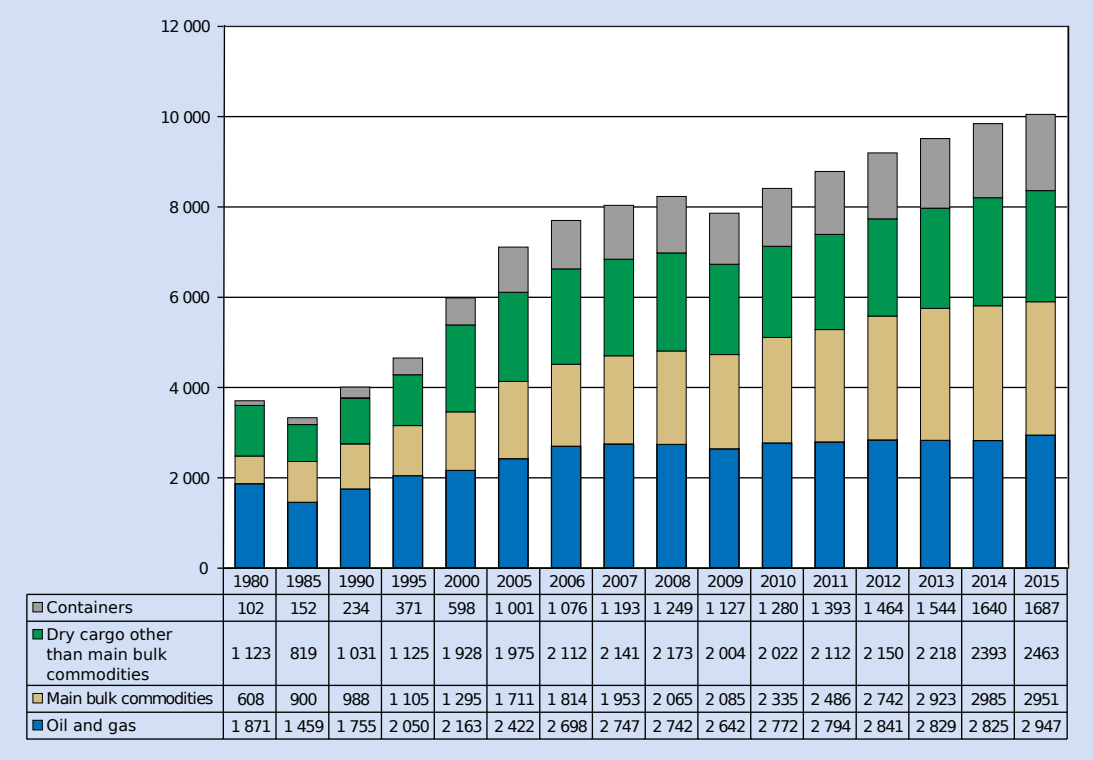
2009, ch. 13; and Notteboom, 2012). Between 1968 and 1970, several ISO recommendations, especially ISO 668, established a standard for freight containers, thus defining sizes and maximum weights, identification markings and other details. As a result, the 20-foot (6.06 m long, 2.44 m wide and 2.59 m high) and 40-foot standard containers were increasingly used in maritime, road and rail transport (see Levinson, 2008). The dimensions of a 20-foot container, known as a Twenty-foot Equivalent Unit (TEU), are used as a common unit of container traffic and capacity, though nowadays 40-foot containers (2 TEU) are the most common. Container ships are usually operated by shipping lines, which are dedicated to providing a fixed service at regular intervals between selected ports, thereby offering transport of any goods ready to be loaded in a terminal on a scheduled basis (see Stopford, 2009, ch. 3).

According to the UNCTAD (2016, ch. 2), in 2016 the world commercial fleet consisted of 90 917 vessels overall, amounting to a combined total of 1.8 billion DWT.[1] Most of them are oil tankers and dry bulk carriers, which transport grain and raw materials used in a wide range of services and production processes. Goods of very different kinds, but especially manufactured products, are mostly carried in container ships. Since the emergence of the unitization of cargo by means of standardized containers, the construction of container ships has increased substantially, replacing most of the general cargo vessels. Over the last decade, the accumulated capacity of existing container ships increased by more than 90%, reaching 244 million DWT in 2016. Moreover, their average capacity has risen by more than 20% in the last five years (UNCTAD, 2017). Indeed, the capacity of the container ships built within the last four years was 79 877 DWT on average, 2.8 times the capacity of those built 15–19 years ago. In the case of new bulk carriers and new oil tankers it was slightly lower, at 78 988 DWT and 77 324 DWT respectively. By the end of 2015, the average capacity of the container ships operated by the 50 shipping companies with the largest overall fleet capacity was 4390 TEU. The largest container ships built to date, such as MSC Oscar, are able to carry more than 19 200 TEU, with a tonnage exceeding 195 000 DWT.

2.1.2 Seaborne freight traffic

In 2015, world seaborne trade surpassed 10 billion tonnes, hitting a historical record (Figure 2.2). Since 1970, maritime trade has increased by 285%, while over the last decade it expanded by 30%. Most of this trade was in dry cargo, consisting of bulk goods and containers (UNCTAD, 2016, ch. 1). Over the same decade, shipments of the main bulk commodities (iron ore, coal, grain, bauxite and alumina, and phosphate rock) grew by 63%, and container shipments increased by 57%, reaching 175 million TEU. In terms of value, containerized cargo accounts for more than half the value of all international seaborne trade. Unlike dry cargo, the absolute amount of seaborne oil and gas trade underwent only a slight increase in the same period.

[1] DWT: deadweight tonnes. It is a measure of the weight-carrying capacity of a vessel in tonnes, not including the weight of the ship itself. It includes cargo, fuel and stores.



	1980	1985	1990	1995	2000	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Containers	102	152	234	371	598	1 001	1 076	1 193	1 249	1 127	1 280	1 393	1 464	1 544	1640	1687
Dry cargo other than main bulk commodities	1 123	819	1 031	1 125	1 928	1 975	2 112	2 141	2 173	2 004	2 022	2 112	2 150	2 218	2393	2463
Main bulk commodities	608	900	988	1 105	1 295	1 711	1 814	1 953	2 065	2 085	2 335	2 486	2 742	2 923	2985	2951
Oil and gas	1 871	1 459	1 755	2 050	2 163	2 422	2 698	2 747	2 742	2 642	2 772	2 794	2 841	2 829	2 825	2 947

Figure 2.2: International seaborne trade in millions of tonnes loaded (UNCTAD, 2016).

The transport routes of both bulk cargo and fuels differ considerably depending on the specific type of goods carried, given that the different commodities are extracted or produced at different locations around the world (see Meersman and de Voorde, 2010; Stopford, 2009, ch. 9). However, most containerized traffic between continents follows a main east-west sea lane which connects the largest productive regions and markets (Figure 2.3). Along this lane, containers are transported from East Asia to Europe through the Suez Canal, and to North America crossing the Pacific Ocean. More than 20 million TEU are carried over a year on each of those routes, jointly accounting for more than 20% of global containerized trade (UNCTAD, 2016, p. 18). Another important connection between continents is the Transatlantic route, which connects Europe and North America. In this route, container traffic exceeds 6 million TEU. Other sea lanes extend north-south, notably those connecting Europe and North America with South America. There are also many secondary routes that connect various southern regions in different continents and other routes that connect regions within the same continents. Intraregional and south-south container traffic accounts for 40% of total containerized trade, reaching 70 million TEU in 2015.

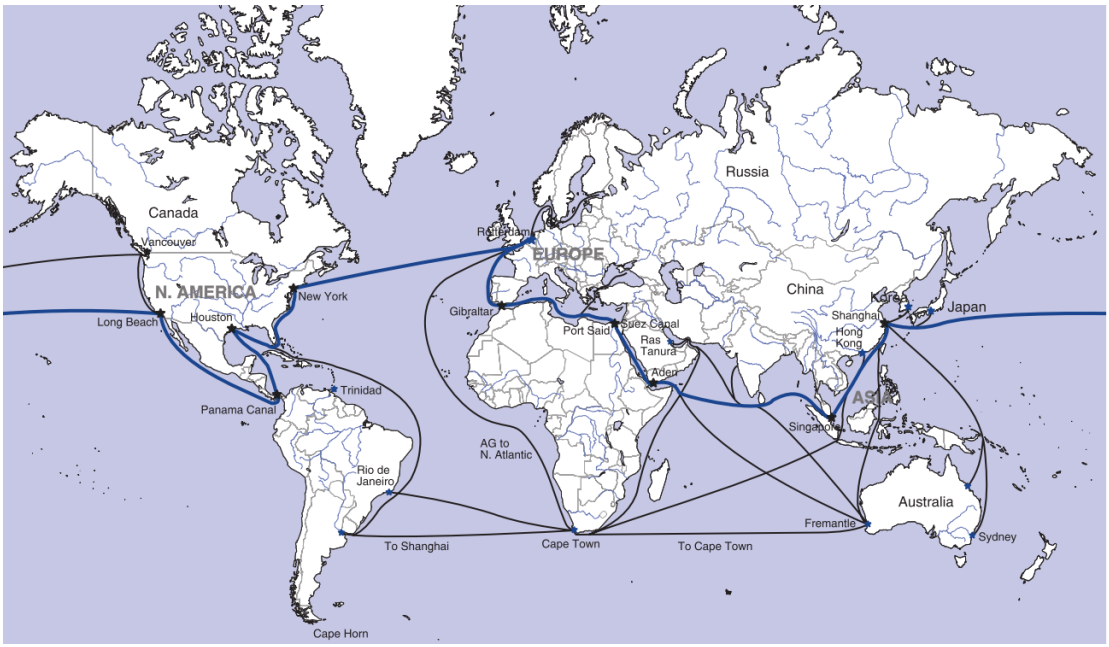


Figure 2.3: Major shipping routes (adapted from Stopford, 2009, p. 348).

2.1.3 Ports

Seaports are an essential factor in maritime transport. Ports may be public entities, government organizations or private companies. The port authority is the organization that controls one or more ports and is responsible for providing a range of services to the agents operating therein. A port may consist of one or more terminals, each consisting of one or more berths dedicated to a specific type of cargo handling. Thus there are container terminals, tank terminals, bulk terminals, and even terminals specializing in a single commodity, such as coal. Terminals may be operated by the port authority or by companies, which may own them or obtain a concession agreement for their exclusive use (Stopford, 2009, p. 81). Tank and bulk terminals are usually operated by a single company which also owns the cargo, whereas container terminals are multiple-user terminals, since they handle cargo for a number of owners. Although container terminals generally admit vessels regardless of the shipping company to which they belong, several shipping lines have dedicated terminals at which only they can call.

The leading ports by total weight handled in a year are located in East Asia, mainly in China. The port of Ningbo-Zhoushan headed the list for more than three consecutive years, with an average of over 800 millions tonnes handled each year. It is followed by the ports of Shanghai, Singapore, Tianjin, and Suzhou, each with an annual volume exceeding 450 million tonnes each one. The ports of Rotterdam, in Europe, and Hedland, in Australia, with similar volumes, stand out over the rest in their respective continents. In North America, the port of South Louisiana handles 250 million tonnes each year on average. A similar pattern can be seen in the annual throughput of container terminals, with East Asia, especially China, leading the ranking. Both Shanghai and Singapore

manage over 30 million TEU per year on average. Between East Asia and Europe, the ports of Dubai stand out, handling annually more than 14 million TEU on average. In Europe, the port of Rotterdam, with over 11 million TEU per year, is followed by the ports of Antwerp and Hamburg, each exceeding 8 million TEU annually. In North America, the ports of Los Angeles and Long Beach reach 7 million TEU each year.

According to the most recent public data on port performance reported by the UNCTAD (2015, p. 83), in 2014 the 10 most efficient container terminals in the world in terms of berth productivity ranged from the 117 crane moves per ship per hour on average of the Nansha Phase I terminal (China) to the 180 of the APM's terminal at the port of Yokohama (Japan). If we compare whole ports instead of individual terminals, then the values range from 90, corresponding to the port of Xiamen (China), to 138 moves per ship per hour on average, achieved by the port of Jebel Ali (Unit Arab Emirates). The competition between ports, and even between terminals within the same port, puts strong pressure on their processes. The operations have to be efficiently performed to reduce costs and provide a high-quality service to clients. Hence, port productivity and efficiency have become important matters of research. The next section examines these issues in more detail.

2.2 Port productivity and efficiency

The efficient management of port infrastructures and services has been one of the most important challenges throughout the history of shipping (see Heaver, 2012). Terminal operators consider productivity improvements to be indispensable to deal with the enormous scale of maritime trade and the fierce competition both within ports and between ports located in the same region, especially in the case of container traffic. Moreover, it plays a major part in reducing sea transport costs (see Stopford, 2009, ch. 2.8) and can be an important strategic factor for a region to attract new industries and improve its economic position (Salim, 2015, Meersman and de Voorde, 2010). This interest is reflected in the increasing number of academic studies on port competitiveness and performance since the late 1990s. According to the estimates reported by Woo et al. (2012), during the first decade of this century the proportion of scientific papers related to these topics increased from 11% to 24% of all port research papers. Practitioners, as well as the UNCTAD, also emphasize the importance of port performance in their publications and even publish port productivity rankings (Journal of Commerce, 2014).

Besides governments and terminal operators, the agents most interested in port performance are the shipping lines. Several empirical studies show that operational efficiency is one of the most decisive factors in port competitiveness and plays a major role in port choice by shipping lines (Tongzon and Heng, 2005, Tang et al., 2011). Given that these companies provide regular services based on a tight schedule, the turnaround time of vessels at the ports in the itinerary becomes critical, especially in a context of increasing integration of container supply chains (Sciomachen et al., 2009). A delay in departure from one port turns into a cascade of delayed arrivals in the next ports or into additional expenses resulting from the increase in fuel consumption necessary for the vessel to arrive on time. Empirical data analysed by Notteboom (2006) showed that

65.5% of schedule disturbances were caused by unexpected waiting times of vessels before berthing, while 20.6% resulted from unexpected low productivity at the terminals. Although many shipping lines' managers recognize that their excessively tight schedules are partly to blame, it seems clear that the time spent by vessels at ports is a major source of schedule unreliability (Chung and Chiang, 2011). Additionally, terminal operators experience higher pressure when carriers decide to use *low steaming*, the reduction of fuel consumption aimed at reducing costs. This policy is applied especially when oil prices rise, and it results in a reduction of sailing speed. As a consequence, if delivery due times are not changed accordingly, terminal operations are required to handle the vessels faster (Sciomachen et al., 2009).

A closely related factor that puts pressure on port productivity is the growing congestion of vessels experienced at ports, which also reveals a shortage of capacity. This is mainly due to the deployment of very large vessels and the continuous increase in the number of containers worldwide, which in turn is a consequence of increasing world trade and, especially, Asian exports (Islam and Olsen, 2013). With the aim of meeting the agreed schedules and preventing their clients from choosing their competitors, terminals attempt to process more vessels, whose size is greater in both length and beam, rapidly and simultaneously. According to van Marle (2015), the increases in beam and thus in the number of containers transported per vessel cannot be appropriately addressed by adding more quay cranes, as in the case of the increases in length. Consequently, the efficient performance of the operations becomes decisive.

In order to better know the actual performance of a port, it is important to distinguish productivity from efficiency, since their corresponding measures provide different information. Although the terms *productivity* and *efficiency* are frequently used interchangeably, the former refers to a broader concept. According to Suárez-Alemán et al. (2016), productivity is the ratio of output to input, whereas efficiency is a relative measure referring to either the maximum output that can be achieved under a given amount of input or the minimum input required to achieve a given amount of output. Thus, efficiency measures how close a system is to its optimal productivity. Port productivity is usually measured using the operational indicators proposed in UNCTAD (1976), such as: tonnes per ship-hour in port, turnaround time, and, especially in the case of container terminals, number of container moves per ship-hour at berth (Journal of Commerce, 2014, Esmer, 2008). According to van Marle (2015), this last measure is commonly used by shipping lines to compare productivity between container terminals. However, given that terminals are endowed with different inputs (berths, cranes, facilities, labour force, etc.), simple input-output may not properly inform us about how efficient they are. Therefore, in order to estimate terminal efficiency, researchers apply different methods based on production or cost frontiers, such as Data Envelopment Analysis (DEA) and Stocastic Frontier Models (SFM). Two important reviews of their application to seaports were presented by González and Trujillo (2008) and Panayides et al. (2009), while Meersman and de Voorde (2010) provides a general introduction to port efficiency.

Suárez-Alemán et al. (2016) state that terminals can improve their productivity mainly by changes in technology or by raising their efficiency. The efficiency can be increased through adjustments aimed at optimizing the size of their operations (scale

efficiency) or by a better use of the inputs (pure technical efficiency). In this study, the authors evaluated the efficiency of 203 ports of 70 developing countries (including China) from 2000 to 2010 and concluded that technology did not drive the observed variance in productivity growth. Therefore, efficiency emerges as a clearly determinant factor of increases in productivity. Indeed, technology changes frequently entail costly and long-term investments, and terminals face a pressing demand for greater productivity while attempting to minimize costs (Islam and Olsen, 2013). For this reason, the development of new methods aimed at improving efficiency at ports is gaining increasing attention within the Operations Research community. The next section outlines the common operational problems faced in port terminals and the role played by Operations Research in their analysis and solution.

2.3 Operations Research at ports

Operations Research (OR) is a scientific field that developed in the context of military forces early in World War II in an attempt to devise formal methods to better analyse and solve the management problems posed by complex organizations. This approach rapidly intertwined with the methods and interests of private businesses and thus acquired a broader scope, also known as *Management Science* (Hillier and Lieberman, 2015). According to the *Encyclopedia of Operations Research and Management Science* (Gass and Fu, 2013, Preface),

> Operations Research can be defined as: (1) the application of the methods of science to complex problems arising in the direction and management of large systems of men, machine, materials, and money in industry, business, government, and defense; (2) the science of deciding how to best design and operate man-machine systems; (3) a scientific method for providing executive departments with a quantitative basis for decision making. Management Science can be defined as: (1) the application of scientific methodology or principles to management decisions; (2) the use of quantitative methods for solving management and organizational decision problems. Together, OR and MS may be thought of as the science of operational processes, decision making, and management.

Since its beginnings, one of the main interests in the field has been the achievement of optimal decisions in order to either minimize the amount of resources utilized or maximize the throughput. Hence, researchers have focused on the analysis, modelling, and solution of optimization problems. During the 1940s and 1950s, Operations Research made a great progress with the development of mathematical theories such as linear programming, dynamic programming, inventory theory, and queueing theory. The proposal of the *Simplex* method made it possible to solve *linear programs* efficiently, that is, optimization problems formulated using real variables, constraints on the variables modelled as linear inequalities, and a linear function that is to be minimized or maximized.

By the end of the 1950s, the proposal of methods such as *Branch & Bound* also made it possible, in a general fashion, to address *combinatorial optimization problems* mod-

elled as linear programs with integer variables, known as *integer linear programs*. Such problems arise in many real situations that require obtaining an optimal arrangement of a finite set of objects, and some of them are characterized by their high *computational complexity*, for we do not know any fast procedure able to solve them optimally and its existence is considered unlikely (see Korte and Vygen, 2012).

In the following decades, the increasing advances in computing systems facilitated the application of new analytical and computational methods such as search algorithms and simulation techniques. The use of all these developments spread through many fields, both in science and industry, thereby contributing to the analysis and solution of their particular problems. Since the 1980s, the upsurge in personal computers and the development of *metaheuristic methods* that yield good suboptimal solutions in short computing times has given a new impetus to OR. Today it deals with problems in transportation, construction, manufacturing, health care, and public services, among many others (see Hillier and Lieberman, 2015, ch. 1; Gass and Assad, 2005).

In seaport research, the number of OR studies published in scientific journals has been increasing during the last two decades. Furthermore, according to the estimates provided by Woo et al. (2011), their share of total port research studies doubled during the same period, rising from 10% in the late 1990s to 20% in 2009. This reveals a growing interest in the field and an increasing use of the methods provided by OR, probably related to the growing interest in port competitiveness and efficiency. With respect to terminal operations, the same estimates also show a rise in both the number of studies and their share of total port research papers. This increase is confirmed by Islam and Olsen (2013), who focus on the papers related to container terminals. In this study, the authors also reported that from 2006 to 2010 more than 85% of the journal publications adopted analytical approaches that exclusively apply optimization algorithms. More recently, reviews on the planning problems faced in each area of container terminals, such as those of Bierwirth and Meisel (2015) and Carlo et al. (2014a), corroborate this trend.

The planning problems at port terminals tackled by OR can be classified by the level of abstraction and temporal scope as strategic, tactical, or operational. The *strategic level* concerns planning decisions whose execution or impact extends for several years, such as the design and location of new infrastructure or the acquisition of costly equipment. The *tactical level* involves planning decisions related to the usage of space in the terminal and policies whose scope may extend for weeks or months. For example, tactical decisions may involve the positioning of container blocks and the rearrangement of resources in different areas in order to adapt to new traffic forecasts. Finally, the *operational level* concerns the planning of the operations performed to fulfil the fundamental services of the terminal. These operations last from seconds up to several days and range from scheduling internal transport and work shifts to assigning resources to specific tasks (Meisel, 2009, ch. 3.1; Meersmans and Dekker, 2001).

This thesis focuses on certain operational planning problems encountered particularly in container terminals. The next section describes the typical structure of this kind of terminal and outlines the most important operational processes performed in them that are tackled by Operations Research.

2.4 Container terminals: structure and operational planning

A container terminal is a port area specializing in the management of container ships and their related processes. The general planning problems are common to all container terminals, although the specific functioning of the terminal depends mainly upon the available equipment. Detailed descriptions of container terminals and their different modes of operation are provided by Steenken et al. (2004) and Brinkmann (2011). The academic works published on operational planning problems in container terminals have been comprehensively reviewed by Steenken et al. (2004), Stahlbock and Voß (2008), Goodchild et al. (2011), Rashidi and Tsang (2013), Gharehgozli et al. (2016), Li et al. (2015), and Kim and Lee (2015).

In a typical container terminal we can identify three main areas (Figures 2.4 and 2.5):

- The *Seaside*, where vessels are moored and containers are loaded and unloaded.
- The *Yard*, where containers are stored temporarily.
- The *Landside*, where trucks and trains carry containers to and from the hinterland.

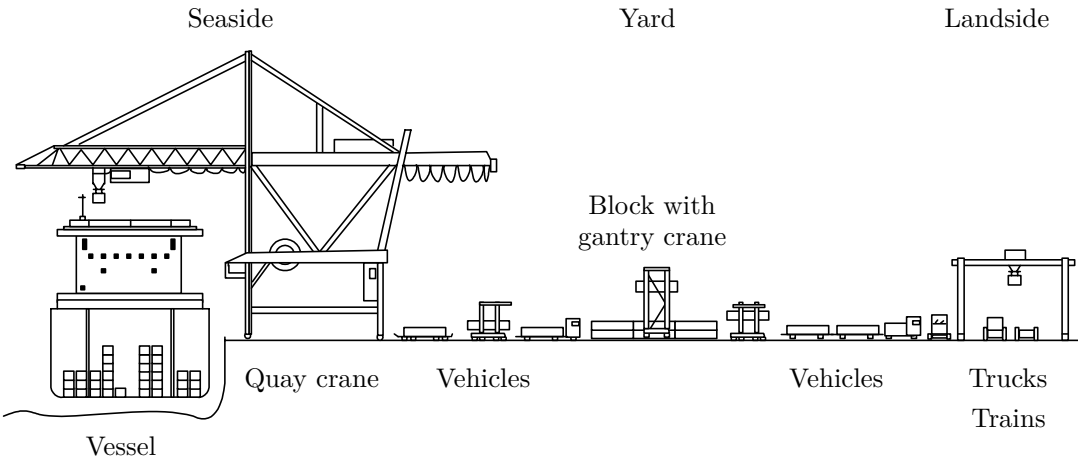


Figure 2.4: Schematic side view of a container terminal (adapted from Steenken et al., 2004).

These areas are interconnected by various specialized vehicles capable of carrying containers. This is usually known as *horizontal transport*. Yard trucks and straddle carriers are the most common manned vehicles. The former can only transport containers, and thus require cranes for both loading and unloading operations, while the latter can also load and pile up containers. Both kinds of vehicles have their robotic counterparts: Automatic Guided Vehicles (AGV) and Automated Lifting Vehicles (ALV), respectively. They are usually slower than manned vehicles and involve higher investment, so they are preferred where labour costs are high.

The first planning problem confronted in a container terminal, as in any other organization, is the assignment of workers to tasks and the scheduling of these tasks. The equipment is operated and supervised by specialized workers in different shifts along the day, which must be properly organized in order to meet the work demands in each area. An efficient division and distribution of labour is fundamental to the smooth functioning of the terminal. Meisel (2009) reviews some studies that address these problems.

The rest of this section describes each area and its main operational planning problems, and then outlines how they are usually integrated.

2.4.1 Seaside

The seaside consists of one or more quays with several quay cranes (QC). The terminal operator and the operator of each vessel usually agree in advance on certain conditions such as the expected time of arrival, the latest departure time and a series of penalty clauses. Vessels may wait in the anchorage, outside the port, until they are given an actual berthing time. Once the berthing of a vessel has been completed, the QCs start to load/unload the planned containers. Quay cranes are mounted on rails and thus can move along the quay to serve moored vessels. The *handling time* of a vessel, also known as its *processing time*, depends upon the number of QCs serving it. A loading or unloading operation of a container performed by a quay crane is called a *move*, and the number of moves per hour is a measure of crane productivity. According to practitioners consulted by van Marle (2015), real crane productivity ranges between 20 and 30 moves per hour, which is far from the technical limit of 50–60 moves per hour (Steenken et al., 2004). The maximum number of QCs able to work simultaneously on a single vessel depends upon its length. Van Marle also reports that nowadays most terminals deploy up to six cranes on the largest vessels, while only in exceptional cases have more than 10 cranes been deployed simultaneously on the same vessel.

The operational planning problems and their multiple variants posed by seaside management have been extensively reviewed by Bierwirth and Meisel (2010, 2015) and Carlo et al. (2015). Following the terminology used in these works, the main problems are:

- The *Berth Allocation Problem* (BAP). This concerns the assignment of both berthing time and position on the quay to vessels arriving within a time horizon. It is an optimization problem in which the objective function is generally a cost function that has to be minimized. The costs are usually related to the time vessels spend at the port, so it is intended to provide them with a fast and reliable service. Since the present investigation focuses on this problem, a more detailed description is provided in Section 2.5.

- The *Quay Crane Assignment Problem* (QCAP). This is the problem of assigning quay cranes to vessels calling at the terminal to carry out their loading and unloading operations. It takes into account the berth plan and the volume of containers for each vessel. Given that the number of cranes working on a vessel determines its handling time, the main objective is to minimize delays, which usually incur contractual costs. For this reason, this problem is frequently tackled together with

the BAP, as is the case in this study. Section 2.6 describes the problem in more detail.

- The *Quay Crane Scheduling Problem* (QCSP). This problem concerns the scheduling of loading and unloading tasks for a single vessel among its assigned quay cranes. The objective is usually to minimize the makespan of the QC schedule, as it represents the handling time of the vessel.

- *Stowage Planning.* This concerns the assignment of containers to empty positions (slots) within a vessel and consists of two phases. The first is carried out by the vessel operator and produces a rough plan in which only the class of containers to be assigned to each position is determined. The second phase is performed by the terminal operator on the basis of that plan and concerns the assignment of specific containers to empty positions according to their class. The most common objective in this second phase is to minimize container reshuffles both within the vessel and within the yard. A *reshuffle* is the temporal removal of a container in order to access a target container.

2.4.2 Yard

The yard is the area where containers are temporarily stored before being loaded onto a vessel (*export containers* and *transshipment containers*), or onto trains or trucks (*import containers*). It consists of a number of container *stacks* separated by the traffic lanes used by internal vehicles (Figure 2.5). Each stack may consist of one or more *piles*, forming *bays*, and each pile may consist of one or more *tiers*. Containers are usually segregated in different stacks according to their destination (import, export or transshipment), their characteristics (normal, refrigerated, hazardous, etc.), or their state (filled or empty). The stacks may form compact units called *block stacks* (Figure 2.6), or may be separated enough to make it possible to straddle carriers so as to access all the positions, thus forming *linear stacks* (Brinkmann, 2011). Blocks are operated by specialized yard cranes, which can be Rail Mounted Gantry Cranes (RMGC), Rubber Tired Gantry Cranes (RTGC), or Overhead Bridge Cranes (OBC). All these types of cranes can pile containers up to six tiers high, and even higher in the case of OBCs, but usually the last tier is left empty to allow the crane to move along the bays while carrying a container. At least one row of each block is frequently dedicated to making it possible for the crane to load or unload containers from the vehicles.

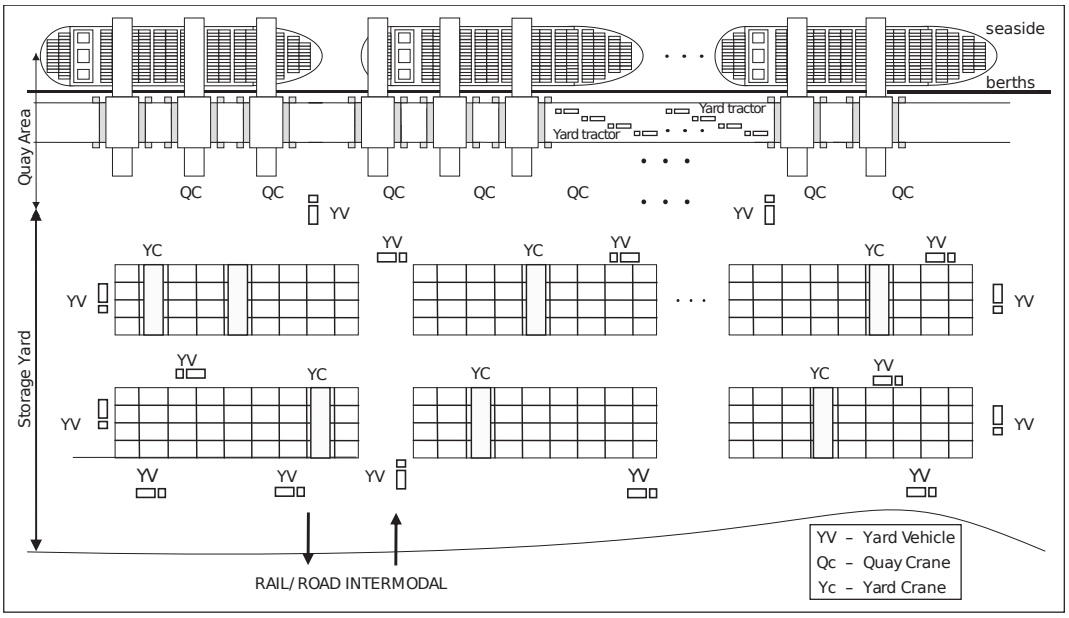


Figure 2.5: Schematic overhead view of a container terminal (Gudelj et al., 2010).

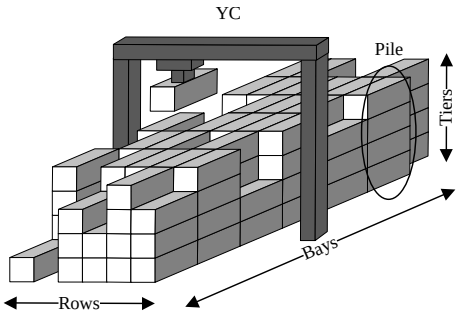


Figure 2.6: Block of containers (Gharehgozli et al., 2016).

The operational management of the yard comprises the decisions related to space usage and location of containers, vehicle dispatching, and management of yard cranes. Researchers have identified many operational planning problems in these contexts and have proposed different ways of classifying them. A comprehensive review is provided by Zhen et al. (2013). The most studied sets of operational planning problems are the following:

- *Storage allocation.* This is done in two phases: the space planning phase, and the real-time locating phase. In the first phase a storage space is reserved and pre-planned before containers arrive at the yard. The second phase is performed when the containers arrive at the terminal by vessel or by land modes of transport and concerns the assignment of a specific location to each individual container.

Following the criteria proposed by Lehnfeld and Knust (2014), many problems in this second phase can be classified as *storing loading problems*, when a location is to be assigned to new containers; *storing unloading problems*, when the retrieval of containers involves a potential relocation; and *premarshalling problems*, when container relocations may ease future loading and unloading operations. Common objectives in the first phase are minimizing vehicle travel distances between the yard and the corresponding vessel and minimizing the congestion of yard cranes and vehicles. In the second phase, the objective is usually to minimize yard crane movements or minimize the number of container relocations (Kim and Lee, 2015). The papers of Carlo et al. (2014a), Lehnfeld and Knust (2014), and Caserta et al. (2011) present in-depth reviews of these problems.

- *Vehicle dispatching.* These problems concern the assignment of vehicles to containers in order to transport them through the yard to and from the seaside/landside. Vehicles can also be assigned to quay cranes to transport their respective containers. Other problems focus on routing and traffic control. In general, their objective is to minimize the transportation time. These problems can be integrated with related problems such as yard crane scheduling, location assignment, quay crane scheduling, or combinations thereof. An extensive review of transport operations is provided by Carlo et al. (2014b).

- *Yard crane scheduling.* These problems involve determining the route of each yard crane within a block. The main objective is to minimize the makespan and the total delay time of all requests carried out by a yard crane. In the case in which a single crane is considered, the problem entails determining the number of containers to manage in each bay as well as the sequence of bays that are to be visited. When two or more yard cranes are considered, the cooperation and the interferences between them must be taken into account.

2.4.3 Landside

In the landside, containers are loaded/unloaded to and from external trucks and trains. Large container terminals serve thousands of trucks and several trains a day. The railway tracks lead into the terminal, while external trucks must be checked in the gatehouses. If the yard is operated by gantry cranes, these are also used to perform the loading and unloading operations. Internal vehicles are needed to transport the containers loaded or unloaded by gantry cranes serving a train, whereas in the case of external trucks they can be directly served at dedicated blocks in the external part of the yard. If straddle carriers are used instead, they can perform such operations directly on the train or on the trucks located in the delivery parking area. The arrival of export containers in the terminal from the hinterland is usually booked by costumers in order to reduce both the waiting times of external trucks and the congestion at the gates. The pick up of import containers is also booked by their corresponding consignees. The operational planning problems concern the scheduling of these appointment times, the assignment of the horizontal transport demanded by loading and unloading operations, and the

scheduling of the gantry cranes that serve trains and trucks. Papers on these issues were reviewed by Stahlbock and Voß (2008) and Steenken et al. (2004).

2.4.4 Integration of planning decisions

In real-world situations, planning problems are organized hierarchically and solved in different phases using specialized software known as *Terminal Operating Systems* (TOS). These systems provide tools for managing, planning, scheduling, and executing the multiple processes involved in the normal functioning of a terminal. Academic research generally contributes to these systems by providing new and better solution methods for the problems found in each area. The overall performance of the terminal can thereby be further increased, provided that terminal operators solve the bottlenecks that arise between processes resulting from the lack of equipment and workers. A review of the most recent developments in this field is provided by Kim and Lee (2015).

An additional way of improving the technical efficiency of the terminal is by integrating the operational problems previously outlined. Several optimization problems within and between the main areas of container terminals are increasingly tackled in an integrative manner in academic literature. This is possible due to the growing computational power and parallelization of computers, the continuous enhancement of the software specially designed to solve mathematical programming models (also known as *solvers*), and the development of new tailored heuristic algorithms.

Following previous authors, Bierwirth and Meisel (2010) distinguish *deep integration* from *functional integration* and propose a scheme for representing the different integration alternatives. Let us consider two different problems A and B. Deep integration, expressed by $\boxed{\text{A, B}}$, merges both problems in a monolithic problem formulation, whereas functional integration keeps an independent formulation for each problem, solving them sequentially either in a feedback loop ($\text{A} \rightleftarrows \text{B}$) or in a preprocessing scheme ($\text{A} \rightarrow \text{B}$). In the feedback loop scheme, the solution of the top-level problem is used as part of the input of the base-level problem. The solution obtained for this problem is then used to adapt the input of the top-level problem and solve it again. This process is repeated until a steady state is reached. In the pre-processing scheme, the base-level problem is solved first to adapt the input of the top-level problem accordingly, and then both problems are solved sequentially.

Given that the deep integration approach formulates a new problem that integrates the decisions from at least two problems, it theoretically makes it possible to solve the whole problem to optimality. However, the problem is consequently more difficult and therefore real-world instances may be unsolvable in practice. Functional approaches are limited only by the difficulty of the problems taken in isolation, but they perform a refinement process that may not achieve an optimal solution.

The literature reviews concerning the problems faced in each area also discuss the integrative approaches proposed by OR researchers. In the seaside, problems such as the BAP and the QCAP are often tackled following various integration schemes. The next sections describe the Berth Allocation Problem, the Quay Crane Assignment Problem and their integration in depth, as they are the problems tackled in this thesis.

2.5 Berth Allocation Problem

The *Berth Allocation Problem* (BAP) in a port terminal is confronted by the terminal operator and concerns the planning of quay usage by vessels whose arrival is expected within a given time horizon. Thus the terminal operator has to assign a berthing time and a position to each vessel under given physical and time constraints. Since the terminal operator is interested in achieving the most efficient quay usage, the objective is to optimize one or more given functions. In doing so, it can use information about the vessels known in advance, such as their length or their expected time of arrival. A solution to the problem, or *berth plan*, can be rendered as a space-time diagram in which the vertical axis represents the mooring positions, the horizontal axis represents time, and each vessel is a rectangle whose length represents its handling time and whose width represents its length (Figure 2.7). A feasible solution is a berth plan in which the berthing times and positions assigned to the vessels satisfy all the given constraints. The basic constraints prevent vessels from being allocated impossible times and positions: a vessel cannot be moored 1) outside the quay; nor 2) at the same time and position as any other vessel; nor 3) before its arrival. However, other constraints can be considered, such as restrictions on the departure time or on the berthing positions according to the water depth. Since a finite set of vessels can be arranged in multiple ways in time and space to form a feasible berth plan and the objective is to obtain the optimal berth plan from the set of all the feasible berth plans, the BAP can be classified as a combinatorial optimization problem.

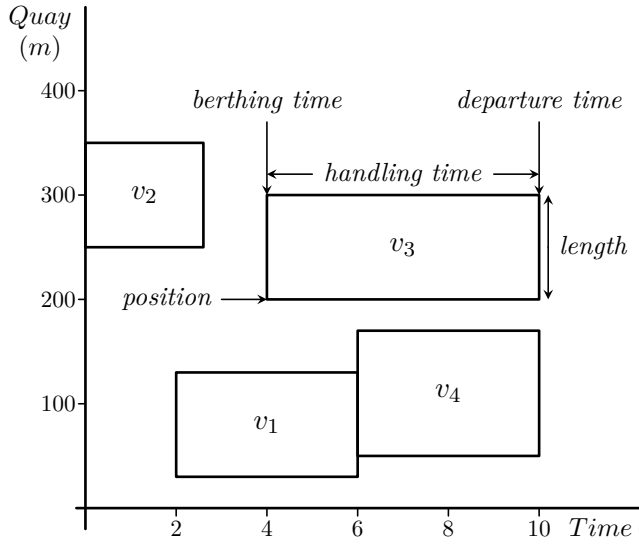


Figure 2.7: Graphic representation of a berth plan with 4 vessels and a quay 400 m long.

2.5.1 Variants

There are a number of variants of the BAP depending upon the assumptions made on various attributes. According to Bierwirth and Meisel (2015), who synthesized well-known criteria present in the literature, these are the spatial attribute, the temporal attribute, the handling time attribute, and the performance measure. In particular, the alternatives for each attribute are as follows:

- **Spatial attribute**. This concerns the characteristics of the quay layout, which can be (Figure 2.8):
  - *Discrete*. The quay is divided into several sections, called *berths*. In each berth, only a single vessel can be processed at a time. This partitioning may result from the actual layout of the quay or from organizational criteria.
  - *Continuous*. The quay is treated as a continuous segment along which vessels can be moored. The planning in this case is more complex than in the discrete case, but it allows better utilization of the space.
  - *Hybrid*. This is a discrete layout in which each berth may admit more than one vessel or vessels may occupy more than one berth under certain conditions. Here we also find the special case of indented berths, in which two berths are opposite each other.

  Besides the quay layout, draft constraints may limit quay/berth usage depending upon the water depth required by vessels.

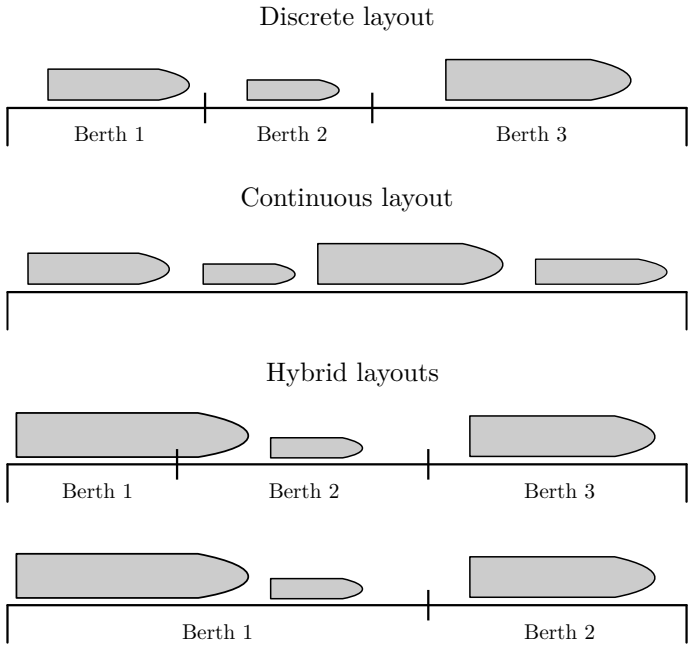


Figure 2.8: Types of quay layout.

- **Temporal attribute**. This concerns restrictions on berthing times and departure times. In particular, a primary consideration is the arrival of the vessel, which can be:
  - *Static*. No arrival times are given for the vessels or they are soft constraints. In the first case, it is considered that vessels are already waiting at the port, so they can berth at any time. In the second case, a vessel can berth before its expected arrival time at the expense of the cost of speeding up its arrival at the terminal.
  - *Dynamic*. A fixed arrival time is given for each vessel, so it cannot be moored before that time.
  - *Stochastic*. Arrival times are obtained from random distributions or may correspond to random scenarios.
  - *Cyclic*. Vessels call at the terminal periodically in fixed time intervals.

  Besides the considerations on arrival times, additional constraints may determine a maximum waiting time or a maximum departure time allowed for each vessel.

- **Handling time attribute**. The handling times of vessels can be:
  - *Fixed*. They are given in advance as an input to the problem and cannot be changed. For each vessel, the handling time may derive from a prior estimate made by both the terminal and the vessel operators.
  - *Position-dependent*. The handling time of a vessel depends upon its berthing position.
  - *QCAP-dependent*. They are obtained by solving the Quay Crane Assignment Problem jointly with the BAP. Thus the handling time of a vessel depends on the number of cranes assigned to it.
  - *QCSP-dependent*. They are obtained by solving the Quay Crane Scheduling Problem jointly with the BAP. Thus the handling time of a vessel depends on the work schedules of the assigned cranes.
  - *Stochastic*. They are determined from random distributions.

- **Performance measure**. This concerns the factors taken into account in the objective function which is to be optimized. In most cases, the objective is to minimize a function comprising, for each vessel, elements such as waiting time before berthing, handling time, completion time, delay, speed-up cost, deviation from the desired position on the quay, or usage of equipment and manpower. Different weights are commonly used to set priorities between the elements considered and the vessels. In general, the time spent by each vessel at the terminal is usually considered in one way or another, since it is a crucial factor in terminal competitiveness, as we saw in Section 2.2.

2.5.2 Research overview

Combining the different alternatives for each attribute gives rise to a great variety of versions of the BAP. Researchers in OR have studied many of them since the 1990s, especially during the last decade. The first papers used serial representations of the problem and applied queuing theory, heuristics or simulation techniques. By the late 1990s, various versions of both the discrete and the continuous BAP were formalized as mathematical programs. Li et al. (1998) formulated the continuous static BAP as a multiple-job-in-one-processor scheduling problem in which the objective was to minimize the makespan. A similar BAP was formulated by Guan et al. (2002) as a multiprocessor task-scheduling problem with the objective of minimizing the total weighted completion time. In doing so, they proved the continuous static BAP to be $\mathcal{NP}$-hard and, therefore, a computationally difficult problem. Lim (1998) formulated a continuous dynamic BAP in which the berthing time is not a decision variable as a restricted form of the two-dimensional packing problem, thereby proving it is also $\mathcal{NP}$-hard. The discrete static BAP was formulated by Imai et al. (2001) and Hansen and Oğuz (2003) as an integer three-dimensional assignment problem in which the objective is to minimize the sum of the waiting and handling times of vessels. Additionally, they also formulated the discrete dynamic BAP and showed it is $\mathcal{NP}$-hard even in the case of a single berth, as it reduces to the problem of minimizing completion time with release dates on a single machine.

Many studies followed these early approaches with the aim of both capturing more real-world aspects and improving on the solution methods proposed. Given that these problems are $\mathcal{NP}$-hard, the existence of a fast algorithm for solving instances of any size to optimality is unlikely, and thus the performance differences between the solution approaches become decisive. Indeed, the computational complexity of BAP instances depends mainly upon the number of vessels arriving within the time horizon, which is a factor of great interest for terminal operators. Since container ship traffic continues to grow, they demand fast solution methods capable of obtaining optimal or near-optimal solutions for increasingly large instances. Similarly, researchers are interested in analysing these problems rigorously and finding out which solution strategies yield the best results under given conditions. Hence during the last decade the number of research papers on the BAP has experienced an upsurge.

The studies on the BAP have been systematically reviewed and classified in recent years by Theofanis et al. (2009), Bierwirth and Meisel (2010, 2015), and Carlo et al. (2015). According to these analyses, both discrete and continuous variants continue to attract most attention, while problems with hybrid layouts or draft constraints are in the minority. The dynamic version predominates clearly over the static one and the cyclic variant has recently appeared as a novelty. The research also seems to concentrate on versions with handling times that depend upon the position of the vessels or the number of cranes assigned, and some authors are beginning to tackle stochastic variants (see, for example, Ursavas and Zhu, 2016). With respect to the objective function, waiting and handling times are taken into account in most studies, and penalties related to delays and deviations from preferred positions are also common. Around 75% of the solution methods proposed are metaheuristics, whereas the remaining are exact approaches, based

generally on integer linear models implemented on solvers. Exact methods are able to solve instances with a few vessels to optimality, while metaheuristics usually obtain good solutions in both small and large instances in very short computation times. New real-world aspects considered in recent studies include the impact of tides, the operation ranges of cranes, and fuel consumption and emissions.

Another key resource that decisively affects terminal performance is quay cranes. The next section introduces the crane assignment problem and its different versions.

2.6 Quay Crane Assignment Problem

In addition to quay space, the cranes located alongside the quay are also a scarce resource that poses an additional planning problem. Given a berth plan, the maximum number of cranes that can serve each vessel concurrently, and the corresponding volume of containers to be loaded and unloaded, the *Quay Crane Assignment Problem* (QCAP) concerns the assignment of cranes to vessels with the aim of making the most efficient use of them. In general, it is assumed that all the QCs can move to every position on the quay provided that they do not cross each other. It is also common to consider a minimum number of QCs that must serve each vessel concurrently, as this is usually specified in contracts between vessel and terminal operators. Given that the number of cranes working on a vessel determines its handling time, the main objective is frequently to minimize contractual penalties caused by departure delays.

This problem can be decomposed into two subproblems:

1. QCAP(number), the assignment of a number of cranes to each vessel.
2. QCAP(specific), the assignment of specific cranes to each vessel.

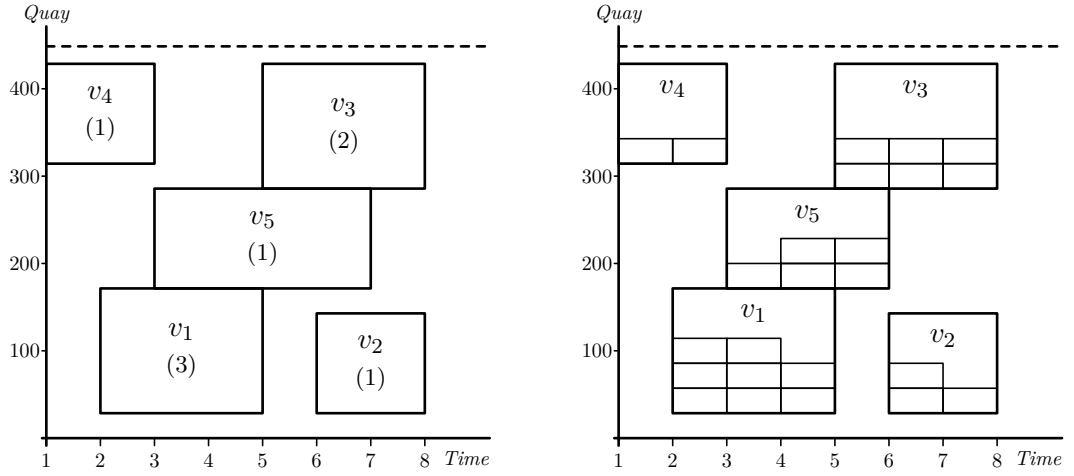
The first problem must respect the number of cranes available at the quay, the maximum and minimum number of cranes allowed for each vessel, and the berthing and departure times of the vessels. The second problem must take into account that cranes cannot cross each other and thus working cranes may impede adjacent cranes from moving to serve other vessels.

2.6.1 Variants

The assignment of a number of cranes to each vessel gives rise to two variants of the problem (Figure 2.9):

- *Time-invariant.* The number of cranes assigned to a vessel is fixed during its handling. Consequently, the specific cranes assigned cannot be reassigned to any other vessel until its handling has finished.
- *Variable-in-time.* The number of cranes assigned to a vessel can change during its handling. Therefore, the specific cranes assigned may also change. This version entails segmenting time into regular periods so that the crane assignment can be determined for each vessel and time slot. As a consequence, the number of decision

variables is greater than in the time-invariant version, and therefore it is generally more difficult to solve.



(a) Time-invariant QCAP solution for a terminal with four quay cranes. The number of cranes assigned is shown in parentheses.

(b) Variable-in-time QCAP solution. The cranes are represented as rectangles inside each vessel assignment.

Figure 2.9: QCAP variants.

Both versions of the crane assignment are interesting in practice and are receiving increasing attention in the literature. However, since a vessel's handling time is mostly determined by the number of cranes assigned to it, this problem is often tackled together with the BAP.

2.6.2 Integration with the BAP

Including the crane assignment in the process of determining the berthing times and positions enable us to adjust the handling time of vessels so that more vessels can be scheduled in less time and the overall cost is reduced. Thus the integration of the BAP and the QCAP makes it possible to attain better quay usage at the expense of greater computational effort. The integrated problem is usually known as the *Berth Allocation and Quay Crane Assignment Problem*, although no acronym is generally accepted. This problem also presents different versions, depending on the kind of crane assignment assumed, either time-invariant or variable-in-time. Moreover, given that the QCAP can be decomposed into the problem of assigning the number of cranes and the problem of assigning the specific cranes, several different combinations have been addressed in the literature (Bierwirth and Meisel, 2010, 2015). Such approaches can be referred to using the scheme proposed by these authors, already described in Section 2.4.4.

The pioneering study of Park and Kim (2003) tackled a BAP, QCAP(number) → QCAP(specific) in which the first phase was solved by means of a mixed integer

linear model and the second phase by means of a dynamic programming algorithm. Since then, especially during the last decade, numerous models and heuristics have been proposed to solve different integration approaches. The most common schemes are BAP, QCAP(number), in which only the number of cranes is determined for each vessel; and BAP, QCAP, in which both the number of cranes and the specific cranes are assigned. In the rest of this thesis, the former will be referred to as the BACAP, and to the latter as the BACASP. As well as the functional scheme proposed by Park and Kim (2003), some authors have also proposed other functional approaches, such as BAP $\rightleftarrows$ QCAP and BAP, QCAP(number) $\rightleftarrows$ QCAP(specific). The differences in performance observed between the proposals cannot be attributed solely to the scheme applied in each case, as they also depend upon the specific models and algorithms developed. Therefore, the choice between them must be made according to the specific needs and objectives in each case. In theory, functional approaches can deal with larger instances than deep approaches in the same time, but they may obtain worse solutions on small instances, on which deep approaches can obtain optimal solutions in short times.

Problems such as the BAP, the BACAP, and the BACASP are tackled using different techniques. The next section presents the most common methods used to address them.

2.7 Methodology

Combinatorial optimization problems are often tackled by using complementary methods with characteristics that make them suitable for different purposes. This investigation makes extensive use of the main techniques that exist in the field: integer programming, heuristics, and matheuristics. These approaches are outlined in this section.

2.7.1 Integer linear programming methods

Integer linear programming involves modelling optimization problems by specifying a finite set of non-negative integer decision variables, a finite set of constraints represented as linear inequalities, and a linear objective function as follows:

$$\max \quad cx \tag{2.1}$$

$$\text{subject to} \quad Ax \leq b \tag{2.2}$$

$$x \geq 0 \quad \text{integer} \tag{2.3}$$

The column n-vector x represents the decision variables, the row n-vector c contains the coefficients applied to the variables in the function that is to be maximized (2.1), and the $m \times n$ matrix A together with the column m-vector b expresses the constraints (2.2). The minimization problem is analogous. An instance of a problem is determined by given values of c, A, and b. A combination of the values of the variables x that satisfies the constraints is called a *feasible solution*, and the set of all the feasible solutions is known as the *feasible region* or the *solution space*. A feasible solution that optimizes the objective function is called an *optimal solution*. This kind of formulation, known as an *integer linear program*, allows us to represent the search for the values of x that satisfy

the constraints and optimize the objective function. A formulation of the same form in which some of the variables, but not all, allow non-negative real values is called a *mixed integer linear program* (MILP). Both pure and mixed integer linear programs are usually tackled using the same techniques. Conforti et al. (2014) and Chen et al. (2011) present comprehensive introductions to integer programming and related solution methods.

The modelling of a real-world combinatorial optimization problem as an integer program entails a process of analysis in which its elements must be identified and abstracted and the relations between them are to be expressed precisely to achieve a correct formulation. This process is not systematized and relies mainly on a thorough study of the problem and the different alternative ways of expressing it. Thus a single problem may be modelled through different equivalent formulations consisting of different variables and inequalities.

Formulating combinatorial problems as integer programs makes it possible to use solution strategies capable of solving them to optimality, called *exact methods*, such as *Branch & Bound* and *Branch & Cut*. Such algorithms constitute the core of *solvers*, the software applications specially designed to solve the problems by using their mathematical formulations. Although in theory these techniques guarantee that an optimal solution will eventually be found, in practice their performance depends drastically upon the number of variables and the number of constraints in the integer program corresponding to each problem instance, so they may not reach any optimal solution in a reasonable time. These numbers depend directly upon two factors: the instance size and the problem formulation. Whereas these methods cannot deal efficiently with the instance size and consequently suffer the *combinatorial explosion*, the problem formulation may be enhanced or replaced by an alternative model to improve its performance. Thus many researchers in OR propose new formulations and tailor-made exact strategies for well-known problems to solve larger instances optimally in shorter computation times.

Another kind of solution method with different characteristics and scope is heuristic algorithms, which are outlined in the following section.

2.7.2 Heuristic methods

A *heuristic* is a procedure designed to find good solutions to a given difficult problem or class of problems by exploiting its characteristics, using rules of thumb or applying successful ideas (see Laguna and Martí, 2013). This kind of procedure is implemented by means of programming languages and run on computers. Heuristics may be simple or sophisticated and are usually devised taking computing time constraints into account. In general, they combine rules and criteria that lead to *high-quality* solutions with processes that generate *diverse* solutions. Thus they can obtain good solutions in short computation times to instances of a wide range of sizes at the expense of being unable to guarantee optimality, unlike exact methods.

Heuristic procedures can be designed following general strategies, known as *metaheuristics*, that proved to be successful in the past. Nowadays, several metaheuristics are widely recognized and applied in OR. Books that introduce and show examples of their application were published by Gendreau and Potvin (2010) and Siarry (2016). Ac-

cording to the way solutions are manipulated, Sörensen and Glover (2013) classify them as:

- *Local search metaheuristics.* These work by making "small" changes to a single solution. The set of solutions that can be obtained in this way from a given solution is called the *neighbourhood* of that solution. Thus these metaheuristics explore the solution space by moving through the solution neighbourhoods applying different strategies and criteria. Some examples are *Tabu Search* (TS), *Iterated Local Search* (ILS), *Variable Neighbourhood Search* (VNS), and *Guided Local Search* (GLS).

- *Construction metaheuristics.* These add elements to a partial solution step by step until a complete solution is attained. This process is commonly repeated randomizing the choice of element to produce diverse solutions. For example, *Greedy Randomized Adaptive Search Procedure* (GRASP), *Ant Colony Optimization* (ACO), and *Look-ahead strategies.*

- *Population-based metaheuristics.* These first construct a set of initial solutions and then select and combine these solutions repeatedly. *Evolutionary Algorithms* are the most widespread methods of this kind, outstanding among which are *Genetic Algorithms* (GA). These encode solutions as lists of elements (*chromosomes*) in order to both recombine them using operations inspired by sexual reproduction (*crossover*) and alter them by applying random changes (*mutations*). Other population-based methods are *Scatter Search* and *Path Relinking.*

- *Hybrid metaheuristics.* These combine ideas from different metaheuristics. Some examples are *Memetic Algorithms*, which combine GA and local search heuristics; and GRASP, which combines construction and local search methods.

Combining exact and heuristic methods leads to so-called matheuristics.

2.7.3 Matheuristic methods

A *matheuristic* combines a heuristic procedure with a solution method based on mathematical programming models in order to take advantage of both approaches (see Sörensen and Glover, 2013). As they can be integrated in very different ways and frequently provide notable improvements, this topic is attracting increasing attention in the OR community. In general, they present a structure that includes a guiding process and an application process, so either the heuristic controls the calls to the exact method or the exact method controls the calls to the heuristic (Caserta and Voß, 2010). For example, exact methods can be used to solve subproblems or specific cases of the problem optimally within a heuristic framework, aiming to improve the quality of the solutions. Conversely, heuristics can be used within an exact method to speed up the process. Modern solvers, in fact, exploit partial information to enhance branch-and-cut algorithms by applying this approach. Further information is provided in the general introduction to the topic published by Maniezzo et al. (2010).

Chapter 3

The continuous Berth Allocation Problem in container terminals with multiple quays

3.1 Introduction

In the previous chapter we have seen that up to now, in all previous studies dealing with the BAP, the quay has been considered as a single line of a given length (*continuous* variant) or as a set of quay sections, each one admitting a single vessel (*discrete* variant) or a limited number of vessels under special conditions (*hybrid* variant). However, none of these approximations seems good enough to make the most of the quay space in terminals with more than one quay. This is the case, for instance, in one of the terminals in the port of Valencia (Spain), which consists of two quays. Many other examples can be found in other ports around the world. In Singapore, Keppel terminal is divided into four quays, and Tanjong Pagar and Brani terminals into three. Maher and APM terminals in the port of New York and New Jersey (USA) have two quays each, while Terminal 1 of Jebel Ali Port (Dubai) has five quays. It could be argued that algorithms designed for a single quay or several quay sections can be adapted to the multiple quay case. Nevertheless, addressing the problem as it really is, with multiple continuous quays, can lead to better algorithms and, more importantly, can allow us to include more realistic characteristics. Restrictions such as those affecting a given quay for technical or contractual reasons, or the assignment of different costs for mooring at different quays, can be easily considered in algorithms specially designed for such terminals.

This chapter extends the dynamic and continuous Berth Allocation Problem to address the case of terminals with several quays. Considering multiple quays to accommodate a set of calling vessels adds a problem of assigning vessels to quays to the basic problem of deciding the time and position on the quay for each vessel. In order to solve this problem, both exact and heuristic methods are proposed. First a mixed integer linear model is presented, with the objective of minimizing the sum of all operational costs involved. Since the model is capable of solving instances only up to a certain size,

several heuristics and a Genetic Algorithm (GA) are also proposed to obtain good solutions on large instances in short computing times. The solutions obtained by the GA are further improved by a Local Search procedure. All these solutions methods are tested and compared with existing approaches in the literature through different computational experiments over different sets of instances. This chapter is an extended version of the following published paper:

> Frojan, P.; Correcher, J.F; Alvarez-Valdes, R.; Koulouris, G; and Tamarit, J.M. 2015. **The continuous Berth Allocation Problem in a container terminal with multiple quays**. *Expert Systems with Applications*, 42(21):7356–7366.

The rest of this chapter is organized as follows. Section 3.2 reviews previous studies in the literature relating to the BAP. Next, Section 3.3 describes the problem and its assumptions. Section 3.4 presents a mathematical programming formulation for the problem, and Section 3.5 explains the different elements that make up the metaheuristic approach adopted: the Genetic Algorithm scheme, the constructive algorithms, and the Local Search procedure. Section 3.6 presents the computational experiments with the integer formulation and the Genetic Algorithm. Finally, in Section 3.7 some concluding remarks are made.

3.2 Literature review

As we saw in the previous chapter, during the last decade the Berth Allocation Problem has been the object of numerous studies, which have been reviewed in recent publications. This review focus on the continuous BAP with handling time estimates given in advance, especially on those papers that show distinctive characteristics or are related to the present approach. Some representative examples of discrete BAP approaches are the studies of Ting et al. (2014), Lalla-Ruiz et al. (2012), and Hansen et al. (2008).

After the early investigations on the continuous BAP carried out by Li et al. (1998) and Lim (1998), many other researchers have addressed this variant of the problem. Park and Kim (2002) tackled a continuous dynamic BAP in which the objective was to minimize the delay cost and the cost associated with deviation from the desired position of each vessel, which was included as a novelty. They formulated this problem by means of a tailor-made mixed integer linear model and used Lagrangian relaxation to solve small instances. The same problem was addressed by Kim and Moon (2003), who proposed a Simulated Annealing heuristic capable of obtaining near-optimal solutions in computation times acceptable in practical situations. Cordeau et al. (2005) modelled the discrete BAP as a vehicle routing problem with time-windows and solved it by a Tabu Search algorithm which they then adapted to the continuous case. Wang and Lim (2007) considered the possibility of not berthing all the vessels within the time horizon as well as an objective function that combined rejection costs, delay costs, and the cost of berthing a vessel away from its desired position on the quay. To solve this problem, they proposed a Stochastic Beam Search (SBS) which proved capable of solving instances

with up to 400 vessels. Lee et al. (2010) developed a new integer linear model and two GRASP algorithms, which outperformed the SBS of Wang and Lim (2007) on instances with up to 200 vessels. Some authors also integrate the BAP with certain yard planning problems, as in the case of the study carried out by Zhen et al. (2011). However, most recent studies address the BAP jointly with the QCAP, which will be reviewed in the following chapter.

Although studies on the BAP commonly consider a single objective function, usually combining different costs into a linear expression, there is an increasing number of papers that consider a multi-objective approach. Zhen and Chang (2012) included maximization of schedule robustness, which is measured taking into account the time between vessels occupying the same berth space. Xu et al. (2012) also considered uncertainty in the vessel arrival and handling times and used time buffers between the vessels occupying the same berthing location. The objective was to optimize a combination of service quality and solution robustness, for which they proposed a solution method that integrated Branch-and-Bound and Simulated Annealing. Du et al. (2011) introduced fuel consumption and vessel emissions into the problem. They ended up with a bi-objective non-linear model, but by means of second order cone programming and the ϵ-constraint approach for bi-objective optimization, they showed that if the speed of vessels is controlled it is possible to reduce fuel consumption and vessel emissions significantly, while keeping waiting costs of the vessels at the terminal at a satisfactory level. This study was improved by Wang et al. (2013), who proposed two quadratic outer approximations capable of obtaining results in a more efficient way.

The continuous BAP with multiple quays has not been studied at the operational level considered in this study, but there are several papers dealing with multi-terminal allocation problems at a tactical or strategic level. Hendriks et al. (2010, 2012) considered the case in which several terminals are managed by the same operator and the problem cannot be solved considering one terminal at a time. Their objective was the strategic allocation of cyclically calling vessels, taking into account operational costs at each terminal and transportation costs between terminals. The solution approach was based on an integer linear model and was able to solve real-world instances in satisfactory computing times. Lee et al. (2012) also considered several terminals in their study of a berth and yard allocation problem for a transshipment hub. They proposed a two-level heuristic algorithm which produced high quality solutions efficiently.

Also related to the continuous BAP with multiple quays is the case of hybrid layouts, in which the quay is divided into sections and the condition of assigning exactly one vessel to each section is relaxed, allowing more than one small vessel to occupy the same section or a large vessel to occupy more than one section. An example of this is the study of Imai et al. (2013), in which the sections were designed for mega-ships and could be occupied by two smaller feeders. Even more closely related to the research developed here is the study of Cheong et al. (2010), who also considered the quay divided into sections, each one able to accommodate many vessels as long as they did not overlap and the total length of the vessels being served simultaneously did not exceed the length of the section. In this respect, they considered the same terminal layout as the one considered here, although their approach was completely different. They focused on a multi-objective

approach aimed at minimizing both the makespan and the total waiting time of the vessels, and maximizing the degree of adherence to a pre-determined priority schedule. In doing so, they did not take into account other elements, such as the costs of assigning vessels to sections or the desired position of each vessel in each section. Generalizing the continuous variant of the BAP and considering these costs allow us to better exploit the characteristics of the multiple quay layout.

The next section details the elements and costs involved in the generalized continuous Berth Allocation Problem in terminals with multiple quays.

3.3 Problem description

3.3.1 Overview

The continuous BAP with multiple quays consists of a set of quays with known lengths and a set of vessels with known characteristics, and its objective is to determine the berthing time, the quay, and the position on the quay for each vessel so that the total assignment cost is minimized. While the basic problem can be represented in a single space-time diagram, the problem with multiple quays requires a multiple space-time representation, as can be seen in Figure 3.1, in which each space-time diagram shows the distribution of the vessels assigned to each quay.

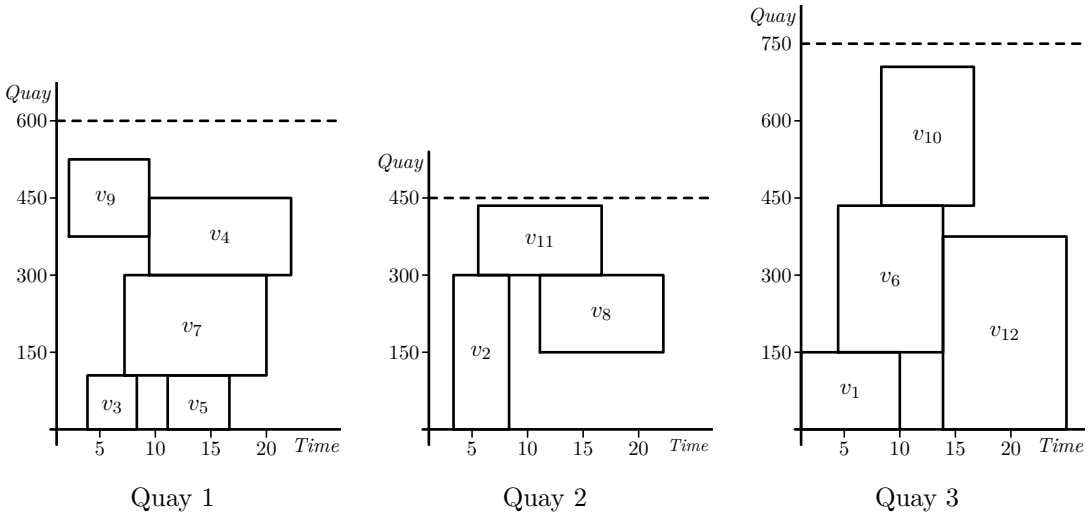


Figure 3.1: A BAP solution with 3 quays and 12 vessels.

The cost of each vessel has four components: the cost related to the waiting time before berthing, which is proportional to the difference between the estimated arrival time and the assigned berthing time; the cost related to the delay, which is proportional to the difference between the desired departure time and the end of the handling of the vessel; the cost of assigning the vessel to the quay; and the cost related to the deviation of the berthing position from the ideal position on the quay. Cost coefficients allow us to set priorities between the vessels or to represent different characteristics and constraints. For

example, some vessel assignments to quays can be forbidden for contractual reasons or because the draft or the machinery on the quay are not suitable for the vessel. Some other possible assignments can simply be made less desirable by assigning high assignment unit costs. Moreover, not all the positions along each quay may be equally desirable. In other planning problems, deviation from the ideal position, which reflects the location of the containers to be loaded or the slots in which to place the containers to be unloaded, may affect the handling time, but in this BAP the handling time is considered fixed and the deviation is included as a cost, because if the containers have to be moved a longer distance and the handling time is fixed, more resources have to be allocated. In terms of the scheme proposed by Bierwirth and Meisel (2010), this problem can be classified as $hybr, draft \,|\, dyn \,|\, fix \,|\, \sum(w_1\, pos + w_2\, wait + w_3\, tard + w_4\, misc)$, although new categories will be necessary to describe it properly as a fully generalized continuous BAP with more than one quay with an specific assignment cost for each quay.

3.3.2 Assumptions

This problem is based on the following assumptions:

- Each position at each quay can accommodate at most one vessel at a time.
- Once a vessel is moored, its position cannot be changed.
- The inter-ship clearance is included in the length of the vessels. In general, for vessels longer than 130 m, this clearance corresponds to 10% of the vessel's length. For small vessels, the minimum clearance is 10 m (Lee et al., 2010)
- The handling time of a vessel is considered fixed and independent of its berthing position. This assumption is reasonable if the quay has enough machinery and workers for the loading/unloading tasks at any moment (Lim, 1998).
- The time for docking and undocking maneuvers is considered to be included in the vessel handling time.
- Once the handling of a vessel has started, it cannot be interrupted, to avoid additional costs.
- Each vessel may have a different relative importance. Therefore, the four cost coefficients specific to each vessel may be used to set priorities.

3.3.3 Parameters

The following data are known in advance and define an instance of the problem:

- Set of quays: Q. Each quay $q \in Q$ has a length L_q
- Set of calling vessels: V
- For each vessel $i \in V$, the following information is known:

- Length: l_i
- Estimated arrival time: a_i
- Estimated handling time: h_i
- Desired departure time: s_i
- Ideal position on quay q: d_{iq}
- Cost per unit time of waiting for berth: C_i^w
- Cost per unit time of delay with respect to the desired departure time: C_i^d
- Assignment cost to quay q: C_{iq}^a
- Cost of unit deviation from the ideal position on quay q: C_{iq}^p

In the next section the problem is formulated as a mixed integer linear program considering the assumptions, the description, and the parameters that define the problem.

3.4 A mixed integer linear model

The following variables, objective function, and constraints are defined, thereby extending the model proposed by Park and Kim (2002).

3.4.1 Variables

$$m_{iq} = \begin{cases} 1, & \text{if vessel } i \text{ is moored at quay } q \\ 0, & \text{otherwise} \end{cases}$$

p_i = berthing position of vessel i on the quay to which it is assigned

t_i = berthing time of vessel i

u_i = delay of vessel i

e_i = deviation of the assigned position of vessel i from its ideal position on the quay to which it is assigned

$$\sigma_{ij} = \begin{cases} 1, & \text{if vessel } i \text{ is completely to the left of vessel } j \text{ in the space-time} \\ & \text{diagram, that is, vessel } i \text{ is completely handled before } j \\ 0, & \text{otherwise} \end{cases}$$

$$\delta_{ij} = \begin{cases} 1, & \text{if vessel } i \text{ is completely below vessel } j \text{ in the space-time diagram,} \\ & \text{that is, vessel } i \text{ is positioned completely to the right of vessel } j \\ & \text{looking at them from the quay} \\ 0, & \text{otherwise} \end{cases}$$

3.4.2 Objective and constraints

$$Min \sum_{i \in V} (C_i^w (t_i - a_i) + C_i^d u_i) + \sum_{q \in Q} \sum_{i \in V} (C_{iq}^a m_{iq} + C_{iq}^p e_i) \tag{3.1}$$

s. t.

$$t_i \geq a_i, \qquad \forall i \in V \tag{3.2}$$

$$\sum_{q \in Q} m_{iq} = 1, \qquad \forall i \in V \tag{3.3}$$

$$p_i + l_i \leq \sum_{q \in Q} m_{iq} L_q, \qquad \forall i \in V \tag{3.4}$$

$$u_i \geq t_i + h_i - s_i, \qquad \forall i \in V \tag{3.5}$$

$$e_i \geq p_i - \sum_{q \in Q} d_{iq} m_{iq}, \qquad \forall i \in V \tag{3.6}$$

$$e_i \geq \sum_{q \in Q} d_{iq} m_{iq} - p_i, \qquad \forall i \in V \tag{3.7}$$

$$t_j - (t_i + h_i) - (\sigma_{ij} - 1)T \geq 0, \qquad \forall i, j \in V, i \neq j \tag{3.8}$$

$$p_j - (p_i + l_i) - (\delta_{ij} - 1)L_{max} \geq 0, \qquad \forall i, j \in V, i \neq j \tag{3.9}$$

$$\sigma_{ij} + \sigma_{ji} + \delta_{ij} + \delta_{ji} \geq m_{iq} + m_{jq} - 1, \qquad \forall i, j \in V, i \neq j, \forall q \in Q \tag{3.10}$$

$$\sigma_{ij}, \delta_{ij} \in \{0, 1\}, \qquad \forall i, j \in V, i \neq j \tag{3.11}$$

$$t_i, p_i, u_i, e_i \geq 0, \qquad \forall i \in V \tag{3.12}$$

The objective function (3.1) minimizes the sum of the four types of costs. The first term corresponds to the waiting and delay costs of each vessel, and the second to the assignment and deviation costs of each vessel in its assigned quay. Constraints (3.2) establish that the start of the handling of each vessel is equal to or greater than its arrival time. Constraints (3.3) ensure that every vessel is assigned to a quay, and constraints (3.4) that its position on the quay is valid. The delay of each vessel is defined in constraints (3.5), and the deviation from the desired position in constraints (3.6) and (3.7). Constraints (3.8) and (3.9) define σ and δ variables, and constraints (3.10) prevent overlapping in time and space of vessels assigned to the same quay. Finally, constraints (3.11) and (3.12) define the types of the variables. $L_{max} = \max_{q \in Q}(L_q)$ and T is an upper bound on the total time required to service all the vessels.

In other planning models it is possible to advance the arrival of a vessel with an associated cost, and then t_i can be made lower than the initially estimated a_i. This could easily be accommodated in the model, but as BAP models usually include constraints (3.2), they have been retained.

3.5 Metaheuristic approach

The integer linear program presented in the previous section is very useful for describing the elements of the problem and their interactions, but it can produce optimal solutions

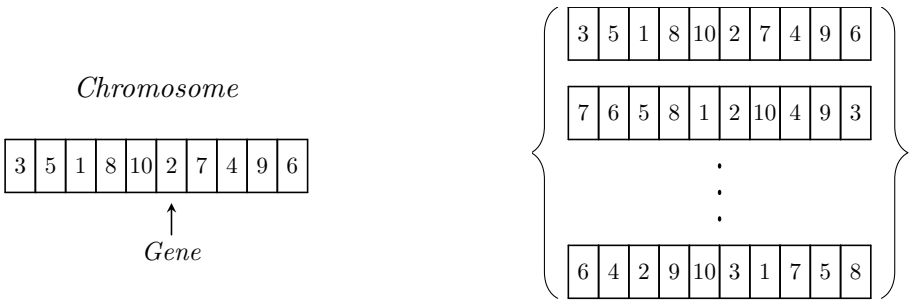
only for small to medium size instances, as will be detailed in Section 3.6. Therefore, in order to solve large instances efficiently, such as those solved by Lee et al. (2010) with up to 200 vessels, a metaheuristic algorithm is necessary. In this study a new Genetic Algorithm was developed, as this kind of metaheuristic produced very good results in other versions of the BAP (see, for example, Nishimura et al., 2001, and many other papers since then). This section describes first the elements of the GA, in which the representation of a solution is a sequence of vessels, then the decoding algorithms developed to produce a feasible solution for the BAP from a vessel sequence, and finally the Local Search procedure.

3.5.1 A Genetic Algorithm

The basic decision when designing a Genetic Algorithm is the way in which a solution is represented. Certain genetic operators (crossover and mutation) are defined according to this representation and applied to the individuals in the population in order to make it evolve and produce high-quality solutions. The main elements of this GA are:

- **Representation**

  Instead of using the diagram representation of a solution, as shown in Figure 3.1, the GA uses a simpler representation in which the chromosome is an ordered list of vessels, as in Figure 3.2a. Therefore its length is the number of vessels. In order to obtain a solution, the chromosome is decoded by using any of the procedures described in the next subsection.



(a) Representation of a solution. (b) Population.

Figure 3.2: Elements of the Genetic Algorithm.

- **Initial population**

  The initial population is composed of N_P individuals, with at least 25 corresponding to the sequences obtained by using the 25 priority rules described in Table 3.1. In the cases in which $N_P > 25$, the additional sequences are randomly generated. The first group of rules corresponds to individual characteristics of the vessels, while a second group combines these features into more complex rules. Previous studies have shown that including good solutions in the initial population produces

good results (see, for instance, Vallada and Ruiz, 2011). A possible disadvantage could have been that this population was too homogeneous for values of N_P close to 25, because that could have provoked a premature convergence. However, the priority rules order the vessels in different ways and produce quite different solutions. The additional randomly generated solutions add even more diversity. In fact, a preliminary analysis revealed that by using this initial population better solutions were obtained than with a completely random initialization.

Table 3.1: Priority rules used for building the initial population.

Name	Sorting criterion	
Estimated arrival time	Non-decreasing	a_i
Handling time	Non-increasing	h_i
Maximum length	Non-increasing	l_i
Minimum length	Non-decreasing	l_i
Waiting cost	Non-increasing	C_i^w
Delay cost	Non-increasing	C_i^d
Departure time	Non-decreasing	s_i
Area	Non-increasing	$AH_i = l_i h_i$
Weighted area 1	Non-increasing	$APD_i = AH_i/C_i^d$
Weighted area 2	Non-increasing	$APM_i = AH_i C_i^d$
Weighted handling time	Non-increasing	$TPP_i = C_i^d h_i$
Slack	Non-decreasing	$G_i = s_i - (a_i + h_i)$
Weighted slack	Non-decreasing	$GW_i = G_i/C_i^d$
Area-slack ratio	Non-increasing	$APG_i = APM_i/G_i$
C1	Non-decreasing	$(a_i + h_i)/s_i$
C2	Non-decreasing	$DR_i = (a_i - h_i)/s_i$
C3	Non-decreasing	$(a_i + h_i + l_i)/(a_i + s_i)$
C4	Non-decreasing	$(h_i + l_i)/(a_i + s_i)$
C5	Non-increasing	$h_i s_i C_i^d$
C6	Non-increasing	GW_i/DR_i
C7	Non-decreasing	GW_i/DR_i
C8	Non-increasing	$C_i^w(h_i + l_i)/(a_i + s_i)$
C9	Non-increasing	$C_i^d(h_i + l_i)/(a_i + s_i)$
C10	Non-increasing	$((a_i - h_i - s_i)/(C_i^d))$ $*(h_i + l_i)/(a_i + s_i)$
C11	Non-increasing	$(a_i h_i)/s_i$

- **Measure of fitness**

  The total assignment cost of all the vessels is taken as a measure of fitness. As the GA is dealing with a minimization problem, the smaller the objective function value, the higher the fitness value.

- **Reproduction**

  A total of $\lceil N_P/2 \rceil$ pairs of individuals are chosen using the roulette method, in which each individual has a probability proportional to its fitness. The operator

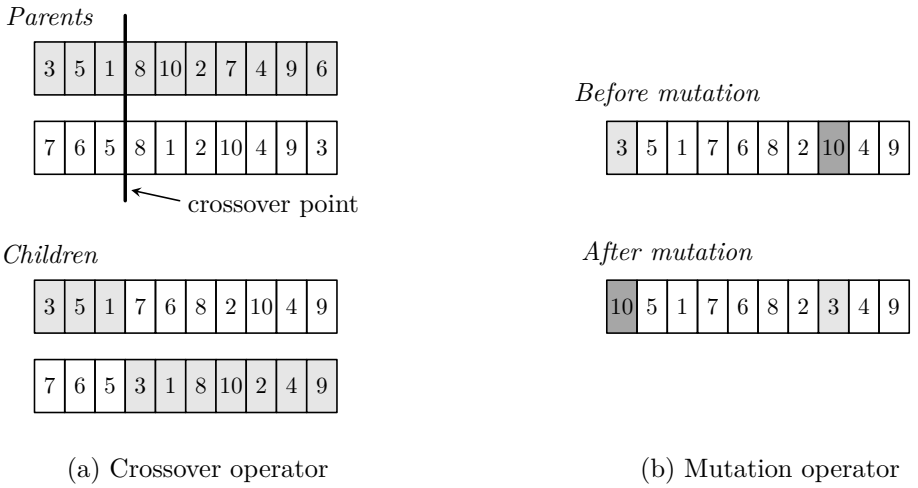
used is the one-point crossover in which the parent sequences are cut at a randomly selected position and each child gets the first subsequence of one of the parents and completes it with the sequence of the other. Thus two children are obtained and each child has a complete ordered sequence of vessels which corresponds to a feasible solution to the problem (see Figure 3.3a).

- **Mutation**

  Each child obtained by the crossover operation has a probability P_m of suffering a mutation, which consists of exchanging the positions of two randomly selected vessels (see Figure 3.3b).

- **New population**

  The individuals in the current generation together with the children obtained through the reproduction and mutation processes are ordered by non-increasing fitness. The N_P individuals with higher fitness will pass to the next generation.



(a) Crossover operator (b) Mutation operator

Figure 3.3: Genetic operators.

3.5.2 Algorithms to decode a vessel sequence into a BAP solution

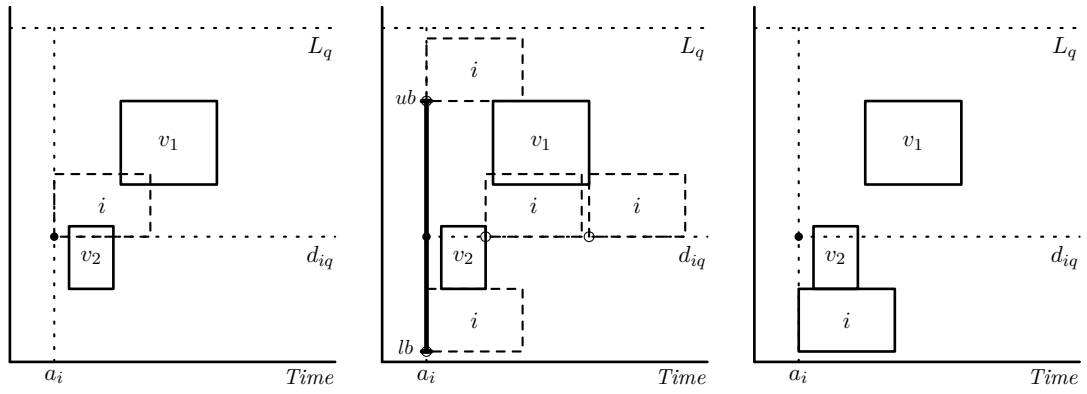
Four constructive algorithms that build a feasible solution from an ordered list of vessels were developed. These algorithms take one vessel at a time and look for its least-cost assignment to a quay, a position, and a berthing time at which it does not overlap with previous vessel assignments. The first two algorithms are heuristics, while the other two are matheuristics based on the integer model. After presenting these algorithms, one is selected to be integrated into the Genetic Algorithm for the rest of the study.

3.5.2.1 Exploratory constructive algorithm

The first algorithm, denoted as the *Exploratory Constructive Algorithm (ECA)*, looks for the best assignment for each vessel by building and exploring a set of candidate assign-

ments, ordered by non-decreasing cost. Given a vessel from the list, the set of candidate assignments for this vessel initially contains the assignment in which the berthing time is its arrival time and its berthing position is its desired position on the quay. This assignment is examined to check whether berthing the vessel at this time, quay, and position overlaps any other previously assigned vessel. If it does, new candidate assignments are identified around this vessel such that the overlap is avoided. In order to prevent redundant tests, the interval of positions that would lead to a new overlap considering the same berthing time is also identified. This process is repeated taking the least-cost candidate from the set of candidates until a feasible assignment is found. The procedure is explained in more detail in Algorithm 1, where K is the set of candidate assignments and F is the set of forbidden intervals. Each candidate is identified by (q, t, p), where q is the quay, t the berthing time, and p the berthing position on quay q. Similarly, each tuple defining a forbidden interval is identified by (q, t, U), where U is the interval of forbidden positions on quay q at time t.

Figure 3.4 shows an example of the process in a terminal with a single quay. A partial solution composed of vessels v_1 and v_2 has already been built and the next vessel in the sequence, vessel i, has to be included. In Figure 3.4a, vessel i is allocated its best assignment, given by its arrival time and its desired position on the quay with the lowest assignment cost. As it overlaps with both v_1 and v_2, another assignment must be found. In Figure 3.4b, a vertical segment has been identified in which the bottom-left corner of vessel i cannot be placed. Using this segment and the positions of v_1 and v_2, some new candidate assignments for vessel i have been identified (in dashed lines). In Figure 3.4c, vessel i is allocated the least-cost feasible assignment. In the general case, if all the candidate assignments in Figure 3.4b had been unfeasible, overlapping other vessels in the partial solution, the process would have continued, identifying new alternatives until a feasible assignment appeared.



(a) Trying to assign vessel i to its best time and position.

(b) Identifying new candidates surrounding v_1 and v_2.

(c) Vessel i is allocated the best feasible assignment.

Figure 3.4: Process of the Exploratory Constructive Algorithm.

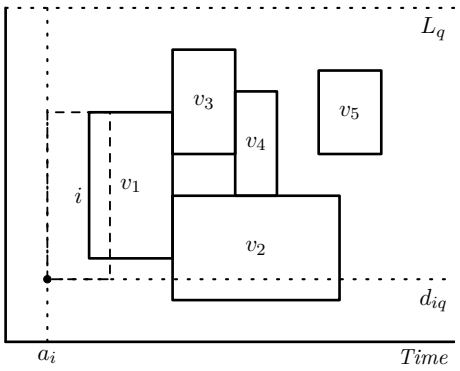
Algorithm 1 Exploratory Constructive Algorithm

Input: list of vessels
Output: solution containing the vessels in the input list

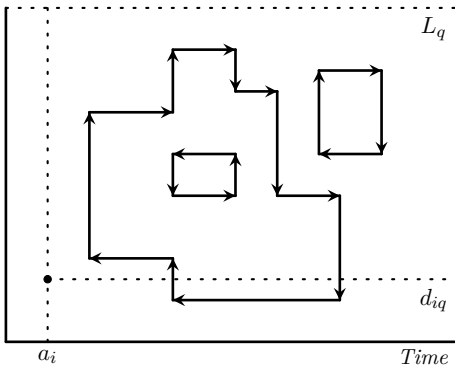
for all vessel i in the list, according to its order **do**
  $assigned \leftarrow$ **false**
  $K \leftarrow \emptyset$
  $F \leftarrow \emptyset$
  for all $q \in Q$ **do**
    if $l_i \leq L_q$ **then**
      if $(d_{iq} + l_i) > L_q$ **then**
        Insert $(q, a_i, L_q - l_i)$ in K
      else
        Insert (q, a_i, d_{iq}) in K
      end if
    end if
  end for
  while $assigned =$ **false** and $K \neq \emptyset$ **do**
    Extract the least-cost candidate (q, t, p) from K
    if $\nexists (q, t, U) \in F : p \in U$ **then**
      if vessel i would not overlap with a previously assigned vessel if assigned to time t and position p at quay q **then**
        Assign position p at quay q at time t to vessel i
        $assigned \leftarrow$ **true**
      else
        Identify the vessels corresponding to the overlaps found
        Compute the lower bound (lb) and the upper bound (ub) of the interval of forbidden positions for i in t according to the overlaps found
        Insert (q, t, lb) in K if it fits in quay q
        Insert (q, t, ub) in K if it fits in quay q
        Insert $(q, t,]lb, ub[)$ in F
        for all vessel v which overlaps with i **do**
          Insert $(q, t_v + h_v, p)$ in K if it fits in quay q
        end for
      end if
    end if
  end while
end for

3.5.2.2 Analytic constructive algorithm

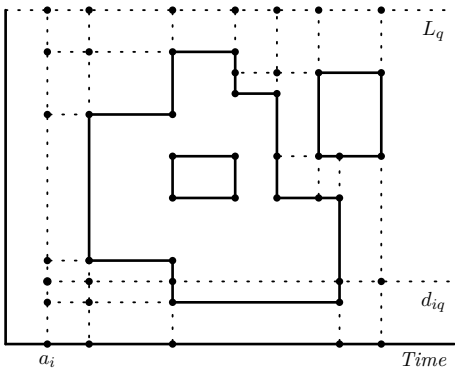
The second algorithm, called the *Analytic Constructive Algorithm (ACA)*, performs an analysis of the preexisting plan for each quay. This analysis is based on a geometric representation of the current partial assignment at each quay, which is built incrementally each time a vessel is assigned to the quay. The process is described in Algorithm 2 and in Figure 3.5. In Figure 3.5a, vessel i has been taken from the list, but when assigned to its best position it overlaps with vessels already assigned. Then the analytic process starts. In a first phase (Figure 3.5b) the segments defining the geometric representation are identified and their vertices are given a label indicating whether there is free space in any of the quadrants around them, as in Lee et al. (2010). The list of potential relevant points for assigning the next vessel includes these vertices and also some other points obtained by using the information about the vessel i being assigned. These points are identified by drawing vertical lines that extend the vertical segments of the representation, including a vertical line at a_i; and also horizontal lines extending the horizontal segments, including a line at d_{iq}, as in Figure 3.5c. The intersections of these lines define new points, which are labelled in a similar way to the vertices in the diagram representation. For each point identified in this process, the candidate assignments indicated by its label are obtained and put into a candidate set, ordered by their cost. The candidates are examined one at a time until a feasible assignment is found (Figure 3.5d).



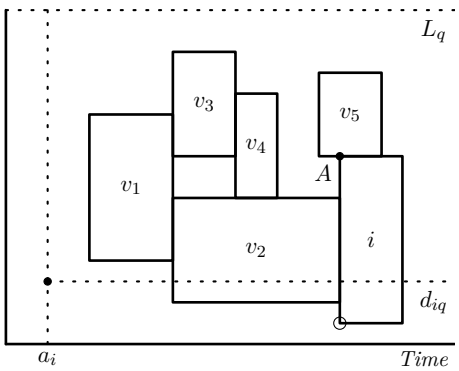
(a) Trying to assign vessel i at (q, a_i, d_{iq}).



(b) Geometric representation.



(c) Looking for relevant points.



(d) Assignment of vessel i to a position obtained from point A.

Figure 3.5: Process of the Analytic Constructive Algorithm.

Algorithm 2 Analytic Constructive Algorithm

Input: list of vessels, current geometrical representation of each quay diagram
Output: a solution containing the vessels in the input list

for all vessel i in the list, according to its order **do**
  $K \leftarrow \emptyset$
  $Cost_{max} \leftarrow \infty$
  for all $q \in Q$ **do**
    if $K \neq \emptyset$ **then**
      $Cost_{max} \leftarrow \max_{k \in K}\{\text{cost for vessel } i \text{ in } k\}$
    end if
    $Cost_{q_{min}} \leftarrow$ cost for vessel i at its ideal assignment on quay q
    ▷ A quay is analysed only if it can yield better candidate assignments than the quays already analysed
    if $Cost_{q_{min}} < Cost_{max}$ **then**
      Add-candidates-quay(q, i, K)
    end if
  end for
  $assigned \leftarrow$ **false**
  while $assigned =$ **false** and $K \neq \emptyset$ **do**
    Extract the least-cost candidate (q, t, p) from K
    if vessel i would not overlap with any other vessel if assigned to (q, t, p) **then**
      Assign position p on quay q at time t to vessel i
      Update incrementally the geometric representation of quay q
      $assigned \leftarrow$ **true**
    end if
  end while
end for

Algorithm 3 Subroutine: Add-candidates-quay

Input: current representation of quay q, vessel i, ordered set of candidates K
Output: the set K containing the relevant candidates for quay q

Get the end points of the segments in the representation of quay q and compute their tag
Get the intersection points between the vertical line a_i and each horizontal segment in the representation
Get the intersection points between the horizontal line d_{iq} and each vertical segment in the representation located to the right of the vertical line a_i
Get additional intersection points related to virtual segments resulting from inspecting the tags of the previous points
for all previous discovered point **do**
  Insert the candidates defined by their tag in K, ordered by their cost
end for

3.5.2.3 Matheuristic constructive algorithm M1

As in the previous heuristics, this algorithm constructs a solution assigning the vessels in the input list one by one according to their order.

Let us suppose that in the construction process some vessels, $i \in \{1, \ldots, j-1\}$, have been assigned. In order to determine the best assignment for the next vessel j a reduced version of the integer linear model described above can be used. Let us suppose that the berthing times and positions of the previously assigned vessels, given by variables p_i, t_i, $m_{iq}, \forall i \in \{1, \ldots, j-1\}, \forall q \in Q$, are fixed. Hence, their relative positions, given by variables σ_{ik}, $\delta_{ik}, \forall i, k \in \{1, \ldots, j-1\}$, are also fixed. The problem to determine the least-cost position of vessel j, non-overlapping with the previously assigned vessels, is:

$$Min \;\; C_j^w(t_j - a_j) + C_j^d u_j + \sum_{q \in Q} (C_{jq}^a m_{jq} + C_{jq}^p e_j) \tag{3.13}$$

s. t.

$$t_j \geq a_j \tag{3.14}$$

$$\sum_{q \in Q} m_{jq} = 1 \tag{3.15}$$

$$p_j + l_j \leq \sum_{q \in Q} m_{jq} L_q \tag{3.16}$$

$$u_j \geq t_j + h_j - s_j \tag{3.17}$$

$$e_j \geq p_j - \sum_{q \in Q} d_{jq} m_{jq} \tag{3.18}$$

$$e_j \geq \sum_{q \in Q} d_{jq} m_{jq} - p_j \tag{3.19}$$

$$t_j - (t_i + h_i) - (\sigma_{ij} - 1)T \geq 0, \qquad \forall i \in \{1, \ldots, j-1\} \tag{3.20}$$

$$t_i - (t_j + h_j) - (\sigma_{ji} - 1)T \geq 0, \qquad \forall i \in \{1, \ldots, j-1\} \tag{3.21}$$

$$p_j - (p_i + l_i) - (\delta_{ij} - 1)L_{max} \geq 0, \qquad \forall i \in \{1, \ldots, j-1\} \tag{3.22}$$

$$p_i - (p_j + l_j) - (\delta_{ji} - 1)L_{max} \geq 0, \qquad \forall i \in \{1, \ldots, j-1\} \tag{3.23}$$

$$\sigma_{ij} + \sigma_{ji} + \delta_{ij} + \delta_{ji} \geq m_{iq} + m_{jq} - 1, \qquad \forall q \in Q, \forall i \in \{1, \ldots, j-1\} \tag{3.24}$$

$$\sigma_{ij}, \sigma_{ji}, \delta_{ij}, \delta_{ji} \in \{0, 1\}, \qquad \forall i \in \{1, \ldots, j-1\} \tag{3.25}$$

$$t_j, p_j, u_j, e_j \geq 0 \tag{3.26}$$

This problem is very easy to solve because it involves only the assignment variables of vessel j and the binary variables indicating whether j is above, below, to the right, or to the left of vessels $i \in \{1, \ldots, j-1\}$, whose positions are fixed. The algorithm solves this problem for each vessel in the input list, one by one. It would therefore produce the same solution as the constructive algorithms *ECA* and *ACA* if the way of breaking the ties between solutions with the same cost were the same in both cases. However, this algorithm leads us to a more interesting matheuristic.

3.5.2.4 Matheuristic constructive algorithm M2

Let us suppose now that the assignments of vessels $i \in \{1 \dots, j-1\}$ to quays and their relative positions are fixed as in the previous case, but the berthing times and positions of these vessels are free. The problem of determining the least-cost position of vessel j, non-overlapping with the previously assigned vessels, would now be:

$$Min \sum_{i=1}^{j}(C_i^w(t_i - a_i) + C_i^d u_i) + \sum_{q \in Q}\sum_{i=1}^{j}(C_{iq}^a m_{iq} + C_{iq}^p e_i) \tag{3.27}$$

s. t.

$$t_i \geq a_i, \qquad \forall i \in \{1, \dots, j\} \tag{3.28}$$

$$\sum_{q \in Q} m_{iq} = 1, \qquad \forall i \in \{1, \dots, j\} \tag{3.29}$$

$$p_i + l_i \leq \sum_{q \in Q} m_{iq} L_q, \qquad \forall i \in \{1, \dots, j\} \tag{3.30}$$

$$u_i \geq t_i + h_i - s_i, \qquad \forall i \in \{1, \dots, j\} \tag{3.31}$$

$$e_i \geq p_i - \sum_{q \in Q} d_{iq} m_{iq}, \qquad \forall i \in \{1, \dots, j\} \tag{3.32}$$

$$e_i \geq \sum_{q \in Q} d_{iq} m_{iq} - p_i, \qquad \forall i \in \{1, \dots, j\} \tag{3.33}$$

$$t_j - (t_i + h_i) - (\sigma_{ij} - 1)T \geq 0, \qquad \forall i \in \{1, \dots, j-1\} \tag{3.34}$$

$$t_i - (t_j + h_j) - (\sigma_{ji} - 1)T \geq 0, \qquad \forall i \in \{1, \dots, j-1\} \tag{3.35}$$

$$p_j - (p_i + l_i) - (\delta_{ij} - 1)L_{max} \geq 0, \qquad \forall i \in \{1, \dots, j-1\} \tag{3.36}$$

$$p_i - (p_j + l_j) - (\delta_{ji} - 1)L_{max} \geq 0, \qquad \forall i \in \{1, \dots, j-1\} \tag{3.37}$$

$$\sigma_{ij} + \sigma_{ji} + \delta_{ij} + \delta_{ji} \geq m_{iq} + m_{jq} - 1, \qquad \forall q \in Q, \forall i \in \{1, \dots, j-1\} \tag{3.38}$$

$$\sigma_{ij}, \sigma_{ji}, \delta_{ij}, \delta_{ji} \in \{0, 1\}, \qquad \forall i \in \{1, \dots, j-1\} \tag{3.39}$$

$$t_i, p_i, u_i, e_i \geq 0, \qquad \forall i \in \{1, \dots, j\} \tag{3.40}$$

This problem is more difficult than the previous one. It has the same set of binary variables, determining the relative position of vessel j with respect to the other vessels $i \in \{1, \dots, j-1\}$, and some more real variables, but now the absolute positions of these vessels are free and therefore the problem has more feasible solutions. This flexibility may lead to better solutions, improving on the results of previous constructive algorithms.

3.5.2.5 Selection of the constructive algorithm

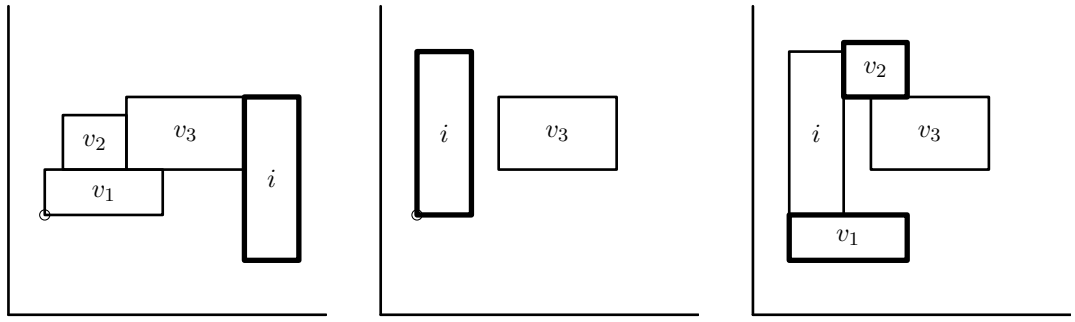
According to preliminary experiments, the heuristic algorithms were much faster than the matheuristics, as was to be expected. $M2$ frequently obtained better results than the other algorithms on random lists of vessels, although sometimes it performed worse,

which is reasonable, since the assignment of better locations for the vessels in the list, one by one, does not guarantee a better complete solution in the end. $M2$ was also the slowest algorithm by far when it was used in the Genetic Algorithm. For example, it completed only one generation on instances with 40 vessels considering $N_P = 30$, $P_m = 0.5$, and a stopping criterion of 100 seconds of computation time, while the heuristics were able to complete more than 4000 generations in the same time. Given that the effort of searching the solution space is performed by the Genetic Algorithm through many generations, the matheuristics were discarded as candidate constructive algorithms for the GA.

Comparing the two heuristic algorithms, the *ECA* was faster than the *ACA* in the test instances used, with up to 200 vessels, although its computation time increased more rapidly with the number of vessels. This would make the *ACA* more attractive for dealing with larger instances. Another point worth noting is that the *ECA* can be used even when the constructive process starts from a partial solution, while in such cases the *ACA* would require building the geometric representation of each quay from scratch, in contrast to its normal process, in which the representation is built incrementally as the vessels are assigned. For these two reasons, the *Exploratory Constructive Algorithm* was selected to be the constructive heuristic of the Genetic Algorithm for the rest of the study.

3.5.3 Improving the best solution obtained through a Local Search procedure

The Genetic Algorithm works on ordered lists of vessels. As a last step in the algorithm, a Local Search (LS) phase was added to explore solutions that are neighbours of the best solution known but do not correspond to orderings of the list of vessels. The move defining the neighbourhood of the solution consists of choosing a vessel i with positive cost, that is, a vessel that is not assigned to its best quay, time, and position, and moving it to the position of another vessel k, on any quay, if that vessel i fits into this position and its assignment cost is decreased. The new position of vessel i will produce an overlap with vessel k and possibly with other vessels. All these vessels are removed from the solution and assigned again by the *ECA* algorithm. In order to reduce the search, not all the vessels are considered, but only a given percentage, $f\%$ of the vessels with the highest costs. These vessels are taken one at a time and the corresponding move is studied. If one of them produces an improved solution, the best solution known is updated and the search starts again from that solution. If all the selected vessels are studied and no improvement is found, the search stops. Figure 3.6 and Algorithm 4 show the procedure in more detail. Note that in Figure 3.6, if the desired position of vessels v_1 and i was the bottom left corner of the position of vessel v_1 in the initial solution and the desired position of vessels v_2 and v_3 was the bottom left corner of the position of vessel v_2, the solution produced after the move could not be obtained with any ordering of vessels.



(a) Vessel i is chosen to be moved.

(b) Vessels v_1 and v_2 are removed.

(c) Vessels v_1 and v_2 are reassigned.

Figure 3.6: Local search. Process of reassigning a high-cost vessel.

Algorithm 4 Local Search

Input: a feasible solution

Output: a solution whose cost is not greater than that of the input solution

1: **repeat**
2: $VS \leftarrow$ {all vessels in the solution}
3: $VC \leftarrow \{i \in VS : Cost(i) > CostIdeal(i)\}$
4: $VMC \leftarrow \{f\%$ most-valued $(Cost(i) - CostIdeal(i))$ vessels in $VC\}$
5: **for all** vmc in VMC **do**
6: **for all** v in VS **do**
7: **if** $(v \neq vmc)$ and (q_v, t_v, p_v) is a feasible assignment for vmc **then**
8: $PotentialCost_{vmc,v} \leftarrow CostInAssignment(vmc, q_v, t_v, p_v)$
9: **if** $PotentialCost_{vmc,v} < Cost(vmc)$ **then**
10: Store the solution resulting from assigning vessel vmc to (q_v, t_v, p_v) and reassigning, by means of the *ECA*, the vessels which overlap with vmc in that assignment (ordered by non-decreasing arrival time). See Figure 3.6.
11: **end if**
12: **end if**
13: **end for**
14: **end for**
15: If the best solution found improves on the best known solution, it is updated and the search starts again.
16: **until** not improved

3.6 Computational experiments

In order to evaluate the performance of the algorithms developed in previous sections, an extensive computational study was designed. First, the behaviour and the limits of using the proposed integer model for solving the problem exactly were evaluated. The

instances kindly provided by Park and Kim (2002) were used, modified to accommodate several quays, as well as the instances by Lee et al. (2010) and by Cordeau et al. (2005). Then the modified set of Park and Kim (2002) was used to adjust the population size of the Genetic Algorithm and to compare its results with those obtained by the 25 priority rules. The Genetic Algorithm was also compared with the metaheuristics developed by Lee et al. (2010) and by Cordeau et al. (2005). Finally, a new set of instances was generated controlling various characteristics. Studying the results obtained by the model and the GA on this set makes it possible to obtain some insight into the more influential factors associated with the problem.

In all the experiments conducted applying the Genetic Algorithm the Exploratory Constructive Algorithm and a mutation probability $P_m = 0.5$ were used. Each experiment was repeated three times considering different random seeds. Then, for each instance, the average cost of the best solution obtained and the average running time over the three repetitions were calculated.

The algorithms were coded in C++11 and run on a PC with an Intel Core i7-2600 at 3.4 GHz with 31.4 GiB of RAM. The operating system used was Ubuntu 14.04 GNU/Linux 3.13 and the compiler was GCC–G++ 6.2. The compilation was performed specifying the special parameters *–O3 –march=corei7* in order to generate an executable file that makes the most of the processor. OpenMP was used to evaluate as many individuals as possible in parallel threads. This did not affect the process in any way apart from the running time, which was thereby reduced. The model was implemented in C++11 and the solver CPLEX 12.6, and was run on the same computer.

3.6.1 Results of the integer linear model

Park and Kim (2002) generated a set of test instances taking the Pusan Eastern Container Terminal in Pusan, South Korea, as a reference. The quay considered is 1200 metres long. First they generated 25 instances ranging from 13 to 20 vessels, referred to as real-world instances, and then 50 instances, 10 for each size: 20, 25, 30, 35, and 40 vessels. Estimated arrival times, handling times, vessel lengths, and costs were drawn from uniform distributions in an attempt to capture the values of real-world problems. The original data were modified in order to extend the model to the case of multiple quays. So besides the 75 instances with one quay, two more sets were generated with two and three quays, with the same vessels, splitting the length of the original quay into two or three parts at random, but ensuring that the shortest quay was able to accommodate at least the shortest vessel.

The results are shown in Table 3.2, which has three parts, corresponding to the number of quays. The instances were divided into classes. The first corresponds to the instances labelled by Park and Kim (2002) as real-world instances, with the number of vessels between 13 and 20. The other classes correspond to the generated instances with 20 to 40 vessels. For each class, the table shows the mean, median, and maximum time in seconds required to solve the instances to optimality. It can be seen on the last line of the first section that there is one instance with 40 vessels which is very hard to solve. Apart from this exceptional case, the number of quays does not seem to have a

significant effect on the computing times. On the contrary, the required times increase very sharply with the number of vessels, but instances of up to 40 vessels can be solved in reasonable times even in the worst cases.

Table 3.2: Time required by the MILP run on CPLEX to solve the instances of Park and Kim optimally.

Quays	Class	Mean (s)	Median (s)	Maximum (s)
1 quay	Real-world	0.03	0.03	0.04
	20	0.05	0.04	0.08
	25	0.10	0.07	0.26
	30	0.13	0.12	0.21
	35	1.52	0.55	6.58
	40	487.36	3.90	4513.68
2 quays	Real-world	0.04	0.04	0.07
	20	0.10	0.08	0.18
	25	0.90	0.13	3.65
	30	0.66	0.27	2.09
	35	4.83	4.34	11.42
	40	32.24	19.26	110.43
3 quays	Real-world	0.04	0.04	0.08
	20	0.07	0.6	0.09
	25	0.16	0.16	0.25
	30	0.22	0.15	0.48
	35	1.18	0.75	3.23
	40	10.62	6.17	43.78

In order to extend the study of the integer model, the instances generated by Lee et al. (2010) were also considered. They produced large instances with 40, 80, 120, 160, and 200 vessels for the problem with just one quay. The 40 vessel instances were addressed first, with a time limit of 3600 seconds per instance, and none of them was solved to optimality within this time limit. Moreover, the average gap was 78%. These instances are so hard for the integer linear model because the arrival times of all the vessels are concentrated at the beginning of the planning period, thereby producing a congestion which appears to be very difficult to manage. This congestion factor is analysed in more detail in Section 3.6.5.

The instances generated by Cordeau et al. (2005), with one quay and 60 vessels, were also addressed with a time limit of 3600 second per instance. Only one of the 30 instances was solved to optimality within the time limit. The average gap was 7.1%. Compared with the Lee et al. (2010) instances, these seem to be easier, but they are larger, and 60 vessels appear to be too many for the exact approach. In these two sets of instances, solving the integer model does not provide good solutions in reasonable computing times and therefore an efficient metaheuristic algorithm is needed for solving large problems.

3.6.2 Results of the Genetic Algorithm

The Genetic Algorithm was also run on the set of instances provided by Park and Kim (2002). First, in order to adjust the population size (N_P), for each $N_P \in \{30, 50, 100\}$ a stopping criterion of 100 generations without improving the best solution attained during the process was considered. The results showed that the average cost obtained with $N_P = 100$ was 1% lower than considering $N_P = 30$, and 0.9% lower than considering $N_P = 50$, without a noticeable increase in computation time, which can be explained by the use of parallelism and the apparently low difficulty of many of the instances. In the light of the results, it was decided to set $N_P = 100$ for the rest of the experiments.

The same set was also used to compare the Genetic Algorithm alone (GA) with the Genetic Algorithm plus a Local Search phase applied on the best solution obtained by the GA (GA+LS), and with the best result obtained when applying the 25 priority rules (Best Rule). The parameter f of the LS was set to 30%, according to the results obtained in preliminary experiments. The stopping criterion of the Genetic Algorithm was again completing 100 consecutive generations without improving on the best solution obtained. The results are summarized in Table 3.3. For each group of 75 instances with 1, 2 and 3 quays, the first row shows the average percentage deviation from the optimum, calculated as $100 \cdot (result - optim)/optim$, where $result$ is the average cost of the best solutions obtained in the three repetitions, and $optim$ is the cost of the optimal solution. The second row shows the number of instances optimally solved in all the repetitions.

The results show that the GA performs very well on this set of instances, obtaining the optimal solution in all but 16 of the 225 test instances. One could also conclude that there is no need to add the Local Search phase to the GA, because it does not produce any improvement. However, as it increases the computation time only by 1% on average, in has been retained in the final version of the GA, because it could be useful when solving larger and more difficult instances, tested in the next sections.

The last column of the table shows the results when each instance is solved with the 25 priority rules and the best solution is kept. The number of optimal solutions is relatively high, though it decreases with the number of quays; while the average distance to the optimal solution is high for one-quay instances and increases very sharply with the number of quays. This indicates that this set contains many instances that are easy to solve with any algorithm, but also hard instances for which only complex metaheuristics, like the GA proposed here, are able to obtain optimal or near-optimal solutions, whereas simple rules or a combination of them fail by a large margin.

Table 3.3: Results of the Genetic Algorithm in the instances of Park and Kim.

		Algorithm		
		GA	GA+LS	Best Rule
1 quay	Average deviation	0.12	0.12	3.29
	Optimal solutions	72	72	62
2 quays	Average deviation	0.53	0.5	12.73
	Optimal solutions	68	68	56
3 quays	Average deviation	0.19	0.19	15.08
	Optimal solutions	69	69	47

3.6.3 The Lee et al. instances

Lee et al. (2010) considered the same problem studied here, but with only one quay and with the single objective of minimizing the total servicing time of the vessels, defined, for each vessel, as the elapsed time from its arrival to the end of its handling. They developed an integer linear model and two GRASP algorithms.

In a first computational study on 30 small instances with 5 and 10 vessels, they compared the results obtained by solving their integer linear model with CPLEX and using GRASP1 and GRASP2. The Genetic Algorithm developed here was also run on these instances, considering the objective defined by them and taking completion of 100 generations without improvement as the stopping criterion. The results are shown in Table 3.4, taken from Lee et al. (2010), to which the last column has been added with the results of the Genetic Algorithm. For each algorithm, the table shows the average percentage deviation from the optimum (Gap) and the average computation time in seconds. It can be seen that solving their integer model is very costly, even for problems with 10 vessels. For these small instances the GA obtains solutions very close to optimal, not as close as those of GRASP2 for 5-vessel instances, but clearly better for 10-vessel instances, in very short computation times.

Table 3.4: Small instances by Lee et al.

Vessels	CPLEX	GRASP1		GRASP2		GA	
	T	T	Gap	T	Gap	T	Gap
5	0.39	2.29	1.62	2.73	0.01	0.04	0.15
10	5977	11.90	7.89	16.59	3.70	0.13	0.54

In a second study, Lee et al. (2010) tested their algorithms on larger instances with 40, 80, 120, 160, and 200 vessels arriving within 20 time units; 30 instances of each size. Table 3.5, also built by using the data kindly provided by the authors, uses the results of the previously published Stochastic Beam Search (SBS) algorithm by Wang and Lim (2007) as an initial reference, and shows the average computation time in seconds and the average cost of the best solutions obtained by the GRASP1 and the GRASP2 proposed by Lee et al. (2010). The right-hand side of the table shows the results obtained by two versions of the algorithm developed here. First, the GA was run with a fixed time limit as similar as possible to that used by the GRASP2. As the computer used, according to `http://www.cpubenchmark.net`, was approximately 23 times faster, the total time limit was set equal to the average running time in seconds reported by Lee et al. (2010) on each group of instances divided by 23. The 95% of this time was allocated to the GA, while the remaining 5% was allocated to the LS. Finally, the last two columns show the average computation time and the average cost of the best solutions obtained by the algorithm using the stopping criterion of 100 generations without improvement. As can be seen, the results of the controlled-time version of the GA were 17% better on average than the GRASP2, so it seems to perform more efficiently while at the same time obtaining better solutions. The improvement achieved by the LS was 0.14% on average,

although it exceeded 1% in some instances. Considering the stopping criterion of 100 generations without improvement and no time limit for the LS, the GA required more time and achieved solutions which were on average 19.5% lower than those obtained by the GRASP2. The LS spent less than 1% of the computation time on average and its mean improvement was not significant, although in some instances it was capable of improving on the solution obtained by the GA by up to 0.44%.

Table 3.5: Large instances by Lee et al.

Vessels	SBS		GRASP1		GRASP2		GA (t.l.)		GA	
	T	Cost	T	Cost	T	Cost	T	Cost	T	Cost
40	188	15094	15	14958	37	14766	2	12262	7	11905
80	790	60132	112	58858	271	56564	11	46511	71	45394
120	911	134300	413	129852	882	125023	38	103592	349	100465
160	1703	237358	1103	229432	2137	218520	92	182238	1069	176415
200	2809	374808	1763	364986	4223	344793	184	288071	2458	277883

3.6.4 The Cordeau et al. instances

Cordeau et al. (2005) studied the discrete problem first, considering only costs related to the total time the vessels are moored at the dock, from their arrival to the end of their servicing. They proposed an integer linear model and a Tabu Search algorithm, which was later adapted to the continuous case. For this continuous case they generated 30 instances with only 1 quay and 60 vessels arriving within one week, taking the real problems encountered at the Gioia Tauro terminal in Italy as a reference. For these instances, the authors presented the results of their Tabu Search (TS) algorithm and compared them with an FCFS-G algorithm, which orders the vessels by non-decreasing arrival time and assigns each vessel to the position and berthing time that minimize the time needed for it to be serviced. According to the authors, the FCFS-G was similar to the procedure used at the terminal.

Table 3.6 shows the results of the FCFS-G and the Tabu Search algorithm reported by Cordeau et al. (2005), as well as the average results obtained in the three repetitions of the experiment by the Genetic Algorithm developed here considering the stopping criterion of 100 generations without improvement. For all the instances, the GA improved on the results of both the FCFS-G and the TS, achieving an average improvement of 18.4% over the FCFS-G and 10.9% over the TS. The average running time was 3.7 seconds and the LS did not achieve any improvement.

Table 3.6: Average costs of the best solutions obtained by each algorithm for the instances of Cordeau et al.

Inst.	FCFS-G	TS	GA	Inst.	FCFS-G	TS	GA
1	1899	1706	1539	16	1854	1715	1417
2	1417	1355	1277	17	1388	1322	1293
3	1349	1286	1156	18	1923	1594	1396
4	1548	1440	1339	19	1829	1673	1428
5	1449	1352	1228	20	1615	1450	1354
6	1747	1565	1249	21	1640	1565	1371
7	1482	1389	1289	22	1747	1618	1355
8	1616	1519	1359	23	1770	1539	1337
9	1873	1713	1517	24	1625	1425	1289
10	1611	1411	1206	25	1845	1590	1454
11	1851	1696	1418	26	1707	1567	1385
12	1814	1629	1425	27	1588	1458	1271
13	1575	1519	1385	28	1669	1550	1413
14	1435	1369	1238	29	1512	1415	1291
15	1609	1455	1318	30	1797	1621	1464

3.6.5 New set of instances

All the instances for the continuous BAP published in the literature consider just one quay. In order to study the problem with multiple quays, an instance generator which accepts all the relevant characteristics of the problem as configurable parameters was developed. It was used to study the effect of the key factors on the complexity of the problem and the performance of the Genetic Algorithm.

Considering a length unit of 10 metres and a time unit of 1 hour, certain parameters were fixed:

- Maximum length of a quay (L_{max}): 150
- Minimum length of a quay (L_{min}): 10
- Maximum waiting cost per unit time (C^w): 5
- Maximum delay cost per unit time (C^d): 10
- Maximum assignment cost of vessel to quay (C^a): 10
- Maximum cost of deviation from ideal position on quay per unit (C^p): 3
- Slack for departure time ($s - a - h$): 5

For each instance, the length of each quay was then randomly drawn from U[10, 150] and the cost coefficients for each vessel and quay were randomly taken in the

interval between 0 and the maximum cost. For the other parameters, several levels were considered:

- Number of quays: 1, 3, 5
- Number of vessels: 20, 30, 40
- Length of vessels (minimum, maximum): (1,10), (1,25), (1,50)
- Handling times (minimum, maximum): (1,10), (1,25), (25,50)
- Maximum arrival time: 20, 50, 100

Using the generator, 5 random instances were created for each combination of the parameter levels, which resulted in a total of 1215 instances. This made it possible to study different configurations for each number of quays and each number of vessels, involving the type of vessels (all small; of varying sizes), the handling times (all short; of different lengths; all large), and the arrival time (all concentrated at the beginning of the planning interval; all scattered over time).

The first part of the study focused on the effect of these factors on the ability of the integer model to obtain optimal solutions for each type of problem. Each of the 1215 instances was solved using CPLEX 12.6, considering a time limit of 600 seconds. Table 3.7 shows the number of instances optimally solved for each number of quays and each number of vessels. We can observe first that these instances are much harder to solve than those of Park and Kim (2002), in which all but one of the instances were solved to optimality in less than 600 seconds. From the instances generated here, only 747 (61%) were solved within the time limit. We can also see that, as expected, the number of vessels has a strong influence on the results. For 20, 30, and 40 vessels, the percentage of problems optimally solved was 84%, 59%, and 45% respectively. Also, and more surprisingly, the number of quays has an effect: the more quays there are, the easier it is to solve (40%, 65%, and 80% for 1, 3, 5 quays respectively). This effect was not observed in previous tests. The main difference may lie in the fact that here assignment costs of vessels to quays and deviation costs from the ideal positions on the quays were added. This seems to introduce some structure into the problem which is used by CPLEX to obtain an optimal solution faster when several quays are involved.

Table 3.7: Number of instances optimally solved by quays and vessels.

Quay	Vessels			Total
	20	30	40	
1	85	47	30	**162**
3	113	86	63	**262**
5	127	107	89	**323**
Total	**325**	**240**	**182**	**747**

Table 3.8 shows the effect of the length and the handling times of the vessels. Both factors have a clear influence on the results. Problems with large vessels and long handling times are very hard to solve to optimality.

Table 3.8: Number of instances optimally solved by lengths and handling times.

Handling time	Length			Total
	(1,10)	(1,25)	(1,50)	
(1,10)	129	116	88	**333**
(1,25)	118	89	61	**268**
(25,50)	94	41	11	**146**
Total	**341**	**246**	**160**	**747**

Finally, Table 3.9 shows the effect of the distribution of the arrival times, which could be called the congestion factor. If all the vessels arrive at the same time, the problem is much harder than if the arrivals are scattered over the time horizon.

Table 3.9: Number of instances optimally solved by quays, vessels, and arrival times.

Quays	Vessels	Arrival time			Total
		(1,20)	(1,50)	(1,100)	
1	20	17	33	35	**85**
	30	7	15	25	**47**
	40	3	10	17	**30**
	Total	**27**	**58**	**77**	**162**
3	20	35	37	41	**113**
	30	15	33	38	**86**
	40	11	20	32	**63**
	Total	**61**	**90**	**111**	**262**
5	20	40	42	45	**127**
	30	29	37	41	**107**
	40	22	29	38	**89**
	Total	**91**	**108**	**124**	**323**
Total		**179**	**256**	**312**	**747**

In the second part of the study, each of the 1215 instances was solved using the Genetic Algorithm with a stopping criterion of 100 seconds as the running time limit and no time limit for the LS. For each instance, the percentage deviation from the solution obtained by the MILP was calculated as $100 \cdot (result_{GA} - result_{MILP})/result_{MILP}$. Table 3.10 shows the average deviations classified by quays and vessels, separating into

two columns the cases in which an optimal solution was found by the MILP from those in which it only returned a feasible solution.

Looking at the column of the instances for which the MILP obtained an optimal solution, the solutions obtained by the GA are on average 1.2% higher, getting nearer as the number of quays increases. For the instances in which the MILP did not obtain an optimal solution within the time limit, the average results clearly favour the GA, which obtained much better results in large instances. For the 468 instances in which the MILP did not get the optimal solution in 600 seconds, the GA obtained a solution better than the MILP in 310: 156 one-quay instances, 97 three-quay instances, and 57 five-quay instances. The time spent by the LS was less than 1 second in all the instances and, even though the mean improvement was not significant, in some instances the improvement achieved exceeded 5%.

Table 3.10: Average percentage deviation of the results obtained by the GA from the results of the MILP.

Quays	Vessels	Optimally solved	Non-optimally solved	Mean
1	20	3.83	3.31	3.64
	30	3.05	-2.87	-0.81
	40	1.51	-10.84	-8.10
	Mean	**2.80**	**-3.46**	**-1.75**
3	20	1.29	6.79	2.18
	30	0.85	-3.93	-0.89
	40	0.33	-12.93	-6.74
	Mean	**0.82**	**-3.36**	**-1.82**
5	20	0.77	10.53	1.35
	30	0.26	-4.24	-0.67
	40	0.23	-12.45	-4.09
	Mean	**0.42**	**-2.05**	**-1.14**
Mean		**1.20**	**-6.00**	**-1.57**

3.7 Concluding remarks

In this chapter the continuous Berth Allocation Problem in terminals with multiples quays has been addressed for the first time at the operational level. A terminal with several quays is a problem for the continuous version, because the terminal is considered as one straight line in which vessels can be berthed according to their length and the positions of the other vessels. In a terminal with multiple quays, in addition to the berthing time and position on the quay, each vessel has to be assigned to one of the quays. Besides the usual costs of waiting to be berthed and of delay with respect to

the ideal departure time, an assignment cost of vessel to quay (which can be used to forbid some vessel-quay combinations) has been included, as well as a deviation cost with respect to the ideal position of the vessel on the quay to which it is actually assigned.

First an integer linear model has been proposed for the problem, extending previous models for the one-quay case. The model can be used to obtain optimal solutions in some instances with up to 40 vessels arriving within a time horizon of one week in reasonable computation times, but other classes of instances have been identified for which obtaining the optimal solution is very hard. In order to tackle these and other larger instances with up to 200 vessels a Genetic Algorithm based on sequences of vessels has been developed. The initial population is built by applying a large set of priority rules which produce good and diverse solutions. For decoding vessel sequences into BAP solutions, two fast and efficient constructive heuristics and two matheuristics have been proposed. Moreover, a Local Search procedure aimed at refining the solutions produced by the Genetic Algorithm has been developed.

Given that there are no similar studies for multiple quays, in order to assess the performance of the Genetic Algorithm it has been applied to three well-known sets of benchmark instances for the case of one quay, each containing up to 200 vessels arriving within a time horizon of one week. The computational study shows that this algorithm outperforms the best results published in the literature for similar continuous dynamic Berth Allocation Problems. An additional experiment conducted on a new set of instances revealed that the complexity of the problem increases with increasing number of vessels, vessel length, vessel handling time, or congestion factor, while it decreases with increasing number of quays. Furthermore, in the same set of instances the Genetic Algorithm proved to perform better than the model considering reasonable computation time limits.

The problem studied in this chapter was limited to the description and assumptions set out in Section 3.3. Here the BAP has been addressed assuming fixed handling times, without considering how it is related to the Quay Crane Assignment Problem. The following chapters will tackle the integrated Berth Allocation and Quay Crane Assignment Problem.

Chapter 4

A new mixed integer linear model for the Berth Allocation and Quay Crane Assignment Problem

4.1 Introduction

As we have seen in the previous chapters, in the standard continuous Berth Allocation Problem the handling times are usually assumed to be fixed and known in advance. However, in the seaside of container terminals, quay cranes (QCs) are also a scarce resource that may affect the service time of vessels. The number of cranes serving a vessel simultaneously is often restricted between a minimum and a maximum number, for either technical or contractual reasons, and several vessels may be concurrently handled at the quay, so an efficient assignment of the cranes is also required to reduce the delays and the costs incurred by the terminal. Given that the number of QCs assigned to a vessel determines its handling time and the handling time is taken into account in the berth allocation, there is an increasing trend to consider these two problems together. In the combined Berth Allocation and Quay Crane Assignment Problem (BACAP), as well as the time and berthing position, a number of cranes has to be assigned to each vessel. Moreover, if is also necessary to determine which specific cranes are to serve each vessel, a Berth Allocation and Specific Quay Crane Assignment Problem (BACASP) arises.

As has already been explained, two versions of the BACAP have been considered in the literature (Chapter 2.6). In the *time-invariant* version, the number of cranes assigned to each vessel remains constant throughout its handling, while in the *variable-in-time* version this number can be changed in each period. The variable-in-time version of the problem allows a more efficient use of cranes, as those initially assigned to a vessel can be reassigned to newly arrived vessels. However, this advantage can turn into a serious problem when applying the plan to real situations. The solutions may entail more complex crane-to-vessel assignments and thus become more difficult for human operators to implement. Moreover, they can also result in a greater number of crane movements, thereby demanding a more effective and efficient management of operations at the docks,

including the organization of workers and equipment. Consequently, variable-in-time plans are more sensitive to contingencies and elements not usually considered in models (such as the moving time of cranes) and thus the real execution may rapidly deviate from the theoretical schedule. Therefore, in these models the number of crane movements should be minimized, taking them into account in the objective function. By contrast, time-invariant models require, in general, fewer variables than variable-in-time models, since the assignment of the number of cranes has to be carried out for each vessel and not for each time period. This makes these models computationally easier to handle, and therefore the size of the instances that can be solved to optimality is usually larger than in the case of variable-in-time models. At any rate, both versions of the problem are interesting in practice and are receiving increasing attention in the literature.

This chapter addresses the time-invariant BACAP, while the time-invariant BACASP will be tackled in the next chapter. The BACAP is formulated by proposing a new mixed integer linear model, which is then enhanced by adding several families of valid inequalities. This approach is evaluated and compared with state-of-the-art proposals through several computational experiments on different sets of instances.

The rest of this chapter is organized as follows. Section 4.2 reviews previous studies relating to both the BACAP and BACASP. Section 4.3 describes the problem and the specific assumptions of this approach. The new model for the BACAP is presented in Section 4.4. In Section 4.5, several constraints are proposed to reinforce the formulation. Section 4.6 describes the experiments conducted and presents the discussion of the results. Finally, some conclusions are drawn in Section 4.7.

4.2 Literature review

In this section, the previous studies on the continuous BACAP and BACASP are reviewed. Comprehensive reviews that address the different versions of these problems, including discrete and hybrid quay variants, were commented on Chapter 2.5.

4.2.1 Variable-in-time crane assignment

The pioneering work on the continuous BACAP was presented by Park and Kim (2003). They formulated an MILP for the problem with variable-in-time crane assignment. Moreover, they proposed a Lagrangean relaxation of the model and used the subgradient method to obtain near-optimal solutions on randomly generated instances of up to 40 vessels within a planning horizon of one week. They also presented a dynamic programming algorithm to obtain a solution for the corresponding BACASP minimizing the number of crane changes. On the basis of this approach, Zhang et al. (2010) proposed a mixed integer linear model for a BACASP that takes into account the limited range of movement of each crane.

Meisel and Bierwirth (2009) studied a continuous BACAP with variable-in-time crane assignment and the possibility of speeding up the arrival of each vessel, incurring a cost proportional to the time advanced. They also considered decreasing marginal crane productivity as the number of cranes serving a vessel increases, due to interferences

between them. The authors proposed an MILP and two metaheuristic approaches: a Tabu Search and a Squeaky Wheel Optimization. They reported good results on previous and newly generated instances of up to 40 vessels within a planning horizon of one week. Elwany et al. (2013) studied the same problem considering water depth variation depending on position. They extended the model of Meisel and Bierwirth (2009) and proposed a Simulated Annealing based on a heuristic capable of constructing feasible solutions from ordered lists of vessels. Xuelian and Zhiying (2012) proposed an integer linear model and a decomposition heuristic procedure to assign quay cranes and berthing positions and times. Liang et al. (2012) first applied the BAP model of Kim and Moon (2003) and then assigned a number of QCs to each vessel assuming that it is dependent on its requested departure time. To do so, they developed several heuristics and applied them in a Sequence Optimized Particle Swarm Optimization framework. At a strategic level, Hendriks et al. (2012) tackled a BACAP considering multi-terminal operations.

More recently, Iris et al. (2015) addressed the same problem in both the variable-in-time and time-invariant versions. They proposed several novel set partitioning formulations and some variable reduction techniques. They compared with Meisel and Bierwirth (2009) on the same instances and reported several improvements on previous results. Raa et al. (2011) and Xiao and Hu (2014) proposed several MILPs for rolling-horizon strategies, and Hu (2015) also considered a rolling horizon with the novelty of periodic balancing utilization of the quay cranes.

Türkoğullari et al. (2016) proposed both an MILP for the deeply integrated variable-in-time BACASP and a cutting plane algorithm based on a decomposition scheme. The results reported show that their method could obtain optimal solutions for instances with up to 60 vessels arriving within a planning horizon of 600 hours. Han et al. (2015) addressed a variable-in-time BACAP taking into consideration the movement ranges of the quay cranes. They proposed an MILP for the BAP, another for the quay crane assignment problem (QCAP), and a Particle Swarm Optimization heuristic. A related approach was proposed by Karam and Eltawil (2016), who developed a functional integration of two independent models for the BAP and the QCAP able to obtain good solutions on instances of up to 21 vessels arriving within a planning horizon of 168 hours.

A further approximation to reality was achieved by Rodriguez-Molins et al. (2014a), who considered the moving time of cranes along the serving vessel and along the quay in both the variable-in-time and the time-invariant BACASP. They proposed a GRASP that was applied on real-life instances of up to 20 vessels. Rodriguez-Molins et al. (2014b) also proposed a multi-objective MILP and a Multi-Objective Genetic Algorithm with Simulated Annealing to perform robust scheduling on realistic instances. Likewise, Shang et al. (2016) considered the setup times of quay cranes, including them in the mathematical formulation. They proposed a Genetic Algorithm and three MILPs: a basic model, a robust model capable of dealing with uncertainties, and a version of the latter with price constraints. These methods were tested on the instances of Meisel and Bierwirth (2009) and obtained optimal solutions on those involving up to 20 vessels.

Chang et al. (2010) introduced the energy consumption of quay cranes in a multi-objective MILP and a Parallel Genetic Algorithm tailored to address a rolling horizon

scheme for the BACASP. They reported good results on instances of up to 40 vessels within a planning horizon of 72 hours. Hu et al. (2014) also included fuel consumption and emissions incurred by vessels and quay cranes in a new multi-objective programming model. Good solutions were obtained on instances with up to 20 vessels within the same planning horizon. More recently, He (2016) also considered energy consumption in the BACAP and proposed an MILP and a Memetic Algorithm capable of obtaining both optimal solutions on instances with up to 24 vessels and good solutions on those with up to 40 vessels.

Beyond this, Meisel and Bierwirth (2013) integrated the main optimization problems that appear in the seaside of container terminals (BAP, QCAP and QCSP) in an iterative framework with three phases in which several MILPs were solved. It was tested on instances of up to 40 vessels within a planning horizon of one week.

4.2.2 Time-invariant crane assignment

Blazewicz et al. (2011) addressed for the first time a continuous BACASP with time-invariant crane assignment by formulating a moldable task scheduling problem. This approach was tested in experiments considering up to 45 vessels within a time horizon of 100 hours. Yang et al. (2012) proposed a Nested Loop-based Evolutionary Algorithm for the BACASP and obtained good results compared with Park and Kim (2003). Le et al. (2012) addressed the BACASP by proposing a multi-objective integer linear model and a Multi-Objective Particle Swarm Optimization.

A different approach was proposed by Chen et al. (2012), who developed a combinatorial Bender's cuts algorithm based on a previous BACASP model. They were able to obtain good solutions on instances with up to 26 vessels within a planning horizon of one week. As has been already mentioned, Iris et al. (2015) also developed a generalized set partitioning formulation for the time-invariant BACAP, while Rodriguez-Molins et al. (2014a) developed a GRASP for the time-invariant BACASP.

Türkoğullari et al. (2014) tackled both the BACAP and the BACASP with time-invariant crane assignment, for which they proposed two MILPs, an algorithm to get a BACASP solution from a BACAP solution, and a cutting plane algorithm. The authors reported that their model for the BACAP and the cutting plane algorithm for the BACASP obtained optimal solutions on instances of up to 60 vessels within a planning horizon of 600 hours.

After reviewing the most recent studies on the continuous BACAP and BACASP, it can be concluded that the time-invariant version of the problem has received less attention than the variable-in-time version. The following section describes the time-invariant BACAP addressed in this study, which is the same problem studied by Türkoğullari et al. (2014).

4.3 Problem description

4.3.1 Overview

The Berth Allocation and quay Crane Assignment Problem (BACAP) is the optimization problem of assigning berth position, number of cranes, and berthing time to calling vessels, minimizing the total assignment cost. In the version of the problem addressed here, there is only one continuous quay and the number of quay cranes serving a vessel, once assigned, is kept fixed during its handling. The arrival time of each vessel is known in advance, as well as the maximum and minimum number of cranes that it can admit and an estimate of its handling time for each number of cranes. Vessels can be moored along the quay within a given time horizon, while quay cranes can move along the quay to serve the vessels provided that they do not cross each other. A berth plan can be rendered as a space-time diagram as in the case of the BAP, now considering also the number of cranes assigned to each vessel (Figure 4.1).

For each vessel, three different costs are considered: the cost of waiting before berthing, the cost of delay after the desired departure time, and the cost of deviating from the desired position on the quay. The waiting time is the difference between the berthing time of the vessel and its expected arrival time, while the delay is the difference between the desired departure time and the assigned departure time, if it is greater than zero. The desired position on the quay is usually the position closest to the location in the yard where the containers to be loaded onto/unloaded from the vessel are placed. Therefore, the deviation from the desired position is the distance between that position and the assigned berthing position. According to the scheme proposed by Bierwirth and Meisel (2010), this problem can be classified as $cont \,|\, dyn \,|\, QCAP \,|\, \sum(w_1\, pos + w_2\, wait + w_3\, tard)$ and BAP, QCAP(number).

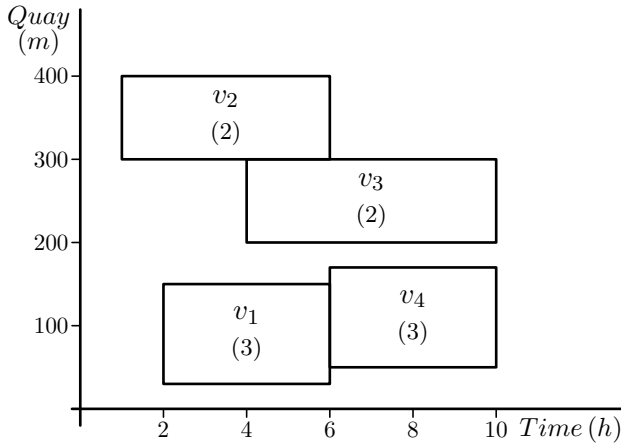


Figure 4.1: BACAP solution with 5 vessels and a quay 400 m long with 7 cranes. The number of cranes appears in parentheses.

4.3.2 Assumptions

The assumptions of this approach are the following:

- Time
  - The planning horizon is divided into multiple equal time segments.
  - Vessels are to be moored within the planning horizon.
- Quay
  - Each position on the quay can accommodate one vessel at a time.
- Vessels
  - When a vessel is moored, the berthing position is kept fixed.
  - Once started, the handling of a vessel cannot be interrupted.
  - The handling time of each vessel is considered to be independent of its berthing position. This assumption is reasonable if the quay has enough machinery and workers for container transportation between the yard and the quay at any moment. Hence, the cranes serving each vessel do not need to wait for vehicles. The increased transportation cost produced if the position of the vessel deviates from its desired position is included in the objective function.
  - The handling time of each vessel depends on the number of cranes assigned to it. No specific relation between them is assumed, so it can be either linear or non-linear.
  - The time for docking and undocking maneuvers is considered to be included in the vessel handling time.
  - Vessels may have different relative importance. Therefore, cost coefficients are specific to each vessel.
  - The inter-ship clearance is included in the vessel length. In general, for vessels longer than 130 m, this clearance corresponds to 10% of its length. For small vessels, the minimum clearance is 10 m.
- Cranes
  - The number of cranes available at the quay is fixed and all the cranes have the same characteristics.
  - All quay cranes can move along the whole length of the quay, but they cannot cross each other.
  - Each quay crane can be assigned to one vessel at most in each time period.
  - The number of quay cranes assigned to a vessel does not change during its stay at the quay.
  - There is a minimum and a maximum number of cranes that can be assigned to a vessel.

4.3.3 Parameters

The following data on the terminal and the vessels are known in advance:

- Set of time periods: $T = \{1, 2, ..., H\}$, where H is the planning horizon
- Length of the quay: L
- Number of quay cranes: Q
- Set of calling vessels: V, with $N = |V|$

  For each vessel $i \in V$, the following information is known:

  - Length: l_i
  - Expected arrival time: a_i
  - Desired departure time: s_i
  - Minimum and maximum number of quay cranes that can be assigned to the vessel: q_i^{min}, q_i^{max}
  - Estimated handling time if it is handled using q cranes: h_i^q
  - Desired position on the quay: d_i
  - Cost per period of waiting for berthing after the expected arrival time: C_i^w
  - Cost per period of delay after the desired departure time: C_i^d
  - Cost per length unit away from the desired position on the quay: C_i^p

The handling time of each vessel can be inversely proportional to the number of cranes, as in Park and Kim (2003), or can take into account decreasing marginal productivity due to crane interferences, as in Meisel and Bierwirth (2009). Any relationship between the number of cranes and the handling times can be included in the model by specifying the particular handling time h_i^q of each vessel i if q quay cranes are considered.

4.4 A mixed integer linear model

A new mixed integer linear model is proposed for the problem previously described, inspired by the models of both Meisel and Bierwirth (2009) and Türkoğullari et al. (2014).

4.4.1 Variables

The following variables are defined:

t_i = berthing time of vessel i

p_i = berthing position of vessel i

u_i = delay of vessel i

e_i = deviation of vessel i from its desired position on the quay

$$r_{iqt} = \begin{cases} 1, & \text{if the handling of vessel } i \text{ with } q \text{ cranes starts at time } t \\ 0, & \text{otherwise} \end{cases}$$

$$\sigma_{ij} = \begin{cases} 1, & \text{if vessel } i \text{ is completely to the left of vessel } j \text{ in the space-time} \\ & \text{diagram, that is, vessel } i \text{ is completely handled before } j \\ 0, & \text{otherwise} \end{cases}$$

$$\delta_{ij} = \begin{cases} 1, & \text{if vessel } i \text{ is completely below vessel } j \text{ in the space-time diagram,} \\ & \text{that is, vessel } i \text{ is positioned completely to the right of vessel } j \\ & \text{looking at them from the quay} \\ 0, & \text{otherwise} \end{cases}$$

4.4.2 Objective and constraints

$$Min \sum_{i \in V} (C_i^w (t_i - a_i) + C_i^d u_i + C_i^p e_i) \tag{4.1}$$

s. t.

$$\sum_{t=a_i}^{H} \sum_{q=q_i^{min}}^{q_i^{max}} r_{iqt} = 1, \qquad \forall i \in V \tag{4.2}$$

$$t_i = \sum_{t=a_i}^{H} \sum_{q=q_i^{min}}^{q_i^{max}} r_{iqt} \cdot t, \qquad \forall i \in V \tag{4.3}$$

$$t_j - t_i - \sum_{t=a_i}^{H} \sum_{q=q_i^{min}}^{q_i^{max}} (r_{iqt} \cdot h_i^q) - (\sigma_{ij} - 1)H \geq 0, \qquad \forall i, j \in V, i \neq j \tag{4.4}$$

$$p_j - (p_i + l_i) - (\delta_{ij} - 1)L \geq 0, \qquad \forall i, j \in V, i \neq j \tag{4.5}$$

$$\sigma_{ij} + \sigma_{ji} + \delta_{ij} + \delta_{ji} \geq 1, \qquad \forall i, j \in V, i \neq j \tag{4.6}$$

$$p_i + l_i \leq L + 1, \qquad \forall i \in V \tag{4.7}$$

$$\sum_{i \in V} \sum_{q=q_i^{min}}^{q_i^{max}} \sum_{\tau=\max(a_i, t-h_i^q+1)}^{t} r_{iq\tau} \cdot q \leq Q, \qquad \forall t \in T \tag{4.8}$$

$$u_i \geq t_i - s_i + \sum_{t=a_i}^{H} \sum_{q=q_i^{min}}^{q_i^{max}} (r_{iqt} \cdot h_i^q) - 1, \qquad \forall i \in V \tag{4.9}$$

$$e_i \geq p_i - d_i, \qquad \forall i \in V \tag{4.10}$$

$$e_i \geq d_i - p_i, \qquad \forall i \in V \tag{4.11}$$

$$\sigma_{ij}, \delta_{ij}, \in \{0, 1\}, \qquad \forall i, j \in V, i \neq j \tag{4.12}$$

$$r_{iqt} \in \{0, 1\}, \qquad \forall i \in V, \forall t \in \{a_i, \ldots, H\}, \forall q \in \{q_i^{min}, \ldots, q_i^{max}\} \tag{4.13}$$

$$u_i, e_i \geq 0, \qquad \forall i \in V \tag{4.14}$$

$$p_i \geq 1, \qquad \forall i \in V \tag{4.15}$$

The objective function (4.1) is the minimization of the overall planning cost, which consists of the sum of the waiting cost, the delay cost, and the deviation cost incurred by the terminal for each vessel. The handling of each vessel starts only once and with a fixed number of cranes (4.2). Constraints (4.3) link the berthing time of a vessel to variables r_{iqt} to prevent inconsistencies. Constraints (4.4)–(4.6) prevent overlaps between vessels in space and time. Constraints (4.7) forbid berthing positions in which the vessel would exceed the quay length. Moreover, the number of cranes assigned to vessels in each time period cannot be greater than the total number of cranes available at the quay due to constraints (4.8). Constraints (4.9) define the delay of each vessel and constraints (4.10)–(4.11) the deviation from the desired position. Finally, constraints (4.12)–(4.15) define the type of the variables. Note that (4.15) determines that the first position available at the quay is 1.

4.5 Enhancing the integer linear model for BACAP: valid inequalities

In this section, several families of valid inequalities are proposed to enhance the model previously described. These inequalities will reduce the solution polyhedron of the linear relaxation corresponding to the integer model, thereby making it closer to the convex hull of the set of integer solutions. Consequently, they will improve the lower bounds and hopefully will reduce the search in branch-and-bound strategies. Nevertheless, it is important not to overload the model with too many additional constraints, because the positive effects can be counteracted by the extra effort needed to solve the linear relaxations. Therefore, the description of each type of valid inequality also indicates the way in which it is implemented, based on the idea that most of the conflicts between vessels appear in the vicinity of their ideal assignments.

4.5.1 Strengthening the non-overlapping constraints

If two vessels i and j are concurrent in time according to variables r_{iqt} and r_{jqt}, then they must be separated in space. Although this constraint can be built in general for any pair of times t_i and t_j that ensure the concurrence, here it is described for the particular case in which they are close to their arrival times, because these are the times at which the vessels will try to be berthed and therefore the constraint to solve the conflict between them will be active.

For each pair of vessels i, j, such that $i \neq j$, $a_i + \min(h_i^q) > a_j$ and $a_j + \min(h_j^q) > a_i$, the following can be stated:

$$\sigma_{ij} + \sigma_{ji} + \sum_{q=q_i^{min}}^{q_i^{max}} \sum_{t=a_i}^{a_j+\min(h_j^q)-1} r_{iqt} + \sum_{q=q_j^{min}}^{q_j^{max}} \sum_{t=a_j}^{a_i+\min(h_i^q)-1} r_{jqt} \leq 2 \tag{4.16}$$

If both terms involving variables r_{iqt} and r_{jqt} take value 1, the vessels overlap in time and then $\sigma_{ij} = \sigma_{ji} = 0$. According to constraint (4.6), $\delta_{ij} + \delta_{ji} \geq 1$ and therefore the vessels will be separated in space.

Figure 4.2 shows an example of concurrence in time. If vessel i starts at a time $t_i \in \{a_i, a_j + \min(h_j^q) - 1\}$ and vessel j starts at a time $t_j \in \{a_j, a_i + \min(h_i^q) - 1\}$, they concur in time and therefore $\delta_{ij} + \delta_{ji} \geq 1$, separating them in space.

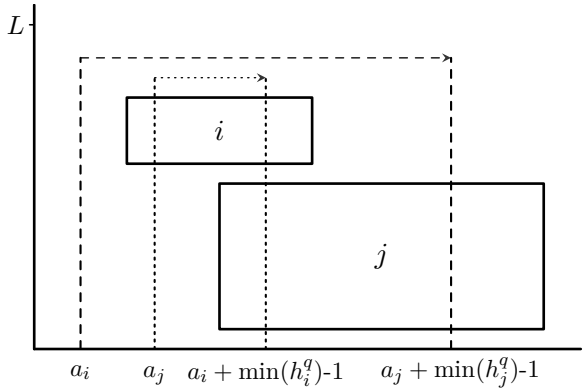


Figure 4.2: If vessels concur in time, they must be separated in space.

4.5.2 Minimum separation in time and space for vessels in their desired positions

If a pair of vessels would overlap if they were moored in their desired positions at their arrival times, a minimum separation between them will be needed, in time or in space. Figure 4.3 shows the case of two vessels i and j at their least-cost assignment. If they are separated in time ($\sigma_{ij} + \sigma_{ji} = 1$), the minimum separation *minTimeMove* (in the figure: $a_i + \min(h_i^q) - a_j$) allows us to state:

$$t_i + t_j \geq a_i + a_j + minTimeMove \cdot (\sigma_{ij} + \sigma_{ji}), \qquad \forall i, j \in V, i \neq j \tag{4.17}$$

If they are separated in space ($\delta_{ij} + \delta_{ji} = 1$), the minimum separation *minSpaceMove* (in the figure: $d_i + l_i - d_j$) leads to:

$$e_i + e_j \geq minSpaceMove \cdot (\delta_{ij} + \delta_{ji}), \qquad \forall i, j \in V, i \neq j \tag{4.18}$$

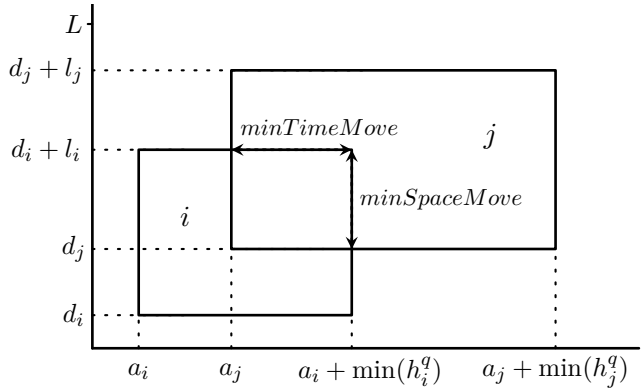


Figure 4.3: Minimal separation in time and space.

4.5.3 Cover constraints on ordered sets of variables δ

Given a subset of vessels such that the sum of their lengths exceeds the quay length, not all of them can be handled concurrently at the quay and thus at least one of them must be separated in time. Figure 4.4 shows an example of three vessels i, j, k, with $l_i + l_j + l_k > L$. Therefore, the inequality $\delta_{ij} + \delta_{jk} + \delta_{ki} \leq 1$ can be added. As this condition is satisfied for any ordering of the vessels, additional constraints corresponding to other orderings can also be included, for instance, $\delta_{ji} + \delta_{ik} + \delta_{kj} \leq 1$.

In general, if a subset of vessels S is identified such that the sum of their lengths exceeds the length of the quay, for each permutation of vessels in S, $i, j, k, .., n, m$, the following constraint can be added:

$$\delta_{ij} + \delta_{jk} + \cdots + \delta_{nm} + \delta_{mi} \leq |S| - 2, \qquad i, j, k, ..., n, m \in S \tag{4.19}$$

In order to avoid adding too many constraints of this kind, they are considered only for minimal subsets S of vessels that do not fit together at the quay and would concur in time if they were moored at their arrival times, considering their maximum handling times. Additionally, a parameter α is used to establish the maximum subset cardinality admitted.

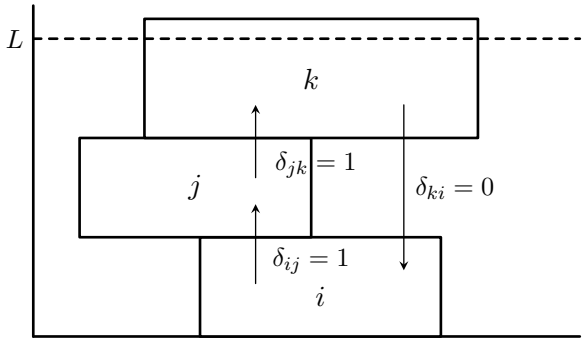


Figure 4.4: Vessels that do not fit together at the quay.

4.5.4 Cover constraints on non-ordered sets of variables δ

Given a subset of vessels S such that the sum of their lengths exceeds the quay length, as in the previous case, instead of considering it as an ordered set, all the permutations and all the variables separating the vessels in space can be considered simultaneously, as can be seen in Figure 4.5. Thus a valid inequality for this set is:

$$\sum_{i \in S} \sum_{j \in S, j \neq i} \delta_{ij} \leq \frac{|S|^2 - |S|}{2} - 1 \tag{4.20}$$

The total number of δ variables corresponding to the vessels in S is $|S|^2 - |S|$, and as there is a pair of variables δ_{ij}, δ_{ji} for each pair of vessels i, j, the maximum value of the sum of the variables will be $(|S|^2 - |S|)/2$. If the sum of the vessel lengths exceeds L, not

all the vessels in S can be separated in space, so the value of the sum of the variables must be lower than or equal to $(|S|^2 - |S|)/2 - 1$.

As in the previous case, this kind of valid inequality is only considered for minimal subsets of vessels S that do not fit together at the quay and concur in time if they are moored at their arrival times, considering their maximum handling times.

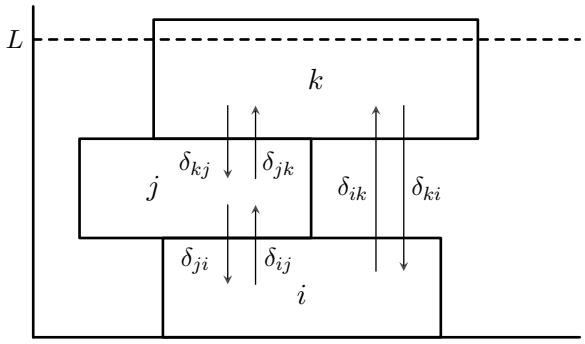


Figure 4.5: Constraint on a non-ordered set of vessels.

4.5.5 Cover constraints on variables r_{iqt}

The last family of valid inequalities also refers to groups of vessels that cannot fit together at the quay, but this time cover constraints on variables r_{iqt} are considered. Now these subsets of vessels are identified among the vessels that would concur in time if they were placed in their ideal assignments, considering their corresponding minimum handling times to ensure validity. Let S be a set of vessels satisfying these conditions and $minDep_i^S$ the minimum departure time in the ideal assignment of the vessels in S, not including vessel i. Thus the following inequality can be added:

$$\sum_{i \in S} \sum_{q=q_i^{min}}^{q_i^{max}} \sum_{t=a_i}^{minDep_i^S - 1} r_{iqt} \leq |S| - 1 \tag{4.21}$$

4.6 Computational experiments

The models and the valid inequalities proposed were tested and compared with other recent proposals to assess their efficiency and limits on instances of realistic size. This section describes the experiments conducted and discusses their results.

4.6.1 Test instances and implementation issues

Several sets of instances were generated according to the criteria used in previous papers. The first set was generated applying the criteria of Park and Kim (2003). From now on, it will be referred to as *GenPK*. For all the instances in this set a quay with length

$L = 1200$ metres and a time horizon $H = 300$ hours are considered, discretized in units of 10 metres and 1 hour respectively. There are $Q = 11$ quay cranes available and $q_i^{min} = 2$, $q_i^{max} = 5$, $\forall i \in V$. The set consists of 50 randomly generated instances, 10 instances for each number of vessels considered: $N \in \{20, 25, 30, 35, 40\}$. The data relating to each vessel are determined by uniform distributions as follows: $U[1, 170]$ for the the arrival time, $U[10, 48]$ for the number of crane-hours required, $U[15, 35]$ for the length, and $U[1, 120]$ for the desired position on the quay. The desired departure time, which is not specified in the paper, is determined by applying the criterion of Meisel and Bierwirth (2009): $s_i = a_i + 1.5 \cdot \min(h_i^q)$. The vessel handling time for each number of cranes results from dividing the number of crane-hours specified for the vessel by the number of cranes, rounded up to the next integer. Finally, the costs are the same for all vessels, $C_i^w = 1000$, $C_i^d = 2000$, $C_i^p = 200$, $\forall i \in V$.

Another set of instances was generated, this time based on the criteria applied by Meisel and Bierwirth (2009). From this point on, it will be referred to as *GenMB-10m*. It consists of 50 instances, 10 for each number of vessels considered: $N \in \{20, 30, 40, 50, 60\}$. The time unit is 1 hour, the planning horizon $H = 210$ hours, and the quay is 1000 metres long. Moreover, there are 10 cranes on the quay and three different kinds of vessels: Feeder, Medium and Jumbo. Within each instance, 60%, 30% and 10% of the vessels correspond to these classes, respectively. The arrival times of the vessels are uniformly distributed between 0 and 168 (one week). A planning horizon of 210 hours was chosen instead of 168 to prevent the generation of infeasible instances. The lengths, workloads (in crane-hours), and min-max number of cranes of the vessels are computed according to Table 4.1, taken from the cited paper, while the desired position of each vessel i is generated from $U[1, L + 1 - l_i]$. For each vessel, its handling time for a given number of cranes assigned to it results from dividing its workload in crane-hours by the number of cranes, rounded up to the nearest integer. The desired departure time of each vessel i is $a_i + 1.5 \cdot \min(h_i^q)$. The costs are the same for all the vessels, so none of them is privileged over the others: $C_i^w = 1000$, $C_i^d = 2000$, and $C_i^p = 200$.

An additional set, called *GenMB-50m*, was generated with the same instances as *GenMB-10m*, changing the discretization of vessel and quay lengths to 50 metres. In particular, the lengths were discretized and rounded to the next integer, and the desired positions were discretized and rounded to the nearest integer. The cost coefficients are also the same, except in the case of the deviation cost coefficient, which was changed to maintain the same cost per metre as in *GenMB-10m*: $C_i^p = 1000$. The set *GenMB-50m* is only considered in comparisons with previous works.

Table 4.1: Specifications of the test instances generated by Meisel and Bierwirth.

Class	Length	Crane-hours	q_i^{min}	q_i^{max}
Feeder	$U\,[8, 21]$	$U\,[5, 15]$	1	2
Medium	$U\,[21, 30]$	$U\,[15, 50]$	2	4
Jumbo	$U\,[30, 40]$	$U\,[50, 65]$	4	6

The model was implemented in C++11 and solved using CPLEX 12.6, limiting the size of the search tree to 30 GiB. The experiments were run on an Intel Core i7 2600 at 3.4 GHz with 31.4 GiB of RAM. The operating system used was Ubuntu 14.04 GNU/Linux 3.13 and the compiler was GCC–G++ 6.2. The compilation was performed specifying the special parameters *–O3 –march=corei7* in order to generate an executable file that makes the most of the processor.

4.6.2 Evaluation of the MILP and the proposed valid inequalities

In order to evaluate the quality of the model proposed and the influence of the proposed valid inequalities, several experiments were conducted, first considering the proposed MILP alone and then adding each of the valid inequalities described in Section 4.5. The parameter α limiting the cardinality of the subsets of vessels involved in constraints (4.19) was set to 4. Each configuration was run on each instance with a time limit of 1 hour.

The results obtained for *GenPK* and *GenMB-10m* are shown in Tables 4.2 and 4.3, respectively. For each subset of 10 instances of a given size, the tables show the number of instances solved to optimality, the average computation time in seconds, and the average and maximum gap in percentage. For each instance, the gap was provided by CPLEX as $100 \cdot (ub - lb)/ub$, where lb is the value of the best lower bound obtained within the time limit, and ub is the value of the objective function corresponding to the best integer solution achieved in the same process.

The results in Table 4.2 show that all but one of the instances in the *GenPK* set were solved to optimality. Only one instance of 40 vessels was not optimally solved, with a gap in the initial configuration of 3.2%. Table 4.3 shows similar results for the *GenMB-10m* set. All the 40-vessel instances were optimally solved, but only 5 of the instances of 50 vessels and none of the instances of 60 vessels could be solved to optimality. For instances of 50 vessels, the gaps for instances not optimally solved were quite low, with a maximum of 7.7% in the initial configuration. For instances of 60 vessels, the gaps were larger, with an average of 20.4% and a maximum of 42.5%. The information in these tables indicates that 50 vessels is the maximum instance size for which the model can be used with a guarantee of obtaining an optimal or quasi-optimal solution.

With respect to the influence of the valid inequalities, Table 4.2 also shows that the added inequalities, separately or all together, do not improve the good performance of the initial model on the set *GenPK*. However, in Table 4.3 it can be observed that although each inequality by itself does not improve the results on the set *GenMB-10m*, by adding all the valid inequalities two more instances were solved to optimality, the average running times were reduced (except on the instances with 60 vessels) and the average and maximum gaps also decreased. Therefore, it was decided to use the model including all the valid inequalities for the next experiments.

Table 4.2: Solving the BACAP on instances *GenPK*.

Model	Vessels	Optimum	Time	Avg. gap	Max. gap
MILP	20	10	1.2	0	0
	25	10	2.3	0	0
	30	10	6.2	0	0
	35	10	39.7	0	0
	40	9	625.2	0.3	3.2
MILP +	20	10	1.2	0	0
constraints (4.16)	25	10	1.9	0	0
	30	10	5.0	0	0
	35	10	49.5	0	0
	40	9	549.3	0.5	5.0
MILP +	20	10	1.1	0	0
constraints (4.17), (4.18)	25	10	2.0	0	0
	30	10	4.8	0	0
	35	10	48.0	0	0
	40	9	585.0	0.5	4.9
MILP +	20	10	1.4	0	0
constraints (4.19)	25	10	2.1	0	0
	30	10	5.0	0	0
	35	10	72.5	0	0
	40	9	494.5	0.5	4.6
MILP +	20	10	1.5	0	0
constraints (4.20)	25	10	2.4	0	0
	30	10	5.6	0	0
	35	10	51.6	0	0
	40	9	561.7	0.8	8.0
MILP +	20	10	1.4	0	0
constraints (4.21)	25	10	2.4	0	0
	30	10	5.2	0	0
	35	10	39.6	0	0
	40	9	532.1	0.5	5.2
MILP +	20	10	1.2	0	0
all constraints	25	10	1.7	0	0
	30	10	4.0	0	0
	35	10	53.8	0	0
	40	9	595.2	0.6	6.4

Table 4.3: Solving the BACAP on instances *GenMB-10m*.

Model	Vessels	Optimum	Time	Avg. gap	Max. gap
MILP	20	10	0.6	0	0
	30	10	5.6	0	0
	40	10	14.6	0	0
	50	5	2440.3	2.6	7.7
	60	0	3600.0	20.4	42.5
MILP +	20	10	0.6	0	0
constraints (4.16)	30	10	5.3	0	0
	40	10	11.3	0	0
	50	5	2455.6	3.0	8.7
	60	0	3600.0	20.4	45.3
MILP +	20	10	0.7	0	0
constraints (4.17), (4.18)	30	10	4.8	0	0
	40	10	15.3	0	0
	50	6	2197.4	2.2	6.4
	60	0	3600.0	16.6	27.5
MILP +	20	10	0.7	0	0
constraints (4.19)	30	10	4.3	0	0
	40	10	12.5	0	0
	50	5	2351.6	2.4	9.3
	60	0	3600.0	18.8	40.1
MILP +	20	10	0.7	0	0
constraints (4.20)	30	10	5.6	0	0
	40	10	16.1	0	0
	50	5	2661.3	3.3	10.8
	60	0	3600.0	19.8	39.9
MILP +	20	10	0.6	0	0
constraints (4.21)	30	10	5.4	0	0
	40	10	19.0	0	0
	50	5	2710.2	3.3	10.2
	60	0	3600.0	18.4	36.2
MILP +	20	10	0.6	0	0
all constraints	30	10	4.5	0	0
	40	10	9.5	0	0
	50	7	2084.4	1.5	6.5
	60	0	3600.0	16.2	30.0

4.6.3 Comparison with Türkoğullari et al.

The proposed model was compared with the BACAP model presented by Türkoğullari et al. (2014), which considers time-invariant crane assignment and solves exactly the same problem. In order to perform a proper comparison, it was implemented in C++11 and solved using CPLEX 12.6 under the same conditions as the model proposed. Thus it was run over the same sets of instances: *GenPK*, *GenMB-10m*, and *GenMB-50m*, on the same computer.

For the set of instances *GenPK*, the model proposed by Türkoğullari et al. (2014) could not even achieve integer solutions for any of the instances within the time limit of 1 hour. The reason is that the memory available in the computer (31.4 GiB) was not enough to construct the model. The results for sets *GenMB-10m* and *GenMB-50m* are shown in Tables 4.4 and 4.5 respectively. As in previous tables, times are given in seconds and gaps in percentages. The number of instances in each group in which at least an integer solution was found is shown in the rows labelled "Solved". The empty fields indicate that no integer solutions were found within the time limit and thus it was not possible to calculate the statistics.

Table 4.4: Results of the BACAP model of Türkoğullari et al. and the model proposed on the set *GenMB-10m*.

Vessels		Model of Türkoğullari et al.	MILP with valid inequalities
20	Solved	10	10
	Optimum	10	10
	Avg. time	309.6	0.6
	Avg. gap	0	0
	Max. gap	0	0
30	Solved	10	10
	Optimum	10	10
	Avg. time	748.6	4.5
	Avg. gap	0	0
	Max. gap	0	0
40	Solved	5	10
	Optimum	5	10
	Avg. time	1567.0	9.5
	Avg. gap	0	0
	Max. gap	0	0
50	Solved	1	10
	Optimum	0	7
	Avg. time	2301.0	2084.4
	Avg. gap	62.4	1.5
	Max. gap	62.4	6.5
60	Solved	0	10
	Optimum		0
	Avg. time		3600.0
	Avg. gap		16.2
	Max. gap		30.0

Table 4.4 shows that the model proposed obtained better results than the model presented by Türkoğullari et al. (2014) over the set of instances *GenMB-10m*. It was able to solve to optimality instances of up to 40 vessels in very short computation times, and almost all the instances of 50 vessels in half an hour, attaining low gaps when optimality could not be proven within the time limit.

Table 4.5: Results of the BACAP model of Türkoğullari et al. and the model proposed on the set *GenMB-50m*.

Vessels		Model of Türkoğullari et al.	MILP with valid inequalities
20	Solved	10	10
	Optimum	10	10
	Avg. time	14.3	0.3
	Avg. gap	0	0
	Max. gap	0	0
30	Solved	10	10
	Optimum	10	10
	Avg. time	28.0	7.3
	Avg. gap	0	0
	Max. gap	0	0
40	Solved	10	10
	Optimum	10	10
	Avg. time	48.1	14.7
	Avg. gap	0	0
	Max. gap	0	0
50	Solved	10	10
	Optimum	9	4
	Avg. time	1015.6	2785.9
	Avg. gap	0.2	3.9
	Max. gap	2.2	15.4
60	Solved	10	10
	Optimum	4	0
	Avg. time	2397.7	3600.0
	Avg. gap	10.7	20.6
	Max. gap	26.0	36.2

With respect to the set of instances *GenMB-50m*, the model of Türkoğullari et al. (2014) obtained more optimal solutions and lower gaps for large instances with 50 and 60 vessels (Table 4.5). The performance of the MILP proposed here was similar to that attained over the set *GenMB-10m*. The different performance of the model of Türkoğullari et al. (2014) shown on the sets *GenMB-10m* and *GenMB-50m* could be explained by its dependence on the discretization factor applied to the lengths of the quay and the vessels. That model considers the position on the quay as an index of its binary variables, and therefore the number of variables increases dramatically when

the discretization is finer. This also explains the memory limitations observed when applying that model.

Given that the set *GenMB-50m* is a discretization of *GenMB-10m*, the results obtained by the models on each instance can be compared. The costs of the solutions obtained by the proposed model with a discretization of 10 metres are lower than those obtained with a discretization of 50 metres. Overall, the costs decrease 11.3% on average, as the better utilization of the quay results in lower costs related to deviations from the desired positions of the vessels and in better vessel assignments which reduce the waiting times and the delays. The greater number of optimal solutions obtained by the model of Türkoğullari et al. (2014) in *GenMB-50m* do not lead to more efficient assignments in terms of cost than those achieved by the proposed model on the set *GenMB-10m*. In fact, the costs of those optimal solutions were 8.7% greater than the costs of the solutions obtained by the proposed model on average. Therefore, unless there are specific reasons at a given terminal for using a discretization of 50 metres, a finer discretization, for example of 10 metres, will produce better solutions.

4.7 Concluding remarks

In this chapter, a new mixed integer linear formulation has been proposed for the continuous time-invariant Berth Allocation and Quay Crane Assignment Problem. In this problem, a number of cranes has to be assigned to each vessel in addition to a berthing time and position. The crane assignment is invariant in time, so the number of cranes assigned to a vessel cannot change during its handling. This version of the problem produces fewer crane changes between vessels than the variable-in-time crane assignment variant and therefore leads to more reliable berth plans.

Unlike other previously proposed formulations, neither the quay length nor the lengths of the vessels are discretized in the model; instead, a continuous variable for the berthing positions of the vessels is used, which makes the model truly continuous. Consequently, its performance does not depend on the discretization factor used for the lengths. Moreover, the initial formulation has been enhanced by adding several families of valid inequalities. As the computational study shows, the model has a stable behaviour, obtaining optimal or near-optimal solutions on different classes of instances with up to 50 vessels arriving within a time horizon of one week.

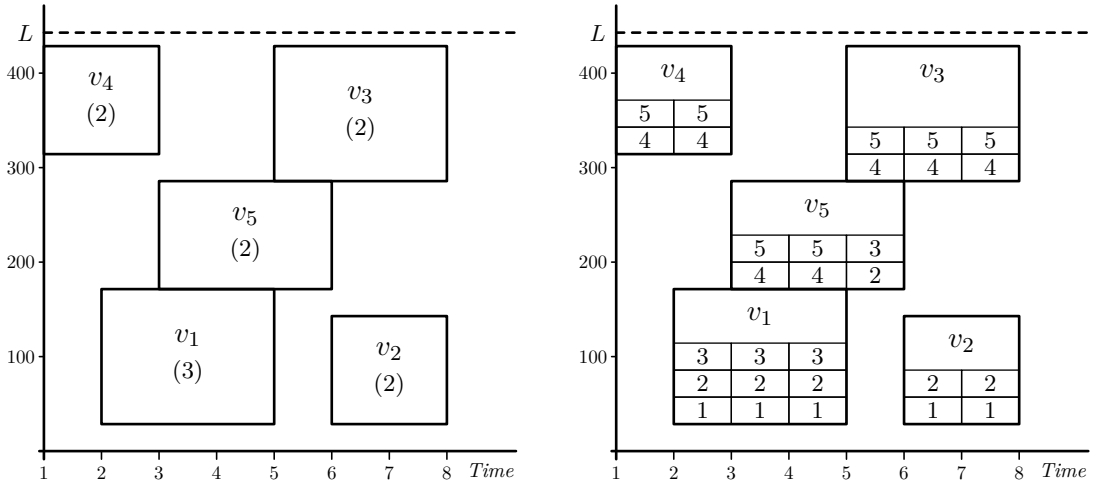
Chapter 5

New exact methods for the Berth Allocation and Specific Quay Crane Assignment Problem

5.1 Introduction

The previous chapter addressed the time-invariant Berth Allocation and Quay Crane Assignment Problem (BACAP). A feasible solution for this problem assigns a number of cranes to each vessel, guaranteeing that in any period the number of cranes being used does not exceed the number of cranes available on the quay. However, it does not specify which groups of cranes are to serve each vessel. When this assignment is considered jointly with the assignment of berthing time and position, taking into account that cranes cannot cross each other, a Berth Allocation and Specific Quay Crane Assignment (BACASP) arises.

Given a feasible solution for the time-invariant BACAP, a feasible solution for the BACASP can be obtained by using the dynamic programming algorithm proposed by Park and Kim (2003), assuming that the number of cranes determined for each vessel is kept fixed throughout its handling and cranes are able to change from one vessel to another in each time period. This algorithm assigns a group of cranes to each vessel and time period, minimizing the number of crane changes (Figure 5.1 illustrates this process). However, the condition stating that cranes can be changed between vessels in each time period implies that each vessel can be served by different groups of cranes during its handling, which may lead to unrealistic BACASP solutions. Indeed, as cranes cannot cross each other, in realistic-size instances the dynamic programming algorithm may produce a great number of crane changes in order to guarantee this condition, thereby affecting the real handling of vessels and making the berth plan unreliable.

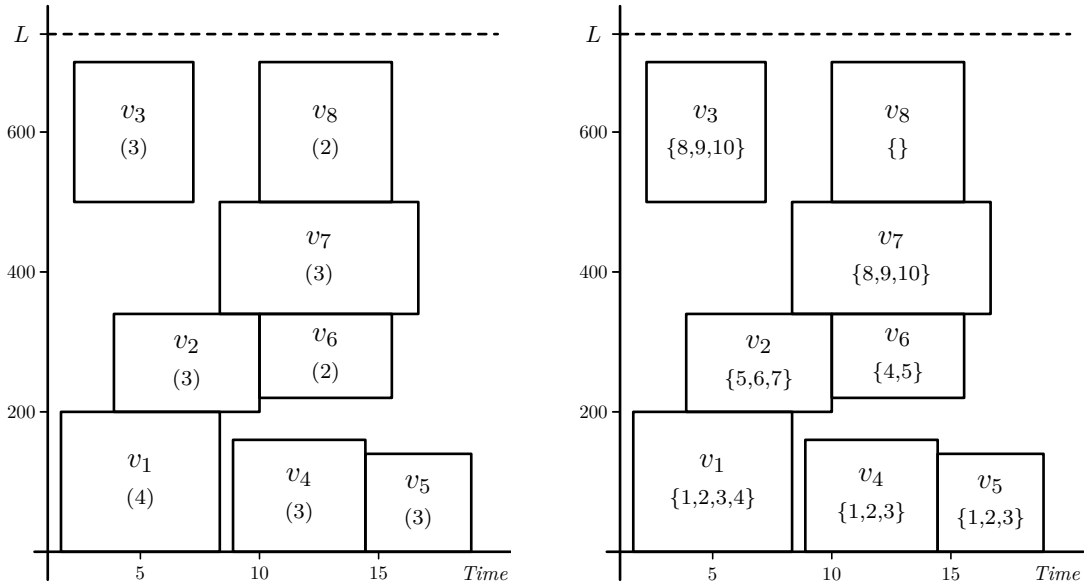


(a) Time-invariant BACAP solution for a terminal with 5 quay cranes. The number of cranes assigned is shown in parentheses.

(b) A BACASP solution generated for the BACAP solution. The cranes are numbered from 1 to 5 from the beginning of the quay and are represented as rectangles inside each vessel assignment.

Figure 5.1: A BACASP solution from a BACAP solution.

In order to prevent these shortcomings, in the time-invariant BACASP it is usually considered that the set of specific cranes assigned to each vessel is kept throughout its handling. In fact, this was assumed in all the papers relating to the time-invariant BACASP reviewed in the previous chapter, with the exception of those of Yang et al. (2012) and Rodriguez-Molins et al. (2014a). That assumption, however, poses a new problem. Given a time-invariant BACAP solution, now it does not ensure that a feasible assignment of specific cranes to vessels is possible. Figure 5.2 illustrates this situation. In Figure 5.2a, the number of the cranes assigned to each vessel appears in parentheses. If there are 10 cranes available, the solution shown is feasible for the BACAP. If we assume that the cranes are numbered from 1 to 10, ordered from the beginning of the quay, then the cranes can be assigned to vessels as shown in Figure 5.2b. Starting from the beginning of the planning period, cranes 1, 2, 3, and 4 are assigned to vessel v_1; cranes 5, 6, and 7 to vessel v_2; and cranes 8, 9, and 10 to v_3. When vessel v_7 starts being handled, v_3 has finished and so cranes 8, 9, and 10 can be assigned to v_7. Similarly, we can assign cranes 1, 2, and 3 to vessel v_4. When v_2 has finished, cranes 4 and 5 can be assigned to vessel v_6. However, no cranes can be assigned to vessel v_8. When it starts being handled, two cranes, 6 and 7, are available, but they cannot be moved to the position of v_8 without causing a disruption in the handling of v_7 and a reallocation of its cranes. Therefore, this time-invariant BACASP requires tailor-made solution methods.



(a) Feasible solution for BACAP. (b) Infeasible solution for BACASP.

Figure 5.2: A solution of the BACAP not feasible for the BACASP.

This chapter addresses the continuous time-invariant BACASP assuming that the set of specific cranes assigned to each vessel is not changed during its handling. First, this problem is tackled by extending the BACAP model proposed in the previous chapter. Then, an alternative method is proposed, based on an iterative algorithm that obtains BACASP solutions by introducing cutting planes on the BACAP model. The two methods are evaluated and compared with existing proposals in the literature through several computational experiments on various sets of instances.

The remainder of this chapter is organized as follows. Section 5.2 describes the problem and Section 5.3 presents the BACASP model. In Section 5.4 the cutting plane algorithm is proposed. Section 5.5 describes and discusses the experiments, and finally in Section 5.6 the contributions are summarized.

5.2 Problem description

The problem addressed here is exactly the same time-invariant BACASP tackled by Türkoğullari et al. (2014) and is an extension of the BACAP studied in the previous chapter. Now, in addition to the berthing time and position, a group of cranes is to be assigned to each vessel. In terms of the scheme proposed by Bierwirth and Meisel (2010), this problem can be classified as: $cont \,|\, dyn \,|\, QCAP \,|\, \sum(w_1\, pos + w_2\, wait + w_3\, tard)$ and BAP, QCAP.

5.2.1 Assumptions

Two new assumptions are introduced in addition to those considered for the BACAP already studied:

- A group of cranes must be assigned to each vessel taking into account that cranes are deployed on the same rail track and thus cannot cross each other when moving to and from vessels.
- The group of cranes assigned to a vessel is kept fixed, serving that vessel until the end of its handling.

5.2.2 Parameters

The following data about the terminal and the vessels are known in advance:

- Length of the quay: L
- Set of time periods: $T = \{1, 2, ..., H\}$, where H is the planning horizon.
- Set of quay cranes: $QC = \{1, 2, \ldots, Q\}$, where Q is the number of cranes. Cranes are numbered from 1 to Q starting from the beginning of the quay.
- Set of crane groups: QG. A *crane group* is defined as an ordered set of consecutive cranes, with cardinality varying from 1 to Q. Thus there will be Q groups composed of 1 crane; $Q-1$ groups of two cranes: $\{1,2\}, \{1,3\}, \ldots$; $Q-2$ groups of three cranes; and so on up to a final group composed of the Q cranes. QG is the set of these crane groups.

  Each group $g \in QG$ is defined by three parameters:

  - Number of cranes in the group: n_g
  - Number of the first crane in the group: f_g
  - Number of the last crane in the group: z_g
- Set of calling vessels: V, with $N = |V|$.

  For each vessel $i \in V$, the following data are known:

  - Length: l_i
  - Expected arrival time: a_i
  - Desired departure time: s_i
  - Minimum and maximum number of quay cranes that can be assigned to the vessel: q_i^{min}, q_i^{max}
  - Estimated handling time if it is handled using q cranes: h_i^q
  - Desired position on the quay: d_i

- Cost per period of waiting for berthing after the expected arrival time: C_i^w
- Cost per period of delay after the desired departure time: C_i^d
- Cost per length unit away from the desired position on the quay: C_i^p

From these parameters, for each vessel $i \in V$ the set of compatible crane groups can be precalculated as:

$$QG_i = \{g \in QG \,|\, q_i^{min} \leq n_g \leq q_i^{max}\}$$

5.3 A new mixed integer linear model

In this section, a new mixed integer linear model is proposed for the BACASP previously defined. This model is based on the BACAP model proposed in the previous chapter.

5.3.1 Variables

The variables are the same, except that now variables r_{igt} refer to crane groups instead of to numbers of cranes.

t_i = berthing time of vessel i

p_i = berthing position of vessel i

u_i = delay of vessel i

e_i = deviation of vessel i from its desired position on the quay

$$r_{igt} = \begin{cases} 1, & \text{if the handling of vessel } i \text{ with crane group } g \text{ starts at time } t \\ 0, & \text{otherwise} \end{cases}$$

$$\sigma_{ij} = \begin{cases} 1, & \text{if vessel } i \text{ is completely to the left of vessel } j \text{ in the space-time} \\ & \text{diagram, that is, vessel } i \text{ is completely handled before } j \\ 0, & \text{otherwise} \end{cases}$$

$$\delta_{ij} = \begin{cases} 1, & \text{if vessel } i \text{ is completely below vessel } j \text{ in the space-time diagram,} \\ & \text{that is, vessel } i \text{ is positioned completely to the right of vessel } j \\ & \text{looking at them from the quay} \\ 0, & \text{otherwise} \end{cases}$$

5.3.2 Objective and constraints

The objective function is also the same, and the constraints are easily adapted to consider the crane groups:

$$Min \sum_{i \in V} (C_i^w (t_i - a_i) + C_i^d u_i + C_i^p e_i) \tag{5.1}$$

s. t.

$$\sum_{t=a_i}^{H} \sum_{g \in QG_i} r_{igt} = 1, \qquad \forall i \in V \tag{5.2}$$

$$t_i = \sum_{t=a_i}^{H} \sum_{g \in QG_i} r_{igt} \cdot t, \qquad \forall i \in V \tag{5.3}$$

$$t_j - t_i - \sum_{t=a_i}^{H} \sum_{g \in QG_i} (r_{igt} \cdot h_i^{n_g}) - (\sigma_{ij} - 1)H \geq 0, \qquad \forall i, j \in V, i \neq j \tag{5.4}$$

$$p_j - (p_i + l_i) - (\delta_{ij} - 1)L \geq 0, \qquad \forall i, j \in V, i \neq j \tag{5.5}$$

$$\sigma_{ij} + \sigma_{ji} + \delta_{ij} + \delta_{ji} \geq 1, \qquad \forall i, j \in V, i \neq j \tag{5.6}$$

$$p_i + l_i \leq L + 1, \qquad \forall i \in V \tag{5.7}$$

$$\sum_{i \in V} \sum_{g \in QG_i} \sum_{\tau = \max(a_i, t - h_i^{n_g} + 1)}^{t} r_{ig\tau} \cdot n_g \leq Q, \qquad \forall t \in T \tag{5.8}$$

$$u_i \geq t_i - s_i + \sum_{t=a_i}^{H} \sum_{g \in QG_i} (r_{igt} \cdot h_i^{n_g}) - 1, \qquad \forall i \in V \tag{5.9}$$

$$e_i \geq p_i - d_i, \qquad \forall i \in V \tag{5.10}$$

$$e_i \geq d_i - p_i, \qquad \forall i \in V \tag{5.11}$$

$$\sigma_{ij}, \delta_{ij}, \in \{0, 1\}, \qquad \forall i, j \in V, i \neq j \tag{5.12}$$

$$r_{igt} \in \{0, 1\}, \qquad \forall i \in V, \forall g \in QG_i, \forall t \in \{a_i, \ldots, H\} \tag{5.13}$$

$$u_i, e_i \geq 0, \qquad \forall i \in V \tag{5.14}$$

$$p_i \geq 1, \qquad \forall i \in V \tag{5.15}$$

However, two more constraints have to be added:

- In each period t, a crane group g can be assigned at most to one vessel.

$$\sum_{i | g \in QG_i} \sum_{\tau = \max(a_i, t - h_i^{n_g} + 1)}^{t} r_{ig\tau} \leq 1, \qquad \forall t \in T, \forall g \in QG \tag{5.16}$$

- If vessel i is moored to the right of vessel j, looking at them from the quay, the numbers of the cranes assigned to i must be lower than the numbers of the cranes assigned to vessel j.

$$\sum_{t=a_j}^{H} \sum_{g \in QG_j} f_g r_{jgt} - \sum_{t=a_i}^{H} \sum_{g \in QG_i} z_g r_{igt} \geq 1 - Q(\delta_{ji} + \sigma_{ij} + \sigma_{ji}), \quad \forall i, j \in V, i \neq j \quad (5.17)$$

If $\delta_{ji} = 1$ or $\sigma_{ij} = 1$ or $\sigma_{ji} = 1$, the constraint is deactivated. But if $\delta_{ji} = \sigma_{ij} = \sigma_{ji} = 0$, then, by constraint (5.6), $\delta_{ij} = 1$, and vessel i is to the right of vessel j. In this case, the constraint ensures that the number of the first crane serving vessel j is greater than the number of the last crane serving vessel i.

5.4 An iterative procedure using the BACAP model

The BACASP model previously described is necessarily more complex than the BACAP model presented in the previous chapter, because instead of having an index q for each crane, variables r_{igt} have an index g for each crane group admissible for vessel i. This added complexity may limit the ability of the model to solve large-size problems. An alternative procedure for solving the BACASP is to use the BACAP model in an iterative way, adding constraints to this model if its solution is not feasible for the BACASP until a feasible and then optimal solution is found.

In the description of this new iterative procedure the following ideas proposed by Türkoğullari et al. (2014) are considered, adapting them to models proposed here. A vessel sequence $v_1, v_2, \ldots, v_n$ in a given feasible solution of the BACAP is *complete* if v_1 is the vessel closest to the beginning of the quay, v_n is the vessel closest to the end of the quay, v_i and v_{i+1} are two consecutive vessels with v_i closer to the beginning of the quay, and two consecutive vessels in the sequence concur at least for one time period. A complete sequence is said to be *proper* if the sum of the number of cranes assigned to vessels in this sequence is less than or equal to the number of cranes available on the quay. Otherwise, it is called an improper complete sequence. For example, in Figure 5.2b vessels v_4, v_6, v_7, and v_8 form a proper complete sequence, while vessels v_1, v_2, v_7, and v_8 form an improper sequence. It is easy to show that given an optimal solution of the BACAP, an optimal solution of the BACASP can be obtained from it in polynomial time if and only if every complete sequence of vessels extracted from the solution of the BACAP is proper. If a complete improper sequence $S = s_1, \ldots, s_n$ is found, the following constraint can be added to the BACAP model to prevent vessels in the sequence from being assigned so as to form an improper complete sequence in that specific order at any given time and position:

$$\sum_{i \in S} \sum_{t=a_i}^{H} \sum_{q=q_i^{min}}^{q_i^{max}} q \cdot r_{iqt} \leq Q + M \left(\sum_{j=s_1}^{s_{n-1}} (\sigma_{j,j+1} + \sigma_{j+1,j} - \delta_{j,j+1}) + n - 1 \right) \quad (5.18)$$

where: $M = \sum_{i \in S} q_i^{max} - Q$. If any pair of consecutive vessels in the sequence, v_j and v_{j+1}, are separated in space, v_{j+1} being above vessel v_j ($\delta_{j,j+1} = 1$), and are concurrent in time ($\sigma_{j,j+1} = \sigma_{j+1,j} = 0$), then the rightmost term in the expression takes value 0 and the assignment of a number of cranes to each vessel, represented by variables r_{iqt}, must satisfy the condition that their sum does not exceed Q. In any other case, the constraint is satisfied for any crane assignment.

Using this constraint, the cutting plane algorithm described in Algorithm 5 is proposed. If just one constraint is added for every improper sequence found, considering the vessels in the order in which they appear in this sequence, the solution of the next BACAP model very frequently includes the same set of vessels, but the order in which they appear in the sequence is changed. In order to prevent this situation, at least partially, instead of including just one constraint for each improper sequence, other permutations of vessels are also included. Nevertheless, as the cardinality of the sequence can be great in large instances, this is applied only if the cardinality of the sequence does not exceed a parameter $MaxSizeSeqForPerm$.

Algorithm 5 Cutting plane algorithm for the BACAP–BACASP

Input: P: the formulation of the BACAP model proposed in Chapter 4.4
Output: A solution for the BACAP without improper complete sequences

Solve P
while there is at least one improper complete sequence in an optimal solution of P **do**
  for all improper complete vessel sequence S found **do**
    Insert in P a cut for S using constraint (5.18)
    if $|S| \leq MaxSizeSeqForPerm$ **then**
      Insert in P such a cut for each other permutation of the vessels in S
    end if
  end for
  Solve P
end while

5.5 Computational experiments

The model and the cutting plane algorithm proposed were tested to assess their efficiency and limits. They were also compared with the cutting-plane algorithm proposed by Türkoğullari et al. (2014), based on their BACAP model, which is the exact approach for the time-invariant BACAP showing the best performance to date, according to the literature review presented in the previous chapter.

5.5.1 Test instances and implementation issues

The experiments were carried out over the same sets of instances used in the experiments described in the previous chapter: *GenPK*, *GenMB-10m*, and *GenMB-50m*. All the methods, including the cutting-plane algorithm proposed by Türkoğullari et al. (2014), were implemented in C++11 using GCC–G++ 6.2 and CPLEX 12.6, limiting the size of the search tree to 30 GiB. The compilation was performed specifying the special parameters *–O3 –march=corei7* in order to generate an executable file that makes the most of the processor. The experiments were run on an Intel Core i7 2600 at 3.4 GHz with 31.4 GiB of RAM, running the operating system Ubuntu 14.04 GNU/Linux 3.13. The cutting plane algorithm proposed was run considering $MaxSizeSeqForPerm = 4$, so only in the case of sequences with up to 4 vessels were the constraints corresponding to all the permutations included.

5.5.2 Results

Tables 5.1, 5.2, and 5.3 contain the results on sets *GenPK*, *GenMB-10m*, and *GenMB-50m*. For each subset of 10 instances of a given size, the tables show the number of instances for which at least an integer solution was found, the number of instances solved to optimality, the average computation time in seconds, and the average and maximum gap in percentage. The empty fields indicate that no integer solutions were found within the time limit and thus it was not possible to calculate the statistics. The integer model can fail to obtain an integer solution within the time limit due to the size of the model. The iterative procedures can fail because until they get to an optimal solution of the BACASP, the solutions obtained are not feasible and therefore the problem is not really solved. When using the model, if not all the instances of a subgroup have been solved, the gaps are calculated considering only those for which an integer solution has been found. In the case of the iterative procedures, there are no gaps, because either the optimal solution is found or there is no feasible solution. Table 5.1 does not show results for the cutting plane proposed by Türkoğullari et al. (2014) because the model on which it is based could not obtain even a feasible solution within the time limit to any instance, as we saw in the previous chapter.

Table 5.1: Comparing approaches to the BACASP on the set *GenPK*.

Vessels		MILP for the BACASP	Iterative procedure
20	Solved	10	10
	Optimum	10	10
	Avg. time	153.3	1.8
	Avg. gap	0	
	Max. gap	0	
25	Solved	10	10
	Optimum	10	10
	Avg. time	329.3	3.1
	Avg. gap	0	
	Max. gap	0	
30	Solved	10	10
	Optimum	10	10
	Avg. time	955.3	15.4
	Avg. gap	0	
	Max. gap	0	
35	Solved	10	9
	Optimum	6	9
	Avg. time	2090.5	640.6
	Avg. gap	32.0	
	Max. gap	98.8	
40	Solved	4	7
	Optimum	1	7
	Avg. time	3584.3	1516.2
	Avg. gap	61.5	
	Max. gap	86.8	

Table 5.1 shows that the two methods proposed were able to solve optimally all the instances with up to 30 vessels on the set *GenPK*. The iterative procedure was clearly faster and solved most instances with 35 and 40 vessels to optimality, while the model attained an optimal solution in 6 instances with 35 vessels and only one with 40 vessels, requiring more computation time on average.

In Table 5.2, the results on the set *GenMB-10m* show that the two methods proposed clearly outperformed the iterative procedure presented by Türkoğullari et al. (2014) both in number of instances solved to optimality and in computation time required. The cutting plane algorithm proposed solved to optimality all the instances with up to 40 vessels in less than 2 minutes and two with 50 vessels in less than one hour. The model was also able to solve most instances with up to 40 vessels optimally, but requiring much more time on average. Nevertheless, it could obtain feasible solutions for several instances with 50 and 60 vessels, unlike the iterative procedure.

Table 5.2: Comparing approaches to the BACASP on the set *GenMB-10m*.

Vessels		MILP for the BACASP	Iterative procedure	Iterative procedure by Türkoğullari et al.
20	Solved	10	10	9
	Optimum	10	10	9
	Avg. time	28.3	0.7	687.8
	Avg. gap	0		
	Max. gap	0		
30	Solved	10	10	6
	Optimum	9	10	6
	Avg. time	629.9	13.5	2077.3
	Avg. gap	3.6		
	Max. gap	36.1		
40	Solved	10	10	0
	Optimum	8	10	0
	Avg. time	1439.8	85.8	3600.0
	Avg. gap	4.0		
	Max. gap	22.3		
50	Solved	6	2	0
	Optimum	0	2	0
	Avg. time	3600.0	3292.1	3600.0
	Avg. gap	69.0		
	Max. gap	91.3		
60	Solved	3	0	0
	Optimum	0	0	0
	Avg. time	3600.0	3600.0	3600.0
	Avg. gap	91.9		
	Max. gap	94.8		

The results obtained by the methods on the set *GenMB-50m* (Table 5.3) show that none was clearly better than the others in terms of number of instances solved to optimality. The three methods seem able to solve instances with up to 40 vessels optimally. The iterative procedure proposed by Türkoğullari et al. (2014) also solved four instances with 50 vessels to optimality, two more than the iterative algorithm proposed. The model was again the slower method, although it could achieve feasible solutions in most instances with 50 vessels and in some with 60 vessels.

Table 5.3: Comparing approaches to the BACASP on the set *GenMB-50m*.

Vessels		MILP for the BACASP	Iterative procedure	Iterative procedure by Türkoğullari et al.
20	Solved	10	10	10
	Optimum	10	10	10
	Avg. time	26.4	0.6	18.9
	Avg. gap	0		
	Max. gap	0		
30	Solved	10	10	10
	Optimum	10	10	10
	Avg. time	617.8	19.6	288.9
	Avg. gap	0		
	Max. gap	0		
40	Solved	10	10	10
	Optimum	9	10	10
	Avg. time	1513.9	43.6	233.1
	Avg. gap	1.2		
	Max. gap	11.6		
50	Solved	8	2	4
	Optimum	0	2	4
	Avg. time	3600.0	3229.8	2732.5
	Avg. gap	65.0		
	Max. gap	82.4		
60	Solved	3	0	0
	Optimum	0	0	0
	Avg. time	3600.0	3600.0	3600.0
	Avg. gap	81.8		
	Max. gap	87.0		

In summary, the results show that the proposed model for the BACASP was able to solve optimally instances of set *GenPK* with up to 35 vessels and instances of sets *GenMB-10m* and *GenMB-50m* with up to 40 vessels. As this model is more complex than the model proposed for the BACAP in the previous chapter, the size of the instances optimally solved decreases. The proposed iterative procedure was faster than the model on the three test sets, although for instances in which the BACAP model was difficult to solve, very few iterations, if any, could be done within the time limit and almost

always the procedure failed to obtain a feasible solution. Comparing with the iterative procedure proposed by Türkoğullari et al. (2014) on sets *GenMB-10m* and *GenMB-50m*, for which their BACAP model was able to obtain solutions, it can be observed that their procedure works slightly better on set *GenMB-50m* and clearly worse on set *GenMB-10m*, basically repeating the behaviour of the BACAP models. Again, both the BACASP model and the iterative procedure proposed are very stable, regardless of the discretization factor used.

5.6 Concluding remarks

This chapter has addressed the time-invariant BACASP, that is, the problem in which, in addition to the berthing time and position, each vessel is to be assigned a set of cranes, taking into account that cranes cannot cross each other and the cranes assigned cannot be changed during the handling of the vessel. The solutions to this problem are thus more realistic and can be used in practical situations.

This problem has been tackled by means of a new mixed integer linear model and a new cutting plane algorithm based in the BACAP model proposed in the previous chapter. The computational experiments conducted show that the model is able to solve optimally instances of up to 35 vessels arriving within a time horizon of one week, while the cutting plane algorithm obtains optimal solutions on instances with up to 40 vessels in short computation times, thereby outperforming other state-of-the-art approaches. Both the model and the algorithm proposed show stable behaviour regardless of the discretization factor applied to the lengths of the quays and the vessels.

Chapter 6

A Biased Random-Key Genetic Algorithm for the Berth Allocation and Quay Crane Assignment Problem

6.1 Introduction

In the previous chapters we have seen that both the continuous time-invariant BACAP and BACASP are very hard problems. In one hour of computation time, the proposed exact methods are able to solve optimally instances with up to 40-50 vessels arriving within a time horizon of one week. The literature review also showed that there is a lack of studies dealing with larger instances of these problems. However, container ship traffic continues growing around the world and many terminals, especially in Asia, are required to serve more than 50 vessels each week. Moreover, as vessel congestion increases, the planning operations must be performed faster, so the available time for a berth planning cycle decreases. For these reasons, new fast methods not limited by the instance size are demanded to solve these problems, and thus heuristic algorithms arise as the best solution strategy.

This chapter presents a new Biased Random-Key Genetic Algorithm (BRKGA) with Memetic improvement and Local Search developed to obtain good solutions in short computation time on instances with up to 100 vessels arriving within a one-week time horizon. The initial heuristic is proposed for the BACAP already studied in Chapter 4. Then, in the next chapter, it will be adapted to be capable of solving the BACASP tackled in Chapter 5. All these methods are adjusted and evaluated through extensive computational experiments.

The remainder of this chapter is organized as follows. Section 6.2 describes the parameters and decision variables of the methods proposed. Next, Section 6.3 presents the Genetic Algorithm, the constructive algorithm, and several Local Search procedures. Section 6.4 describes the computational experiments conducted and discusses the results. Finally, Section 6.5 presents some concluding remarks.

6.2 Problem description

The BACAP addressed here is the same problem described in Chapter 4.3, differing only in the planning horizon, which now is not considered a hard constraint for the assignment of berthing time to vessels. Instead, it is considered as a desired berthing time horizon, so vessels can berth after that point, incurring an additional cost. Thus solutions with all the vessels moored within the time horizon are favoured. This makes it possible to tackle large instances without needing to assess whether they will have feasible solutions. This soft time horizon is now given by the constant E.

Consequently, each vessel has a new cost parameter. For each vessel i, C_i^e is the cost per time period exceeding E until the effective berthing time of the vessel. This cost is also added to the objective function. Taking the decision variables for each vessel i as the berthing time (t_i), the berthing position on the quay (p_i), and the number of cranes assigned (r_i), the objective function can be expressed as follows:

$$Min \sum_{i \in V} (C_i^w(t_i - a_i) + C_i^d(t_i + h_i^{r_i} - s_i - 1)^+ + C_i^p|p_i - d_i| + C_i^e(t_i - E)^+) \qquad (6.1)$$

where $(expression)^+$ is to be interpreted as $\max(expression, 0)$. This objective function is the same as that used for the BACAP model, except for the additional term considering the cost of exceeding the planning horizon E.

6.3 A Biased Random-Key Genetic Algorithm

As we have seen, this problem is hard to solve optimally for large instances, so the best option is to resort to metaheuristic strategies. Evolutionary Computation algorithms, especially Genetic Algorithms, have proved a good approach to tackle similar versions of the BACAP, as in the case addressed by Lalla-Ruiz et al. (2014) and other variants already examined in the literature review in Chapter 4.2. Following this line, this study proposes a new Biased Random-Key Genetic Algorithm with some Memetic characteristics, a constructive algorithm based on ordered lists of vessels, and several Local Search procedures. This kind of GA was first proposed by Bean (1994) and Ericsson and Pardalos (2002), and is characterized by the use of random keys to encode the genes. Genetic Algorithms can also be extended to consider multiple populations, thus allowing parallel evolutions with sporadic migrations of individuals between them. The multi-population scheme of the BRKGA developed here is inspired by Gonçalves and Resende (2012).

6.3.1 The Genetic Algorithm

There is a number N_{pop} of populations with $N_{indiv} = MultPop * N$ individuals each. That is, the number of individuals in a population depends on the number of vessels N and on a constant $MultPop$ that must be specified. Each individual has one chromosome consisting of two lists: a list of keys-to-vessels and a list of numbers of cranes-to-vessels. Each position in these lists corresponds, respectively, to the key and the number of cranes assigned to the vessel with that index. The key is a number between 0 and 1,

while the number of cranes must be between the minimum and the maximum number of cranes allowed for that vessel (Figure 6.1). The effective list of vessels is determined by decoding the list of random keys-to-vessels, which is done by sorting the keys in non-decreasing order and then taking their indexes (Figure 6.2).

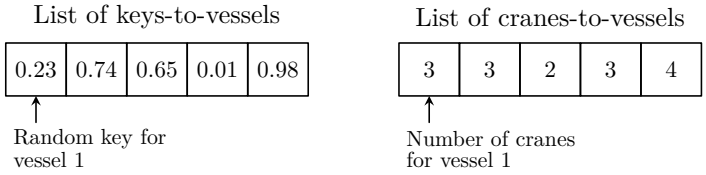


Figure 6.1: Chromosome of an individual for an instance with five vessels.

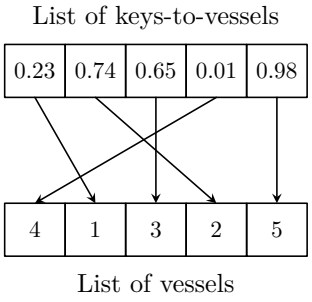


Figure 6.2: Decoding a list of keys-to-vessels to a list of vessels.

The cranes-to-vessels list and the sequence of vessels obtained by sorting the keys represent a solution as long as they are considered together with the constructive algorithm (Section 6.3.2). The constructive algorithm uses this information to generate a feasible solution by adding the vessels one by one to the berth plan. After the construction, a refinement procedure is applied, aiming at improving the solution by reducing the idle time of the cranes to a minimum (Section 6.3.3). It is thereby possible to obtain a new list of cranes which, used together with the list of vessels, may produce a better solution. If this is the case, the chromosome is updated with the new list and can influence the offspring. This inheritance of acquired characteristics turns the Genetic Algorithm into a kind of *Memetic Algorithm*.

The fitness of an individual is the negative of its objective function value, since it is a minimization problem, and the genetic operators are *elite crossover*, *immigration* and *migration*. The crossover works between two parents, considering a bias $CrossBias \in [0, 1]$ in favour of the chromosome from the first parent, which is taken from the elite (Figure 6.3). The same process is applied independently over the list of cranes-to-vessels. Besides the crossover, at each generation new randomly generated individuals are included as a kind of immigration. Moreover, a migration is performed every $GensMig$ generations, adding the best individual among all the populations into the other populations in which it is not present and removing the worst individual to keep the same population size.

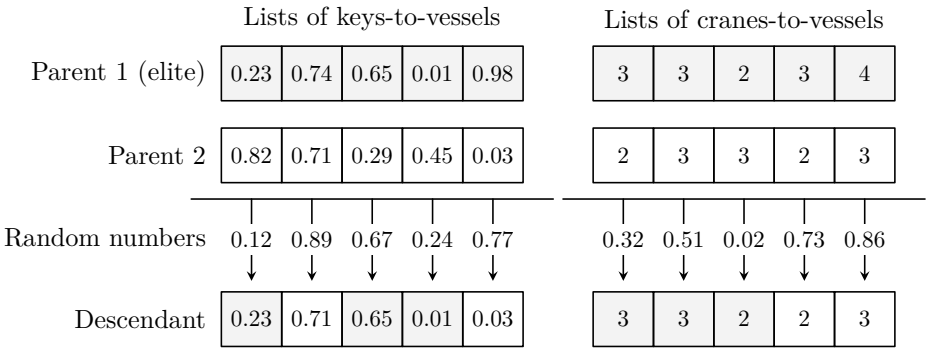


Figure 6.3: Example of a crossover. A gene from the first parent passes to the descendant if the random number is less than $CrossBias = 0.7$.

The initial populations consist of individuals whose keys-to-vessels lists are generated according to several priority rules (Table 6.1). This provides the evolutionary process with information that may help the search, as we saw in Chapter 3. The crane lists are generated by assigning a random number of cranes to each vessel from among its allowed values. This ensures that the descendants' crane lists will be valid too. As the population size can be greater than the number of priority rules, the remaining individuals in each population are generated randomly, thereby introducing more diversity.

Table 6.1: Priority rules used to generate ordered lists of vessel keys.

Name	Sorting criterion	
Arrival time (i)	Non-decreasing	a_i
Arrival time (ii)	Non-increasing	a_i
Length (i)	Non-decreasing	l_i
Length (ii)	Non-increasing	l_i
Desired departure (i)	Non-decreasing	s_i
Desired departure (ii)	Non-increasing	s_i
Max. handling time (i)	Non-decreasing	$h_i^{q_i^{min}}$
Max. handling time (ii)	Non-increasing	$h_i^{q_i^{min}}$
Min. handling time (i)	Non-decreasing	$h_i^{q_i^{max}}$
Min. handling time (ii)	Non-increasing	$h_i^{q_i^{max}}$
Max. area (i)	Non-decreasing	$l_i h_i^{q_i^{min}}$
Max. area (ii)	Non-increasing	$l_i h_i^{q_i^{min}}$
Min. area (i)	Non-decreasing	$l_i h_i^{q_i^{max}}$
Min. area (ii)	Non-increasing	$l_i h_i^{q_i^{max}}$
Max. slack (i)	Non-decreasing	$s_i - (a_i + h_i^{q_i^{min}})$
Max. slack (ii)	Non-increasing	$s_i - (a_i + h_i^{q_i^{min}})$
Min. slack (i)	Non-decreasing	$s_i - (a_i + h_i^{q_i^{max}})$
Min. slack (ii)	Non-increasing	$s_i - (a_i + h_i^{q_i^{max}})$

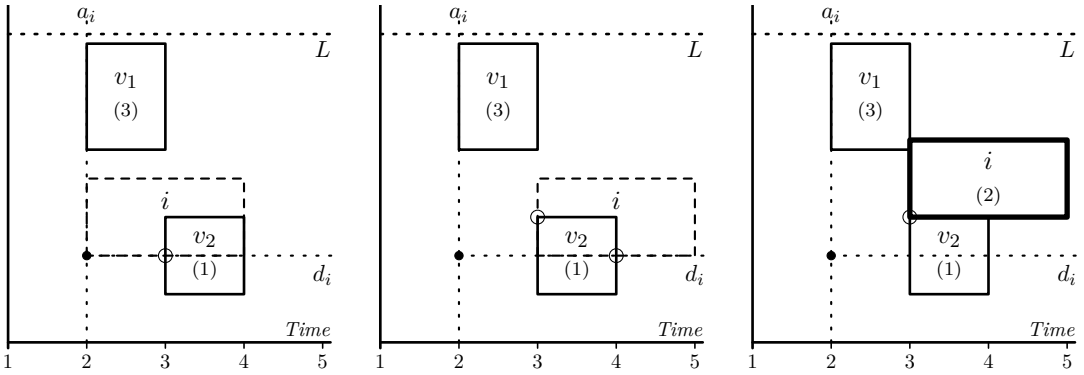
The process applied to each population to generate a new generation is as follows. First, the constructive algorithm is applied to each individual in the population to obtain its corresponding feasible solution. Then, individuals are sorted according to their fitness and the best $E\%$ individuals are marked as the elite. The following generation will be composed of this elite, $I\%$ new individuals randomly generated as immigration, and $100\% - E\% - I\%$ individuals resulting from the crossover between pairs of individuals randomly chosen, the first one being from the elite and the other from the entire population. The parent from the elite is the one favoured by the bias. Each generation is completed for all the populations before passing to the next generation. Migration is performed only every $GensMig$ generations. This process is repeated until a time limit of $TimeLimitGen$ seconds. Afterwards, a Local Search procedure is applied to every individual in each population until a stopping criterion is met, with the aim of refining the best solutions obtained by the Genetic Algorithm (Section 6.3.4).

6.3.2 Constructive algorithm

An individual represents a solution only if it is considered together with a constructive algorithm. This algorithm builds a feasible berth plan from the list of cranes-to-vessels and the list of vessels resulting from sorting the keys. The algorithm developed is based on the *Exploratory Constructive Algorithm (ECA)* proposed in Chapter 3.5.2.1, with the novelty that now it also assigns a number of cranes to each vessel and checks each time that the number of cranes assigned to vessels does not exceed Q.

First, the number of cranes corresponding to each vessel according to the cranes-to-vessels list is kept fixed. Therefore, as the handling time of each vessel is known, a berthing time and position can be assigned to each vessel, taking them sequentially from the ordered list of vessels. The objective is to achieve the best allocation for each vessel in a berth plan in which the previous vessels in the list have already been assigned. To do this, each time a vessel is extracted from the list, a set K of candidate assignments ordered by non-decreasing cost is defined. The set K is filled and explored throughout the process, always extracting the least-cost candidate. The first candidate assignment included in K is the one consisting of the desired position and the arrival time of the vessel.

Once the least-cost candidate has been extracted from K, if the vessel cannot be assigned to that position and time because it would not have enough cranes throughout its handling, a new candidate consisting of the same position and the earliest berthing time in which there are enough cranes is included in K (Figure 6.4(a)). If the vessel at that time and position would overlap with one or more existing vessels in the space-time layout, the best candidate assignments in their contour are included in K (Figure 6.4(b)). Then the next least-cost candidate in K is extracted and the process is repeated until a feasible assignment is found (Figure 6.4(c)). This is done for all the vessels in the list until the berth plan is completed.



(a) Vessel i tries to berth at (a_i, d_i). Vessel i requires 2 cranes and the quay has only 3 cranes, so a new assignment with enough cranes (empty circle) is identified.

(b) In the new assignment, vessel i would overlap with v_2, so new candidates in the contour of v_2 (empty circles) are identified.

(c) Vessel i is assigned to the location of the least-cost feasible candidate.

Figure 6.4: Process of the Exploratory Constructive Algorithm for the BACAP.

6.3.3 Memetic improvement

Once the constructive algorithm has obtained a solution, it is examined looking for idle cranes, because it may be possible that the crane-to-vessel assignments do not require all Q cranes in each period. Thus vessels are examined in non-decreasing departure time order, and if a crane is idle throughout the handling of a vessel, it is assigned to it. The reduction in handling time thus achieved can reduce or even eliminate the delay incurred by the vessel, and so its corresponding cost.

It is possible to attain a further improvement if the constructive algorithm is now run with the new list of cranes and the same list of vessels. The reduction in handling times previously obtained may now allow other vessels to be berthed earlier. Therefore, the constructive algorithm is applied again and if a better solution is obtained, the chromosome is updated with the refined list of cranes. This process is repeated until no further improvement is achieved. The kind of Memetic scheme thus performed is known in the literature as a *Lamarckian Memetic Algorithm*, since the local improvements are reflected in the chromosome and can therefore be inherited by the offspring (Le et al., 2009).

6.3.4 Local Search procedures

The Genetic Algorithm does not explore the entire solution space, but rather a subspace defined by the way in which the constructive algorithm generates feasible solutions following a list of vessels sequentially. In order to reach solutions that cannot be reached when assigning all the vessels one by one, three different Local Search procedures based on directly moving vessels in an existing berth plan are proposed: two heuristics and one matheuristic. These LS procedures are used only as a final attempt to refine the

best solutions attained by the BRKGA, so they are applied to all the individuals in each population as soon as the genetic process has ended. These individuals are sorted by non-increasing value, and starting from the best of them, the procedure is applied until the stopping criterion, a time limit of $TimeLimitLS$ seconds, is met.

6.3.4.1 Simple ruin-and-recreate heuristic

The first heuristic is based on the ruin-and-recreate strategy, so a set of vessels is removed from the solution and then the constructive algorithm is used to reassign them.

This procedure first generates a list of vessels ordered by non-increasing score, which is, for each vessel, the sum of its cost and the cost of the vessels belonging to its cluster, determined by a neighbourhood function applied to the vessel. Then a vessel is selected from the list with a probability $SelProb$, starting from the first element. The selected vessel and the vessels in its cluster are removed from the solution and reassigned in non-increasing cost order by means of the constructive algorithm, using the list of crane assignments of the original solution together with the refinement seen in Section 6.3.3, but without performing the feedback phase (Figure 6.5).

Additional solutions are generated by changing the number of cranes assigned to those vessels. A *Path Relinking* is performed on the list of cranes, keeping the list of vessels to be reassigned unchanged. The Path Relinking technique generates intermediate solutions in the trajectory between a *initiating solution* and a *guiding solution* by systematically changing elements of the initiating solution, step by step, until the guiding solution is reached. The initiating list is the original list of cranes, while the guiding list is selected between two alternatives: the list with the maximum number of cranes allowed for each of those vessels and the list with the minimum number of cranes. These candidates are considered ideal as guiding lists because they lead to extreme solutions. The constructive algorithm builds the solution corresponding to each list and the list leading to the best solution is selected.

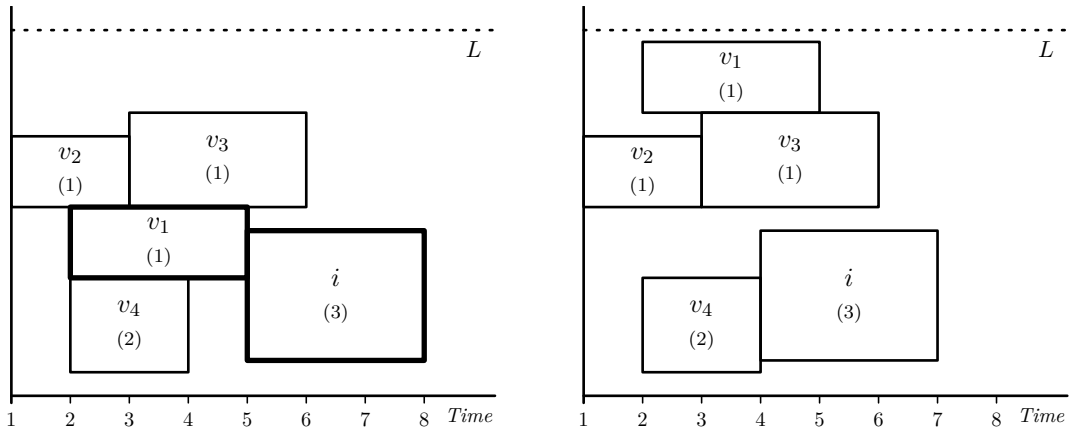
Once a guiding solution has been chosen, the list of cranes at each step results from changing one crane-to-vessel assignment, replacing it with the number of cranes in the guiding list. Hence, a new solution is obtained for each new list of cranes. A number $Paths$ of different paths between the initiating and the guiding lists is thus generated. For the first path, the sequence of indexes that must be changed corresponds to the order in the list of vessels, while for the rest it is determined randomly.

The best solution obtained is stored, and if it is not better than the original one the whole process is repeated with the next vessel in the list of vessels ordered by score. The number of tries is limited by a parameter $MaxTries$. Otherwise, if the solution is better, the whole process is repeated on the new best solution until no further improvement is achieved or until the stopping criterion is met.

The following neighbourhood functions, which lead to different variants of the procedure, were defined:

- *Overlapping vessels.* Given a vessel i, this function returns the vessels that would overlap with i if it were moved to its ideal location, that is, the one consisting of its desired position on the quay and its expected arrival time.

- *Adjacent vessels.* Given a vessel i, this returns the vessels adjacent to i: the vessels whose rectangles are touching the rectangle representing i in the graphic representation of the solution.
- *Connected component of vessel.* Given a vessel i, this returns the vessels adjacent to i and, recursively, all the vessels that are adjacent to them. In other words: if the vessels are considered as the vertices of a graph, connected by edges representing adjacency relations, this function returns the connected component to which i belongs.



(a) Vessel i is removed together with its adjacent vessel v_1. The quay has 5 cranes.

(b) Vessels i and v_1 are reassigned and several configurations of cranes are tested. This is a possible output.

Figure 6.5: Simple ruin-and-recreate LS with the adjacency neighbourhood function.

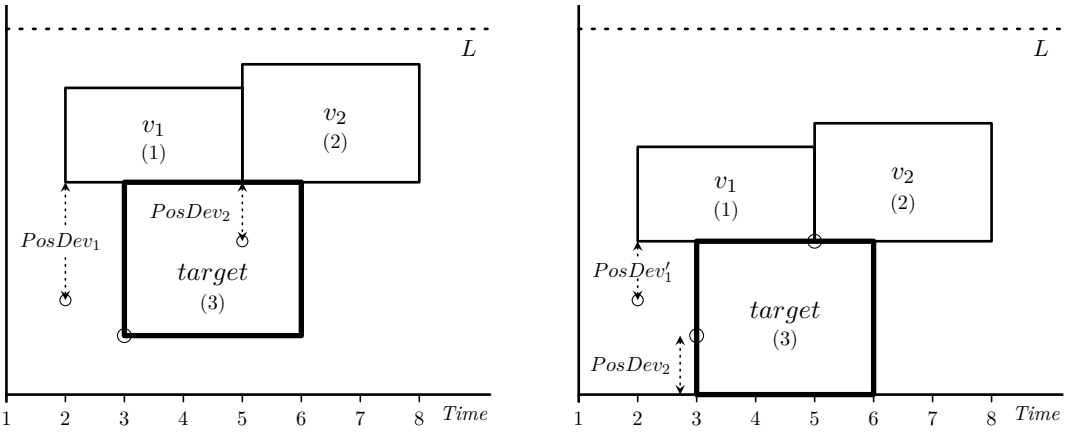
6.3.4.2 Pushing ruin-and-recreate heuristic

This heuristic also removes a cluster of vessels, but before reassigning them, the ideal berthing time and position of one vessel are modified, so the constructive algorithm will try to put it at this new location. Looking at the solutions obtained by the BRKGA, a case was noticed in which the optimal solution could not be reached due to the way in which the constructive algorithm builds the solutions. When assigning berthing times and positions to the vessels in a cluster, the first vessel was put with no cost in its ideal assignment, at the expense of increasing the cost of all the other vessels. The optimal solution in those cases was reached by moving that vessel to a new position and/or time, increasing its cost, but allowing the other vessels to be better placed so that the overall cost of the cluster was reduced. The best way to achieve this cluster reassignment would be to apply a mathematical programming model to the cluster, as we will see in the next subsection; however, the model can only be applied to small clusters. An alternative is to *push* the first vessel to a new location and relocate the remaining vessels using the constructive algorithm. This requires determining heuristically the extent of the movement applied to that vessel, so different values are tested by considering the other vessels' deviations from their ideal assignments. In particular,

a total of 15 movements are tested, resulting from combining the maximum, minimum and average deviations in time ($maxTimeDev$, $minTimeDev$, $avgTimeDev$, respectively) and position ($maxPosDev$, $minPosDev$, $avgPosDev$, respectively) and the null displacement. Each pair $(timeMove, posMove)$ in the following list represents a movement, where $timeMove$ is the number of units moved in time and $posMove$ the number of units moved in space:

$$
\begin{aligned}
&(0, minPosDev), (0, avgPosDev), (0, maxPosDev),\\
&(minTimeDev, 0), (minTimeDev, minPosDev),\\
&(minTimeDev, avgPosDev), (minTimeDev, maxPosDev),\\
&(avgTimeDev, 0), (avgTimeDev, minPosDev),\\
&(avgTimeDev, avgPosDev), (avgTimeDev, maxPosDev)\\
&(maxTimeDev, 0)(maxTimeDev, minPosDev),\\
&(maxTimeDev, avgPosDev), (maxTimeDev, maxPosDev).
\end{aligned}
$$

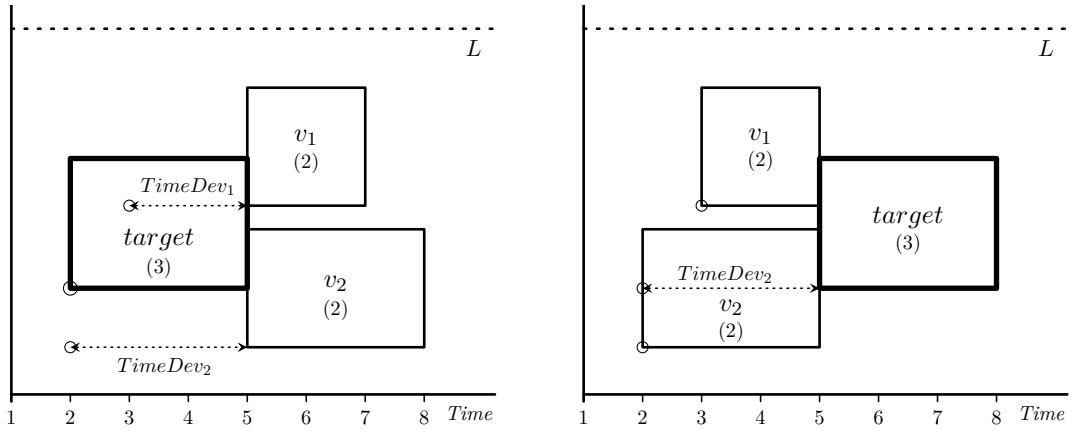
Let $originalPos$ and $originalTime$ be the original position and time of the target vessel in the solution, respectively. For each movement in the list, the target vessel is included in the solution by means of the constructive algorithm, considering that now the desired position is $originalPos - posMove$ and the desired time is $originalTime + timeMove$. The displacement in space moves the vessel contrary to the deviation calculated from the other vessels' deviations, hence the negative sign (Figure 6.6). By contrast, the displacement in time is performed in the same direction because the target vessel is frequently located at its arrival time. This allows other vessels to be assigned prior to it (Figure 6.7).



(a) The target vessel is in its ideal assignment. Vessels v_1 and v_2 are assigned to their arrival time, but deviate from their desired positions.

(b) The movement $(0, minPosDev)$ is applied to *target*. The other vessels are reassigned in order of non-increasing cost. Thus the overall cost is reduced.

Figure 6.6: Pushing ruin-and-recreate LS with the clustering neighbourhood function. Example of a position movement applied to the target vessel.



(a) The target vessel is in its ideal assignment. Vessels v_1 and v_2 are assigned to their desired positions on the quay, but deviate from their arrival time.

(b) The movement $(maxTimeDev, 0)$ is applied to *target*. The other vessels are reassigned in order of non-increasing cost. Thus the overall cost is reduced.

Figure 6.7: Pushing ruin-and-recreate LS with the clustering neighbourhood function. Example of a time movement applied to the target vessel.

6.3.4.3 Matheuristic

An alternative method for improving the cluster reassignment is to use a mathematical programming model. In this case, a modified version of the mixed integer linear model proposed in Chapter 4.4 is used.

The initial steps of the procedure are similar to those of the previous one, but a cluster is selected only if its size is lower than or equal to a parameter *MaxClusterSize*. The vessels in the cluster are completely removed from the solution and the model is used to relocate them. The model is constructed by inserting additional constraints to keep the values of berthing time and position variables fixed for the vessels not in the cluster. The only variables that the solver can change are those corresponding to the vessels in the cluster, so the size of the problem to be solved depends on the cluster size. This makes the method very useful for small clusters, given that a model with few vessels can be rapidly solved to optimality, as reported in previous chapters. Since in the model vessels are not able to berth after the time period H, it is constructed with a new H' equal to the maximum between the original H and the maximum berthing time among the vessels in the solution: $H' = \max\{H, \max_{i \in V}(t_i)\}$. Moreover, in order to consider the new cost of berthing after E, for each vessel $i \in V$ a new variable exc_i and the following constraints are introduced:

$$exc_i \geq t_i - E \tag{6.2}$$

$$exc_i \geq 0 \tag{6.3}$$

Thus, the objective function is now:

$$Min \sum_{i \in V} (C_i^w (t_i - a_i) + C_i^d u_i + C_i^p e_i + C_i^e exc_i) \tag{6.4}$$

The model is run on a solver without including the extra valid inequalities, providing the value of the current solution as an upper bound to reduce the computation time. The solution obtained is then compared with the current best solution obtained in the Local Search process. If it is better, then the Local Search procedure is applied to the next solution. If not, a new vessel with its corresponding cluster is selected according to its score. Since the stopping criterion is a time limit, it may occur that neither optimal nor feasible solutions are found.

6.4 Computational experiments

Several comprehensive computational experiments were conducted over different sets of instances in order to adjust the parameters of the Genetic Algorithm and the Local Search procedures and evaluate their performance. All of them were run on an Intel Core i7 2600 at 3.4 GHz with 31.4 GiB of RAM. The algorithms were implemented in C++11 and OpenMP was applied to evaluate as many individuals as possible in parallel threads, not affecting the algorithms but in the running time. The operating system used was Ubuntu 14.04 GNU/Linux 3.13 and the compiler used was GCC–G++ 6.2. The compilation was performed specifying the special parameters *–O3 –march=corei7* in order to generate an executable file that makes the most of the processor. The model was implemented in C++11 and CPLEX 12.6, limiting the maximum size of the search tree to 30 GiB. The statistical analysis was performed using the programming language R 3.3 and package *ez* 4.4.

The sets of instances used were *GenPK* and *GenMB-10m*, already described in Chapter 4.6. The set *GenMB-10m* was extended with instances generated in the same way with 70, 80, 90, and 100 vessels. In total, the set *GenPK* consisted of the original 50 instances, and the set *GenMB-10m* was composed of 90 instances. The new parameters considered were set as follows. In *GenPK*, $E = H = 300$ hours, while in *GenMB-10m*, $E = H = 210$ hours (125% of 168 h) in instances with up to 70 vessels, and $E = H = 252$ hours (150% of 168 h) in instances with more than 70 vessels. In both sets, for all the vessels, $C_i^e = 5000$. In order to perform a correct adjustment of the algorithms, a training set of instances, called *GenAdjust*, was generated applying the same criteria as in the set *GenMB-10m*, since it is more realistic than *GenPK*. The set *GenAdjust* consists of 45 instances in total: 5 for each number of vessels considered.

Three different experiments were conducted. The first was aimed at obtaining a good configuration of the parameters of the Genetic Algorithm without performing Local Search (Section 6.4.1). The second experiment was conducted to determine a good parameter configuration for each Local Search algorithm (Section 6.4.2). The results obtained made it possible to select the best Local Search procedure (Section 6.4.3). Finally, in the third experiment the performance of the entire algorithm was evaluated,

comparing its results with the best solutions achieved by the BACAP model proposed in Chapter 4.4 on both sets of instances (Section 6.4.4).

6.4.1 Parameter adjustment of the Genetic Algorithm

6.4.1.1 Description

The objective of the first experiment was to determine the best values of the parameters of the Genetic Algorithm. The parameters $E = 15\%$, $I = 15\%$, and $CrossBias = 0.7$ were set, since they were considered reasonable values that produced good results in previous studies (Gonçalves and Resende, 2012) and preliminary experiments. In these preliminary experiments it was also noticed that the most influential parameters seemed to be: $MultPop$, N_{pop}, and $GensMig$. Hence it was decided to test different values for them: $MultPop \in \{1, 10, 20\}$, $N_{pop} \in \{1, 10, 20\}$, and $GensMig \in \{10, 30\}$. This resulted in a total of 15 combinations. The time limit was set to $TimeLimitGen = N$ seconds, so that the computation time for each instance depended on its size.

For each parameter configuration, five independent random repetitions of the Genetic Algorithm were run on each instance in *GenAdjust*. For each repetition, the value of the best solution achieved and the running time were recorded. Then the deviation percentage (DFB) of the result obtained in each repetition ($value$) was calculated from the minimum value attained among the repetitions of all the configurations tested on the same instance ($minValue$): $DFB = 100 \cdot (value - minValue)/minValue$.

In order to determine whether the differences in DFB were statistically significant, a *repeated measures design* was conducted, taking the *average* DFB obtained for each instance from its five repetitions as the dependent variable. Since each configuration was run over the same instances, the *instances* were the subjects of the analysis, while $configuration$ was considered a within-subjects fixed factor with 15 levels corresponding to the different configurations studied. Thus the 15 DFB means obtained for each instance were considered different measures on the same subject. Moreover, as the performance of the algorithm may be different for different problem sizes, the *number of vessels* in the instance was considered as a between-subjects fixed factor, with 9 levels (20–100 vessels). Including these factors allows the statistical analysis to manage the variability they introduce, thereby increasing its sensitivity. The significance level was set to $\alpha = 0.05$ for all the statistical tests.

The repeated measures design can be tackled following either an univariate or a multivariate analysis of variance (ANOVA). The multivariate approach (MANOVA) is sometimes recommended because, unlike the univariate analysis, it does not assume multisample sphericity; however, it assumes multivariate normality, which is clearly not satisfied by the instances used according to the results of Mardian's test. For this reason, the univariate alternative was performed, also known as *mixed-design* ANOVA, whose assumptions result from the combination of both between-subjects and within-subjects designs. According to Maxwell and Delaney (2004), with respect to the between-subjects factor: (i) the observations must be independent between subjects and in each population defined by this factor (ii) the mean score for each subject, averaged over the levels of the within-factor, must be normally distributed. Moreover, (iii) its variance has to be

the same in all those populations. Regarding the within-subjects factor, in each group defined by the combinations of the within-subjects and between-subjects factors, (iv) the dependent variable must be normally distributed and (v) the covariance matrix has to be spherical. Furthermore, (vi) these covariance matrices must be homogeneous across the different levels of the between-subjects factor. Violations of the normality assumptions have little impact and violations of the homogeneity of variances/covariances are not serious when the sample sizes are equal, which they are in this case. However, the tests are not robust to violations of the sphericity assumption, but this issue can be easily overcome by applying the Greenhouse-Geisser correction on the degrees of freedom.

These assumptions of validity were checked on each experiment and considered acceptable. The Greenhouse-Geisser correction was applied when the sphericity assumption did not hold.

6.4.1.2 Results

The ANOVA tested the effect of the *configuration*, the effect of the *number of vessels* and the *interaction* between them. All these effects proved to be statistically significant, with $p < 0.0001$ in all of them and with η^2 (calculated as $SumSquares_{\text{effect}}/SumSquares_{\text{total}}$) equal to 0.134, 0.186, and 0.457, respectively. These results show that the configurations tested differ in their DFB both on its own and jointly with the size of the problem. The effect that explained the largest proportion of the sample DFB variability was the *number of vessels*, reaching 45.7% according to its η^2, followed by the *interaction* and the *configuration*. This indicates that regardless of the algorithm configuration applied, the size of the instance has a strong impact on the performance, while the configurations are less influential but also important, taken alone or jointly with the size of the instance. Such evidence is consistent with expectations, given that heuristics also suffer the effects of the combinatorial explosion, although they can withstand it better than solvers implementing mathematical programming models.

Given that in this experiment the main objective was to determine which *configuration* is the best regardless of the size of the instance and within the margins of the study, the Holm-Bonferroni multiple comparison tests were conducted for this effect alone, specifying that they were paired samples. Thus all the groups of configurations within which their average DFB cannot be considered statistically different were calculated. Table 6.2 shows the average number of generations done and the mean and standard deviation of DFB for each configuration, together with the homogeneous groups. According to the results, configurations with the highest total number of individuals ($N \cdot MultPop \cdot N_{pop}$) show a performance clearly worse than the rest. Examining the data it was observed that in those configurations the DFB in instances with more than 50 vessels was higher than in any other configuration. This is due to the small number of generations performed. Indeed, the configurations that obtained the best DFB results completed more than 24 times the number of generations completed by the worst. Therefore, performing more generations is preferable to increasing the number of populations and individuals therein. Apart from this, the $GensMig$ values tested do not seem to make a significant and consistent difference. The configurations of group B are those with the lowest

means, so for the rest of the study, the one with the lowest sample average among them was selected: $MultiPop = 1$, $N_{pop} = 20$, and $GensMig = 30$.

Table 6.2: Results of the BACAP Genetic Algorithm on the set *GenAdjust*.

Configuration			Generations	DFB (%)		Group		
$MultPop$	N_{pop}	$GensMig$	Mean	Mean	Std. Dev.	A	B	C
1	1	10	30612.8	7.29	6.60	×		
1	10	10	3177.3	5.18	4.30		×	
1	10	30	3180.4	4.96	4.41		×	
1	20	10	1576.1	4.76	4.17		×	
1	20	30	1585.7	4.44	3.89		×	
10	1	10	3017.0	5.89	6.22	×	×	
10	10	10	283.9	5.10	5.32		×	
10	10	30	284.6	5.33	5.82		×	
10	20	10	139.3	8.36	8.48	×		
10	20	30	140.8	8.28	8.45	×		
20	1	10	1420.5	5.77	6.47	×	×	
20	10	10	130.7	8.99	8.92	×		
20	10	30	131.8	8.99	8.87	×		
20	20	10	64.4	14.30	13.90			×
20	20	30	64.7	14.18	13.68			×

6.4.2 Parameter adjustment of the Local Search procedures

6.4.2.1 Description

After determining a proper configuration for each Genetic Algorithm, analogous experiments were conducted to adjust the parameters of the Local Search procedures. In particular, the following algorithms were evaluated: the simple ruin-and-recreate heuristic with the overlap neighbourhood function (*SimpleOverlap*), with the adjacency function (*SimpleAdjacency*), and with the connected component function (*SimpleConnected*); the pushing ruin-and-recreate heuristic with the connected component function (*Pushing*); and the matheuristic with the adjacency function (*ModelAdjacency*) and with the connected component function (*ModelConnected*). The pushing heuristic was used only with the connected component function due to its very design, which is tailored to it. As for the matheuristic, the overlap function was not considered because it performed worse in preliminary experiments.

An overall time limit of one second per vessel was set in each instance (N), of which $TimeLimitGen = 0.9N$ and $TimeLimitLS = 0.1N$ seconds. In preliminary experiments it was noticed that all the algorithms except the matheuristic were able to finish within this time limit considering $MaxTries = N$, hence that value was fixed for them. For the other parameters of the algorithms all the combinations resulting from the following values were tested: $SelProb \in \{0.5, 0.75, 1\}$ and $Paths \in \{30, 45, 60\}$. For

the matheuristic, $MaxClusterSize = 15$ was kept fixed, as it was observed that greater values made the matheuristic so slow that it could not solve many models. All the combinations of $SelProb \in \{0.5, 0.75, 1\}$ and $MaxTries \in \{1, 5, 10\}$ were tested. This resulted in a total of 9 different configurations for each LS procedure.

A single repetition of the experiment consisted of running the Genetic Algorithm on each instance and applying each LS algorithm with each configuration independently to its result. In this way we could properly compare their performance, since all of them were applied to the set of individuals resulting from the same run of the Genetic Algorithm.

6.4.2.2 Results

Six separate ANOVAs were conducted, one for each Local Search procedure, in order to study the effect of the parameters, to identify homogeneous groups, and finally to determine the best configuration of each algorithm. Table 6.3 summarizes the results of the ANOVAs, showing the p value and the η^2 for each effect. According to these results, the effect of the *number of vessels* alone was statistically significant in all the algorithms, the effect of the interaction between *number of vessels* and *configuration* was significant in *SimpleOverlap* and *ModelAdjacency*, and the effect of *configuration* alone was significant in *SimpleAdjacency*, *ModelAdjacency*, and *ModelConnected*. The sample variability was highly influenced by the *number of vessels*, while it was less affected by *interaction* and *configuration* effects. Nevertheless, in some algorithms they were sufficient to make a difference between the configurations tested, as the p value of the ANOVA shows.

Table 6.3: Results of the ANOVA and η^2 calculations for each Local Search algorithm in the adjustment experiment.

Algorithm	Interaction		No. of vessels		Configuration	
	p value	η^2	p value	η^2	p value	η^2
SimpleOverlap	0.004*	< 0.001	0.015*	0.386	0.068	< 0.001
SimpleAdjacency	0.214	0.002	0.009*	0.407	0.02*	< 0.001
SimpleConnected	0.445	< 0.001	0.016*	0.385	0.335	< 0.001
Pushing	0.453	< 0.001	0.020*	0.373	0.324	< 0.001
ModelAdjacency	0.006*	0.069	0.010*	0.305	< 0.001*	0.048
ModelConnected	0.143	0.009	< 0.001*	0.541	< 0.001*	0.011

* Statistically significant.

For the algorithms that showed significant influence of the *configuration* effect, the Holm-Bonferroni tests were performed on this factor alone, indicating that each pair tested was a paired sample. The results showed clear homogeneous groups of configurations in *ModelConnected* (Table 6.4), for which configurations with $MaxTries = 1$ (group A) are significantly worse than the rest (group B). Therefore the configuration

with the least sample mean in group B was selected. In *SimpleAdjacency* and *ModelAdjacency*, even though the ANOVA revealed significant differences, the Holm-Bonferroni tests did not show clear homogeneous groups.

Table 6.4: Results of the multiple comparisons between the configurations tested for the $ModelConnected$ heuristic.

Configuration		DFB (%)		Group	
$SelProb$	$MaxTries$	Mean	Std. Dev.	A	B
0.5	1	4.03	3.09	×	
0.75	1	4.03	3.09	×	
1	1	4.03	3.09	×	
0.5	5	3.35	2.94		×
0.75	5	3.37	2.95		×
1	5	3.35	2.94		×
0.5	10	3.36	2.94		×
0.75	10	3.35	2.94		×
1	10	3.35	2.94		×

Finally, the configuration selected for each algorithm was that with the least sample DFB, but it is worth highlighting the fact that according to the results of the analysis, only for *SimpleAdjacency*, *ModelAdjacency*, and *ModelConnected* were the differences between configurations significant. Table 6.5 shows the configuration selected for each algorithm and its mean and standard deviation of DFB.

Table 6.5: Parameter configuration selected for each Local Search algorithm.

Algorithm	Configuration			DFB (%)	
	$MaxTries$	$SelProb$	$Paths$	Mean	Std. Dev.
$SimpleOverlap$	N	1	60	5.36	3.40
$SimpleAdjacency$	N	0.5	30	4.97	3.32
$SimpleConnected$	N	0.5	30	5.43	3.44
$Pushing$	N	0.75	30	5.21	3.58
$ModelAdjacency$	10	0.5	-	2.88	2.06
$ModelConnected$	10	1	-	3.35	2.94

6.4.3 Selection of a Local Search algorithm

Once a configuration had been selected for each algorithm, another ANOVA was conducted on the results previously obtained to compare the six LS algorithms and the Holm-Bonferroni multiple comparisons over the *algorithm* factor in order to identify groups of algorithms with non-significantly different DFB averages.

The results of the ANOVA for the Local Search algorithms showed that the interaction of *number of vessels* and *algorithm*, *algorithm* alone, and *number of vessels* alone produced statistically significant differences between the means, with p values less than 0.001 in all of them and η^2 equal to 0.111, 0.097, and 0.266, respectively. In contrast to the sample variability observed when comparing different configurations of a single algorithm, now the value of η^2 indicates a greater effect of the *algorithm* and its interaction with the *number of vessels*. This reflects the fact that the differences between the algorithms with respect to DFB were consequently greater. Table 6.6 shows the mean and standard deviation of DFB already reported for each algorithm with the selected configuration, with the homogeneous groups in the remaining columns. One group clearly stands out: group C, with the lowest mean values of DFB, corresponding to the two matheuristics. Therefore, it can be concluded that matheuristics clearly outperformed the other Local Search procedures. Out of all the algorithms, *ModelAdjacency* was thus selected, as it belongs to the best homogeneous group and its sample mean was lower than that obtained by *ModelConnected*.

Table 6.6: Results of the multiple comparisons between the LS algorithms.

Algorithm	DFB (%)		Group		
	Mean	Std. Dev.	A	B	C
SimpleOverlap	5.36	3.40	×		
SimpleAdjacency	4.97	3.32		×	
SimpleConnected	5.43	3.44	×		
Pushing	5.21	3.58	×	×	
ModelAdjacency	2.88	2.06			×
ModelConnected	3.35	2.94			×

The selected Local Search procedure showed a better performance than the others, but it was important to analyse whether the LS algorithm improved significantly on the results obtained by the Genetic Algorithm without any LS algorithm applied at the end. This was analysed through an ANOVA similar to the previous ones. The ANOVA revealed that *number of vessels*, *algorithm*, and their interaction were all significant effects, with p values less than 0.0001. Taking into account that the sample mean DFB obtained by the Genetic Algorithm alone was 5.45% and that obtained by *ModelAdjacency* was 2.88%, it can be concluded that there is a clear performance gain when using this Local Search algorithm.

6.4.4 Evaluation of the Genetic Algorithm

6.4.4.1 Description

The last experiment aimed to evaluate the selected configuration of the Genetic Algorithm with LS over the sets of test instances *GenMB-10m* and *GenPK*. Its results were compared with those achieved by the integer linear model proposed in Chapter 4.4, con-

sidering all the valid inequalities and solved using CPLEX 12.6 with a time limit of 3600 seconds.

Five random repetitions of the Genetic Algorithm were performed on each instance in both sets. As the main objective was to obtain results in short computation times, the overall time limit was set to N seconds (one second per vessel), devoting 90% of the time to the Genetic Algorithm itself and 10% to the Local Search. For each instance and repetition, the cost of the best solution achieved by the Genetic Algorithm alone ($resultGen$) and of that attained after applying the LS procedure ($result$) were recorded. Then the minimum result of the five repetitions ($minResult$) and the deviation percentage of this result from the result of the integer model ($resultModel$) for the same instance were obtained: $minDFM = (minResult - resultModel)/resultModel$. Additionally, the average $result$ over the five repetitions ($avgResult$) and the deviation percentage were calculated: $avgDFM = (avgResult - resultModel)/resultModel$. The average $resultGen$ over the five repetitions ($avgResultGen$) was computed as well to obtain the mean improvement achieved by the LS: $ILS = (avgResultGen - avgResult)/avgResult$. In order to assess the performance of this approach with other reasonable time limits, the experiment was repeated for $3N$, keeping the same time proportions for the *BRKGA* and the *LS*. Thus even for the larger instances of 100 vessels the results were obtained in 5 minutes, which seems acceptable in practical situations.

6.4.4.2 Results

The mixed integer linear model solved all the instances in *GenMB-10m* with up to 40 vessels and seven of those containing 50 vessels to optimality. In two of the instances with 70 vessels, the model could not even obtain a feasible solution, and this also occurred in all the instances containing more than 70 vessels. In one instance of 40 vessels in *GenPK*, optimality was not proven.

The results of the BACAP Genetic Algorithm are summarized in Table 6.7. For each instance size, it shows the mean running time of the MILP; the mean and maximum $minDFM$ and $avgDFM$, considering only the instances for which both the MILP and the algorithm obtained feasible solutions; the percentage of instances solved to optimality by the BRKGA in at least one repetition relative to the number solved optimally by the model, and the percentage of instances solved optimally in all the repetitions; the mean ILS; and the mean cost. The missing values indicate that it was impossible to compare with the model because it could not achieve even feasible solutions. In the case of the percentage of instances solved optimally, it indicates that it is impossible to know whether any optimum was reached, as the model did not achieve any optimal solution.

In column 2 it can be observed that the integer model worked well on small instances, although with sharply increasing computing times in instances with more than 40 vessels. In fact, 50 vessels seems to be the limit for the MILP, as we already saw in previous chapters. For the group of instances with 70 vessels, no solution was found for some instances, and when a feasible solution was found, it was not very good, as compared with the solution obtained by the Genetic Algorithm. The remaining columns in Table 6.7 show the performance of the BRKGA. On very short computing times (1 second per

vessel) it was able to obtain solutions quite close to the optimal solutions, with average deviation no greater than 5% if the best solution obtained in the 5 independent runs on each instance is considered, and no greater than 9% if the average value of the 5 runs is considered, although columns 4 and 6 indicate that there are some instances for which the Genetic Algorithm failed to obtain a really good result. In addition, the contribution of the LS, shown in column 9, was notable, reaching a mean of 2.29% in *GenMB-10m* and 0.59% in *GenPK*, complementing the solution space search done by the Genetic Algorithm in a very efficient way. The last column shows the average cost of the solutions. The sharply increasing values indicate that if a large number of vessels arrives within a time horizon of one week, the problem is highly complex and many conflicts between vessels have to be solved.

Table 6.7: Results of the Genetic Algorithm on *GenMB-10m* and *GenPK* with an overall time limit of one second per vessel (N).

N	Mean time	$minDFM$ (%)		$avgDFM$ (%)		Opt. (%)		Mean	Mean
	model (s)	Avg.	Max.	Avg.	Max.	One	All	ILS (%)	cost
GenMB-10m									
20	0.6	0	0	0.22	2.18	100	90	3.65	13224
30	4.5	0.97	9.36	1.93	9.36	80	30	2.63	41812
40	9.5	1.89	4.43	3.47	10.39	30	20	3.29	55632
50	2084.4	5.00	13.23	8.04	17.20	0	0	1.97	157796
60	3600.0	4.36	13.21	7.32	17.58	-	-	2.48	304000
70	3600.0	-19.05	10.69	-15.78	15.02	-	-	1.90	709388
80	-	-	-	-	-	-	-	2.21	1226960
90	-	-	-	-	-	-	-	1.36	3108228
100	-	-	-	-	-	-	-	1.09	4652508
GenPK									
20	1.2	0.57	5.73	1.21	5.73	90	70	0.58	22300
25	1.7	1.01	10.05	1.21	12.06	90	90	0.05	31936
30	4.0	1.09	5.06	2.04	5.56	60	40	0.55	51224
35	53.8	3.19	10.23	4.92	12.92	40	30	0.40	85676
40	595.2	4.99	10.86	8.62	20.84	22	11	1.38	146680

The results obtained considering a time limit of $3N$ seconds per instance are shown in Table 6.8. The Genetic Algorithm and the associated Local Search made good use of this extended computing time. Minimum, average, and maximum distances to optimal solutions decreased significantly and the number of optimal solutions found by the metaheuristic increased. The average cost of instances in *GenMB-10m* and *GenPK* was reduced by 2.29% and 1.12% respectively.

Table 6.8: Results of the Genetic Algorithm on *GenMB-10m* and *GenPK* with an overall time limit of three seconds per vessel ($3N$).

N	Mean time	$minDFM$ (%)		$avgDFM$ (%)		Opt. (%)		Mean	Mean
	model (s)	Avg.	Max.	Avg.	Max.	One	All	ILS (%)	cost
GenMB-10m									
20	0.6	0	0	0	0	100	100	3.89	13200
30	4.5	0.94	9.36	1.61	9.36	90	40	3.03	41716
40	9.5	1.78	4.35	2.39	6.43	30	30	3.70	55104
50	2084.4	3.45	10.49	6.20	15.21	14	0	2.53	154760
60	3600.0	1.59	6.38	5.85	11.66	-	-	2.39	299488
70	3600.0	-21.23	5.42	-17.65	12.11	-	-	2.20	703168
80	-	-	-	-	-	-	-	2.39	1196556
90	-	-	-	-	-	-	-	1.44	3049068
100	-	-	-	-	-	-	-	1.22	4526332
GenPK									
20	1.2	0.57	5.73	0.94	5.73	90	70	0.83	22252
25	1.7	1.01	10.05	1.14	11.43	90	90	0.04	31912
30	4.0	0.96	5.06	1.92	5.75	70	40	0.53	51176
35	53.8	2.69	9.65	3.64	9.65	50	30	0.48	84584
40	595.2	3.43	8.14	7.14	14.42	22	0	1.47	144140

In summary, these experiments show that the Genetic Algorithm accomplishes the main objective of obtaining good solutions for both small and large instances in short times. The percentage of optimal solutions attained indicates that it performs a good exploration of the solution space. Indeed, the mean deviations show that it is the best option for instances with more than 60 vessels and a fast solving method in all cases. According to the results, the improvement rate is expected to decrease notably as the time is increased above $3N$.

6.5 Concluding remarks

In this chapter, new heuristic methods have been proposed for the Berth Allocation and Quay Crane Assignment Problem (BACAP) in its continuous, dynamic and time-invariant version. The objective was to develop a metaheuristic approach capable of obtaining good solutions in short times for both small and large instances. In particular, a new Biased Random-Key Genetic Algorithm with Memetic characteristics, a constructive algorithm, and several Local Search procedures have been proposed. The parameters of these algorithms were adjusted through extensive experiments and statistical analysis.

It has been observed that the solutions based on the ordering of the vessels provided by the Genetic Algorithm may be not good enough in cases in which clusters of vessels

arriving at similar times and preferring similar positions on the quay have to be assigned in a globally optimal way. To address this issue, a matheuristic procedure, which can dramatically improve the solutions, was proposed.

The results obtained in instances generated according to well-known criteria show that the proposed method is able to achieve good and even optimal solutions in less than 5 minutes on instances with less than 60 vessels within a planning horizon of one week, while it clearly outperforms the exact methods on instances with up to 100 vessels within the same time horizon. Indeed, those methods cannot even produce feasible solutions for such large instances, so this approach broadens the computational capabilities in the field for tackling this variant of the BACAP.

Chapter 7

Extending the Biased Random-Key Genetic Algorithm to solve the Berth Allocation and Specific Quay Crane Assignment Problem

7.1 Introduction

In the previous chapter we have seen that the increasing vessel traffic and congestion experienced by many terminals require new fast methods able to deal with large instances of the continuous time-invariant BACAP. Given that the exact methods proposed for this problem showed important limitations when confronting such instances, a new Biased Random-key Genetic Algorithm was developed to obtain good solutions on instances of a wide range of sizes under severe time constraints.

In Chapter 5 we saw that the BACASP is even more complex than the BACAP and hence the difficulties faced by the exact methods when solving large instances are also exceptionally challenging. In most of the instances with 60 vessels, these methods could not even obtain feasible solutions in one hour. Therefore, a heuristic strategy is a fortiori required to obtain good solutions to BACASP instances of a wide range of sizes in short times.

In this chapter, the Biased Random-key Genetic Algorithm developed for the BACAP in the previous chapter is extended to deal with the BACASP (Section 7.2) and analogous experiments are conducted to adjust its parameters and assess its performance (Section 7.3).

7.2 Extending the algorithms to address the BACASP

In Chapter 5.4 we saw that given an optimal solution of the BACAP, an optimal solution of the BACASP can be easily obtained from it in polynomial time if and only if every complete sequence of vessels in the solution of the BACAP is proper. Taking this into account, it is easy to adapt the Biased Random-key Genetic Algorithm developed for the BACAP to solve the BACASP.

Given that the BRKGA and all the Local Search procedures, except the matheuristic, rely on the constructive algorithm to obtain feasible solutions, if this algorithm is adapted so that it generates only BACAP solutions not containing improper complete sequences of vessels, valid solutions for the BACASP can be obtained from all of them. By contrast, the matheuristic now applies the cutting plane algorithm based on the BACAP model.

7.2.1 The constructive algorithm for the BACASP

The constructive algorithm is the same as that proposed for the BACAP, the only difference being that it now has to prevent vessels being assigned to locations in which they would generate improper complete sequences. Thus when trying to assign berthing position and time to a vessel i, after checking that the candidate assignment satisfies the number of cranes required and does not overlap with any other vessel, a check is also made to see whether it creates one or more improper complete sequences. If so, new minimum-cost alternative assignments that break all the improper complete sequences found are added to the set of candidate assignments. To prevent falling into cycles of repeated candidates, a new set of already visited candidates is included. Figure 7.1 illustrates this process for the case in which vessel i would create an improper complete sequence when assigned to its ideal location (a_i, d_i). The constructive algorithm continues until each vessel is assigned to a position and a time in which all constraints are satisfied.

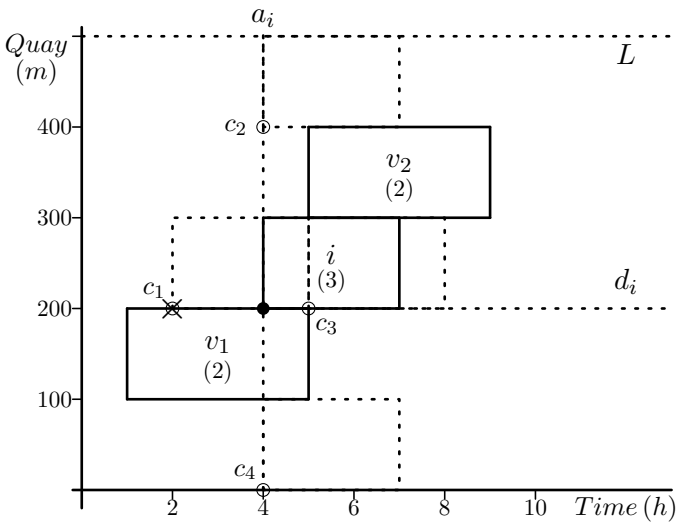


Figure 7.1: Example of applying the BACASP constructive algorithm to a quay with 5 cranes. Vessel i generates an improper complete sequence (v_1, i, v_2) in its ideal assignment. Alternative assignments $c_1, \ldots, c_4$ are considered in order to break the sequence. Note that candidate c_1 is immediately discarded because the time is lower than a_i.

7.2.2 The matheuristic Local Search for the BACASP

The matheuristic is adapted to address the BACASP by using the Algorithm 5 based on cutting planes proposed in Chapter 5.4.

First, a cluster of vessels is selected according to a neighbourhood function and a score based on the costs of the vessels in the cluster. Then, the solver is run for the BACAP model with additional constraints that fix the values of the variables related to the vessels not in the cluster. If a feasible solution is found, it is examined to check whether it contains improper complete sequences of vessels. For each sequence found, a constraint is included in the BACAP model to prevent those vessels from forming that improper complete sequence in the future. If the sequence contains up to $MaxSeqSizeForPerm$ vessels, all the constraints resulting from all the permutations of those vessels are also introduced. Then, the new model is run and the process is repeated until a BACAP solution not containing improper complete sequences is attained. In doing so, the solver is provided with the value of the current solution as an upper bound.

As in the other LS procedures, if the global solution thus obtained is better than the best solution achieved in the search, then the process is applied again to that solution. If not, a new vessel with its corresponding cluster is selected according to the score. In any case, since the stopping criterion is a time limit, it may happen that neither optimal nor feasible solutions are found.

7.3 Computational experiments

The same experiments conducted on the Genetic Algorithm and the Local Search procedures for the BACAP were repeated to adjust and evaluate the versions of these algorithms developed for the BACASP. The conditions of the experiments, including the sets of instances (*GenPK*, *GenMB-10m*, and *GenAdjust*), the computational resources, and the statistical analysis were exactly the same as those described in the previous chapter.

Therefore, three different experiments were conducted. The first was done to obtain a good configuration of the parameters of the Genetic Algorithm without performing Local Search (Section 7.3.1). The second experiment was aimed at selecting a good parameter configuration for each Local Search algorithm (Section 7.3.2). In Section 7.3.3, the best Local Search is selected according to the results obtained in the previous experiment. Finally, the third experiment (Section 7.3.4) evaluated the performance of the entire algorithm and compared its results with those achieved by the cutting plane algorithm proposed in Chapter 5, which was the best exact approach for the BACASP according to the results of the experiments presented therein.

To avoid unnecessary repetitions, these sections focus on the results obtained by the BACASP algorithms, while the detailed description of the experiments can be consulted in the previous chapter.

7.3.1 Parameter adjustment of the Genetic Algorithm

In this experiment, the parameters $E = 15\%$, $I = 15\%$, and $CrossBias = 0.7$ were fixed and 15 different combinations of the parameters $MultPop \in \{1, 10, 20\}$, $N_{pop} \in \{1, 10, 20\}$, and $GensMig \in \{10, 30\}$ were tested and analysed. The algorithm was run considering $TimeLimitGen = N$ seconds.

The tests applied to the results showed statistically significant differences in the average Deviation From the Best result (DFB) due to the interaction between *number of vessels* and *configuration*, *configuration* alone, and *number of vessels* alone, with $p < 0.0001$ in all of them and η^2 equal to 0.062, 0.114, and 0.379, respectively. As in the case of the BACAP algorithm, these results show that the configurations tested differ in their DFB both on its own and jointly with the size of the problem. *Number of vessels* explained most of the sample variability, reaching 37.9%, according to its η^2, followed by *configuration* and *interaction*.

In order to assess the differences between the configurations, the Holm-Bonferroni tests were applied, specifying that the samples were paired. Table 7.1 shows the results. The same effect of the total number of individuals on the number of generations completed and the consequent differences in performance can be seen here as in the case of the BACAP Genetic Algorithm. Compared with the latter, this algorithm completes approximately half the number of generations, which clearly reflects the complexity of the BACASP and the additional time spent on generating solutions without improper complete sequences. Again, the values tested for the parameter $GensMig$ did not seem to make a significant and consistent difference in performance. The homogeneous group B consists of the best configurations, from which the one with the lowest sample average was selected for the rest of the study: $MultiPop = 10$, $N_{pop} = 1$, and $GensMig = 10$.

Table 7.1: Results of the Genetic Algorithm on the set *GenAdjust*.

Configuration			Generations	DFB (%)		Group			
$MultPop$	N_{pop}	$GensMig$	Mean	Mean	Std. Dev.	A	B	C	D
1	1	10	13768.1	9.56	8.21	×			
1	10	10	1375.6	6.28	4.99		×		
1	10	30	1377.5	6.71	5.51		×		
1	20	10	684.8	5.91	5.02		×		
1	20	30	685.8	6.94	5.79		×		
10	1	10	1511.0	5.16	5.04		×		
10	10	10	141.4	13.38	13.77	×			
10	10	30	141.3	13.71	14.08	×			
10	20	10	68.8	21.59	20.36			×	
10	20	30	68.8	21.70	20.36			×	
20	1	10	694.5	5.42	4.98		×		
20	10	10	63.5	22.40	20.81			×	
20	10	30	63.6	22.70	20.58			×	
20	20	10	30.7	33.72	29.82				×
20	20	30	30.8	33.58	29.46				×

7.3.2 Parameter adjustment of the Local Search procedures

Analogous experiments were conducted to adjust the parameters of the LS procedures: *SimpleOverlap*, *SimpleAdjacency*, *SimpleConnected*, *Pushing*, *ModelAdjacency*, and *ModelConnected*. In all the algorithms, except in the matheuristic, the parameter $MaxTries$ was set to N and all the combinations of $SelProb \in \{0.5, 0.75, 1\}$ and $Paths \in \{30, 45, 60\}$ were tested. For the matheuristic, $MaxClusterSize = 15$ and $MaxSeqSizeForPerm = 4$ were kept fixed and all the combinations of $SelProb \in \{0.5, 0.75, 1\}$ and $MaxTries \in \{1, 5, 10\}$ were tested. This resulted in a total of 9 different configurations for each LS procedure. The stopping criterion was $TimeLimitGen = 0.9N$ and $TimeLimitLS = 0.1N$ seconds.

Table 7.2 shows the results of the ANOVA conducted for each algorithm, recording the p value and the η^2 for each effect. The interaction of both *number of vessels* and *configuration* was statistically significant only in *ModelAdjacency*, while *configuration* alone was significant in *SimpleOverlap*, *SimpleAdjacency*, *ModelAdjacency*, and *ModelConnected*. The influence of these effects on the sample variability was low compared to the influence of *number of vessels*. Indeed, this effect was significant in all the algorithms.

Table 7.2: Results of the ANOVA and η^2 calculations for each LS algorithm in the parameter adjustment experiment.

Algorithm	Interaction		No. of vessels		Configuration	
	p value	η^2	p value	η^2	p value	η^2
SimpleOverlap	0.128	< 0.001	0.005*	0.432	0.037*	< 0.001
SimpleAdjacency	0.631	0.002	0.005*	0.515	< 0.001*	< 0.001
SimpleConnected	0.556	< 0.001	0.002*	0.467	0.155	< 0.001
Pushing	$-^a$	< 0.001	0.02*	0.473	$-^a$	< 0.001
ModelAdjacency	< 0.001*	0.018	< 0.001*	0.59	< 0.001*	0.024
ModelConnected	0.246	0.006	< 0.001*	0.63	0.02*	0.003

* Statistically significant.

[a] No value reported by the analysis, as the results were the same in all the configurations tested.

Given that the objective of the experiment was to select a *configuration* for each algorithm, the Holm-Bonferroni multiple comparison tests were conducted over this factor on the results of *SimpleOverlap*, *SimpleAdjacency*, *ModelAdjacency* and *ModelClustering*, indicating that each pair tested was a paired sample. Only in the case of *ModelAdjacency* did the multiple comparisons reveal clear distinct groups, which are summarized in Table 7.3. In particular, the mean DFB in group A was clearly higher than in group E, while groups B and C showed intermediate results. The main cause of these differences lies in the $MaxTries$ parameter, which leads to better results as it is increased.

Table 7.3: Results of the multiple comparisons between the configurations tested for the *ModelAdjacency* heuristic.

Configuration		DFB (%)		Group				
SelProb	*MaxTries*	Mean	Std. Dev.	A	B	C	D	E
0.5	1	4.89	4.05	×				
0.75	1	4.85	4.00	×				
1	1	4.85	4.00	×				
0.5	5	4.15	3.49		×			
0.75	5	3.89	3.30			×		
1	5	3.89	3.30			×		
0.5	10	3.63	3.12				×	
0.75	10	3.53	3.12				×	×
1	10	3.51	3.11					×

For the rest of the study, the configuration that gave rise to the least sample mean DFB was selected for each LS algorithm, but it is worth noting that only in the cases of *SimpleOverlap*, *SimpleAdjacency*, *ModelAdjacency*, and *ModelConnected* are those configurations at least as good as the best, in the population and not only in the sample, according to the significant differences reported by the ANOVA. In Table 7.4 the selected configurations are summarized together with their mean DFB and standard deviation.

Table 7.4: Parameter configuration selected for each LS algorithm.

Algorithm	Configuration			DFB (%)	
	MaxTries	*SelProb*	*Paths*	Mean	Std. Dev.
SimpleOverlap	N	1	60	6.18	4.81
SimpleAdjacency	N	1	60	5.71	4.66
SimpleConnected	N	0.75	30	6.36	4.81
Pushing	N	0.5	30	6.28	4.85
ModelAdjacency	10	1	-	3.51	3.11
ModelConnected	10	1	-	4.95	4.31

7.3.3 Selection of a Local Search algorithm

The results obtained by the algorithms with their selected configurations in the previous experiment allowed a Local Search procedure to be selected for the final version of the BRKGA. The ANOVA revealed that the *number of vessels* alone, the *algorithm* alone, and their interaction were all significant effects for the variations observed in the data. Regarding η^2, the values were 0.42, 0.05, and 0.046, respectively.

Again, the Holm-Bonferroni multiple comparisons were conducted, and their results are shown in Table 7.5 together with the mean and standard deviation of DFB for each

algorithm. Group A comprises the algorithms with the highest mean DFB, group B reached intermediate results, while the algorithm *ModelAdjacency*, the only algorithm in group C, shows the lowest mean. Therefore, this algorithm was selected for the final version of the Genetic Algorithm.

Table 7.5: Results of the multiple comparisons between the LS algorithms.

Algorithm	DFB (%)		Group		
	Mean	Std. Dev.	A	B	C
SimpleOverlap	6.18	4.81	×		
SimpleAdjacency	5.71	4.66		×	
SimpleConnected	6.36	4.81	×		
Pushing	6.28	4.85	×		
ModelAdjacency	3.51	3.11			×
ModelConnected	4.95	4.31		×	

In order to assess whether the selected Local Search improved significantly on the results obtained by the Genetic Algorithm alone, an analogous ANOVA was conducted. The analysis showed that *number of vessels*, *algorithm*, and their interaction were all statistically significant, with p values of less than 0.0001. The sample mean DFB of the Genetic Algorithm was 6.39%, and that of *ModelAdjacency* was 3.51%, which is significantly lower. Therefore, the selected LS procedure is worth applying after the BRKGA, as it significantly improves on its results.

7.3.4 Evaluation of the Genetic Algorithm

The selected Genetic Algorithm was compared with the cutting plane algorithm for the model (CPA) on the sets of instances *GenMB-10m* and *GenPK*. As was reported in Chapter 5.5, this algorithm was able to achieve optimal solutions in all the instances with up to 40 vessels and in two instances with 50 vessels in the set *GenMB-10m*. In *GenPK*, only one instance with 35 vessels and three with 40 vessels could not be solved to optimality. In the instances in which it could not achieve an optimal solution, no feasible solutions were obtained.

Table 7.6 shows the results obtained by the BRKGA in both sets of instances considering a time limit of N seconds. DFM refers to the deviation of the results obtained by the BRKGA from the results obtained by the CPA. In general, they are similar to those obtained in the case of the BACAP, except for a slight decrease in the size of the instances that the CPA was able to solve. Likewise, the Genetic Algorithm achieved an optimal solution in many of the instances with 20 and 30 vessels. The results also show that it did not perform well on at least two instances, one with 20 vessels in *GenMB-10m* and the other with 40 vessels in *GenPK*, whose DFM exceeded 40%. These values affected the means of their corresponding groups, hence the high $avgDFM$ observed in the instances with 40 vessels in *GenPK*. Nevertheless, it is worth noting that the

BACASP is more difficult to solve than the BACAP and that the Genetic Algorithm spends more time constructing solutions without improper complete sequences. As for the Local Search, it attained a mean improvement of 1.58%, proving again to be a good complement to the Genetic Algorithm.

Table 7.6: Results of the BRKGA on *GenMB-10m* and *GenPK* with an overall time limit of one second per vessel (N).

N	Mean time	$minDFM$ (%)		$avgDFM$ (%)		Opt. (%)		Mean	Mean
	model (s)	Avg.	Max.	Avg.	Max.	One	All	ILS (%)	cost
GenMB-10m									
20	0.7	5.14	40.48	5.14	40.48	80	80	0.18	14100
30	13.5	2.53	9.76	2.70	9.76	40	30	3.13	43316
40	85.8	2.24	4.80	4.44	12.44	20	10	3.15	59276
50	3292.1	4.06	6.43	6.51	8.80	0	0	1.29	177960
60	3600.0	-	-	-	-	-	-	1.62	363836
70	3600.0	-	-	-	-	-	-	1.92	838164
80	-	-	-	-	-	-	-	1.49	1438184
90	-	-	-	-	-	-	-	0.82	3311732
100	-	-	-	-	-	-	-	0.62	4778248
GenPK									
20	1.8	0.97	8.33	1.14	8.33	80	70	0.00	22996
25	3.1	0.46	2.30	0.76	2.58	80	60	0.01	32240
30	15.4	0.17	1.11	2.14	6.33	80	50	0.44	53808
35	640.6	1.75	5.34	5.23	12.07	44	22	0.39	90884
40	1516.2	12.3	36.67	20.10	53.86	0	0	3.35	170744

The experiment conducted considering a time limit of $3N$ yielded some improvements relative to the results obtained in N seconds, especially by reducing the maximum $avgDFM$ (Table 7.7). It also reached new optimal solutions in five instances in set *GenPK* and reduced the mean cost by 1.85% in *GenMB-10m* and 2.68% in *GenPK*.

In summary, both experiments show that the proposed Genetic Algorithm is also a good approach for solving the BACASP, particularly when more than 40 vessels are expected to arrive within a time horizon of one week, since the exact method cannot even find feasible solutions. As in the case of the BACAP version of the algorithm, the percentage of optimal solutions attained indicates that it performs a good exploration of the solution space. The improvement rate is expected to decrease notably as the time is increased above $3N$.

Table 7.7: Results of the BRKGA on *GenMB-10m* and *GenPK* with an overall time limit of three seconds per vessel ($3N$).

N	Mean time	$minDFM$ (%)		$avgDFM$ (%)		Opt. (%)		Mean	Mean
	model (s)	Avg.	Max.	Avg.	Max.	One	All	ILS (%)	cost
GenMB-10m									
20	0.7	1.33	10.91	4.38	32.86	80	80	0.64	13972
30	13.5	2.45	9.76	2.68	9.76	30	30	3.01	43300
40	85.8	2.03	4.80	2.97	5.84	20	10	2.31	58272
50	3292.1	4.06	6.43	4.61	6.64	0	0	1.79	174568
60	3600.0	-	-	-	-	-	-	3.03	355704
70	3600.0	-	-	-	-	-	-	1.90	818300
80	-	-	-	-	-	-	-	1.74	1390468
90	-	-	-	-	-	-	-	1.02	3231884
100	-	-	-	-	-	-	-	0.90	4738120
GenPK									
20	1.8	0.97	8.33	0.97	8.33	80	80	0	22940
25	3.1	0	0	0.35	1.57	100	60	0.01	32068
30	15.4	0.53	1.94	1.65	6.96	70	60	0.32	53428
35	640.6	1.54	5.89	3.53	12.15	55	44	0.72	89516
40	1516.2	7.09	29.52	13.62	37.43	14	0	2.71	163052

7.4 Concluding remarks

In this chapter, the Biased Random-Key Genetic Algorithm and the Local Search algorithms proposed for the BACAP in the previous chapter have been extended to address the BACASP.

These algorithms has been adjusted and evaluated through analogous experiments, in which the higher complexity of the BACASP was observed. Indeed, according to the results, the adapted Genetic Algorithm completes half of the number of generations completed by the BACAP version considering the same time limit. Nevertheless, it obtains optimal or quasi-optimal solutions on small instances and is able to solve instances of up to 100 vessels, for which the exact methods cannot even obtain feasible solutions. The matheuristic Local Search proves to be the best option for improving the solutions obtained by the Genetic Algorithm.

Given that the continuous time-invariant BACASP assumes more real-world characteristics and constraints than other Berth Allocation Problems, the proposed Genetic Algorithm for this problem proves to be a more practical approach, closer to the demands posed by container terminals.

Chapter 8

The Berth Allocation Problem in terminals with irregular layouts

8.1 Introduction

Most ports around the world have heterogeneous and irregular layouts, resulting from the use of the natural coastline or the construction of artificial structures that project from the land out into the water. Irregular layouts may impose physical limitations on how vessels sail and berth as additional safety constraints. In terminals such as the one depicted in Figure 8.1, the distances between berths may prevent vessels from approaching or being moored at some berths when they are occupied by other vessels. Given that such restrictions may affect the service of vessels, they are taken into consideration by the terminal operator in berth planning. However, constraints of this kind have received little attention in the scientific literature on the Berth Allocation Problem.

This chapter addresses a hybrid dynamic BAP arising in terminals whose irregular layout gives rise to adjacency, oppositional, and blocking relations between berths. Additionally, vessels may be prevented from being moored in berths depending upon their characteristics. In order to formulate and solve this problem, a mixed integer linear model is proposed and several computational experiments are conducted over different sets of instances.

Although a general version of this problem is tackled here, it originally arose from the particular case of a tank terminal located at the port of Antwerp (Belgium), which I analysed during a research visit to the Combinatorial Optimisation and Decision Support Group (CODeS) of the KU Leuven in Ghent. Tank terminals are especially affected by such constraints because they are often composed of many close narrow piers aimed to make the most of their harbour basin. The space between piers is thus reduced and the distances between moored vessels, even in opposite berths, may become a limitation for accessing other berths. Similarly, such constraints also affect terminals of any kind located in fluvial waterways, as they are usually required to allow navigation through the channels. Another characteristic of tank terminals, as well as many dry bulk terminals, is that berthing points are specific locations on the quays equipped with special

devices; therefore, vessels can only be served there. For example, berths have connections to pipelines that serve different products or, in the case of dry bulk terminals, conveyors, cranes, and machinery that enable goods to be transported from the storage area. The approach presented here also takes this factor into consideration without loss of generality.

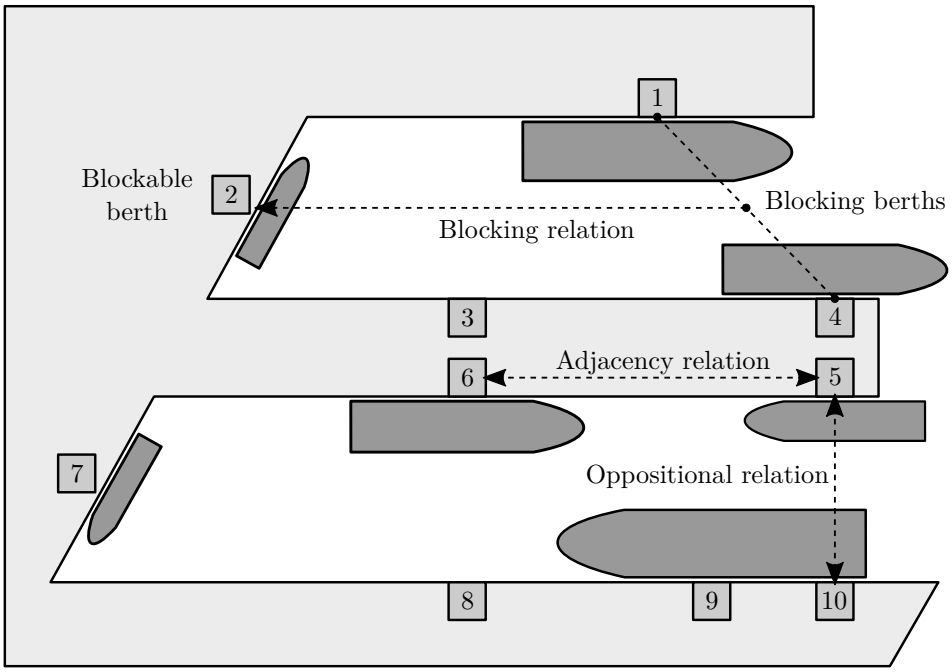


Figure 8.1: Terminal with irregular layout. Only some of the relations are shown.

The structure of this chapter is as follows: Section 8.2 reviews related previous studies. Next, Section 8.3 describes the problem in detail. In Section 8.4, a mixed integer linear programming model tailored to the problem is proposed. The computational experiments and the discussion of the results are addressed in Section 8.5. Finally, Section 8.6 presents some concluding remarks.

8.2 Literature review

The BAP with adjacency, oppositional, and blocking relations between berths has not been tackled before in a general fashion. The closest approaches in the literature seem to be those that study specific indented berths located in particular ports.

Imai et al. (2007) addressed a discrete BAP in which a dedicated indented berth was able to serve either a mega-containership or up to four small vessels. The indented berth studied, located in the port of Amsterdam, had cranes on both sides so that they could serve a single large vessel faster than in an ordinary berth. When there were no such large vessels in the port, terminal operators made use of this berth to serve smaller vessels. In the specific case confronted by the authors, each side of the berth admitted

up to two small vessels at the same time if their combined lengths did not exceed the total length of the indented berth. This posed the problem of accessing or leaving the inner sections of the berth when the outer sections were in use. Consequently, the berth planner had to take into account the constraints resulting from this situation, such as the need to postpone vessel departures from the inner sections until the outer ones had been released. The authors first proposed an MILP for the ordinary case of a hybrid dynamic BAP with given berth-dependent handling times. Next, they extended this model to address the case with several indented berths like the one previously described. They proposed a genetic algorithm to solve this problem and conducted several computational experiments to assess the influence of the indented berth on the service time of vessels.

Subsequently, the same authors (Imai et al., 2013) also proposed a genetic algorithm to solve similar Berth Allocation Problems. In this case, besides the indented berth already mentioned, they also studied what they called *channel berths*, which consist of two parallel quays forming a channel. They modelled the channel berth as an indented berth with both ends open and here again they only allowed to moor in it either one mega-containership or up to four small vessels. They also conducted several computational experiments to compare the service times of vessels in different scenarios representing ordinary terminals, terminals with indented berths, and terminals with channel berths.

The studies that specifically addressed the BAP in bulk terminals did not take adjacency, oppositional, or blocking relations into consideration either. Various authors proposed models taking account of tidal constraints (Barros et al., 2011, Xu et al., 2012, Qin et al., 2016, Ernst et al., 2017), which were also considered in investigations concerning container terminals (see, for instance, Lalla-Ruiz et al., 2016). Umang et al. (2013) and Robenek et al. (2014) proposed various models and heuristics for a hybrid BAP with berths specializing in a range of cargo handling. Bridi et al. (2016) addressed a similar situation arising in a tank terminal by proposing a new model for a continuous BAP. Zhicheng et al. (2013) developed a heuristic procedure for a hybrid BAP in a coal terminal, taking into account spaces outside the quay boundary. Another study related to a coal terminal was presented by Kordić et al. (2015), who proposed a model and various heuristics implementing rules provided by the terminal operator. Ribeiro et al. (2016) addressed a discrete BAP in an ore terminal in which maintenance activities on the berths could be scheduled and so limit the mooring of vessels. A BAP in inland waterway ports was studied by Grubišić et al. (2014), for which they proposed a model taking into account the possibility of berths serving different cargo at different rates.

Based on the literature review, it can be concluded that the version of the BAP addressed in this investigation has not been previously addressed in the field. The next section describes this variant and its assumptions in detail.

8.3 Problem description

8.3.1 Overview

The problem tackled in this study is a hybrid dynamic BAP in which mooring in one or more sets of berths can be limited according to the relations between the berths

or their particular characteristics (Figure 8.1). The objective is to minimize the total assignment cost, which is the sum of the cost of waiting before berthing and the delay cost of each vessel. In terms of the scheme proposed by Bierwirth and Meisel (2010), this problem can be classified as $hybr, draft \,|\, dyn \,|\, pos \,|\, \sum(w_1\, wait + w_2\, tard)$, although new categories will be necessary to properly describe the special relations considered.

A solution to the problem can be rendered as a space-time diagram in which the berths are represented on the vertical axis and the time is represented on the horizontal axis (Figure 8.2). Thus the schedule of a berth is rendered as a horizontal line and the schedule of a vessel is a horizontal segment on one of the berth lines whose left-most point represents its *berthing time* and whose right-most point represents its *departure time*. A vessel may be forced to wait at the berth after the end of its handling as a consequence of blocking restrictions, as we will see in the next subsection.

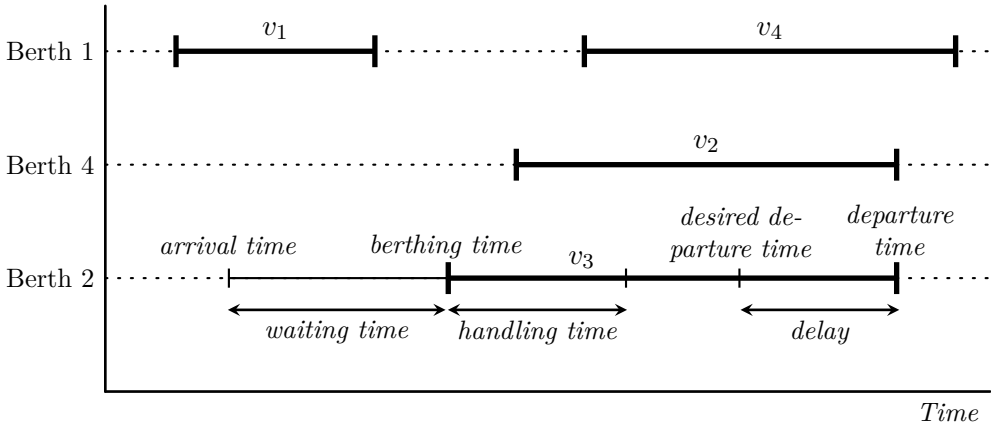


Figure 8.2: A berth plan over three berths and four vessels.

8.3.2 Assumptions

The particular assumptions of this problem are as follows:

- A *berth* is considered as a specific point on the quay (Figure 8.1).
- A berth can accommodate one vessel at a time.
- A vessel is moored lengthwise parallel to the quay with its middle coinciding with its assigned berth.
- Once a vessel is moored, its position cannot be changed, nor can its handling be interrupted.
- A vessel moored at a berth may extend into an adjacent berth if it is long enough. Consequently, that berth will not be available during its stay. For example, in Figure 8.1 this applies to berths 9 and 10. The same may occur between opposite berths, depending on the beam of the vessels. This contrasts with most discrete

BAPs in the literature, in which a berth is a specific section of the quay and no interferences between them are considered. For this reason, this problem can be classified as a hybrid BAP.

- Vessel priorities may be reflected by means of cost coefficients.
- The time for docking and undocking maneuvers is considered to be included in the vessel handling time.

The characteristics of the berths and the spatial relations between berths may give rise to various restrictions:

- Restrictions on mooring
  - *Availability restriction.* This forbids vessels being moored in a given berth from the beginning of the planing interval until a given *release time*.
  - *Structural restriction.* This forbids a given vessel being moored at a given berth at any time. This kind of restriction often arises from physical berth limitations, such as the draft or the type of cargo handled. When a vessel satisfies all the structural restrictions related to a berth, we say that this berth is *compatible* with that vessel.
  - *Adjacency restriction.* This prevents vessels from occupying a pair of adjacent berths concurrently if a given inter-ship clearance in length is not satisfied. For example, in Figure 8.1 the pairs of berths $\{3, 4\}$, $\{5, 6\}$, $\{8, 9\}$, and $\{9, 10\}$ are adjacent and are thus affected by this kind of restriction.
  - *Oppositional restriction.* This prevents vessels from occupying a pair of opposite berths concurrently if a given inter-ship clearance in width is not satisfied. For example, in Figure 8.1 the pairs of berths $\{1, 3\}$, $\{1, 4\}$, $\{5, 9\}$, $\{5, 10\}$, $\{6, 8\}$, and $\{6, 9\}$ are opposite each other and are thus affected by this kind of restriction.
  - *General mooring restriction.* This prevents a given set of vessels from occupying a given set of berths concurrently. This kind of restriction represents decisions of the terminal operator aimed at addressing special situations, such as those involving berths that process hazardous cargo.
- Restrictions on both berthing and departure
  - *Blocking restriction.* A restriction of this kind prevents a given vessel from berthing at or departing from a berth, called a *blockable berth*, when a set of other berths, called *blocking berths*, are concurrently occupied by a given combination of vessels. The blockable berth will be blocked to that vessel during the period in which those vessels coincide in time. For example, in Figure 8.1 berths 1, 2, and 4 and the vessels moored in them are involved in this kind of relationship. Figure 8.3 shows this blocking situation in time. In Figure 8.2, vessel v_3 cannot depart because is blocked by vessels v_2 and v_4 and thus is forced to wait until one of them departs.

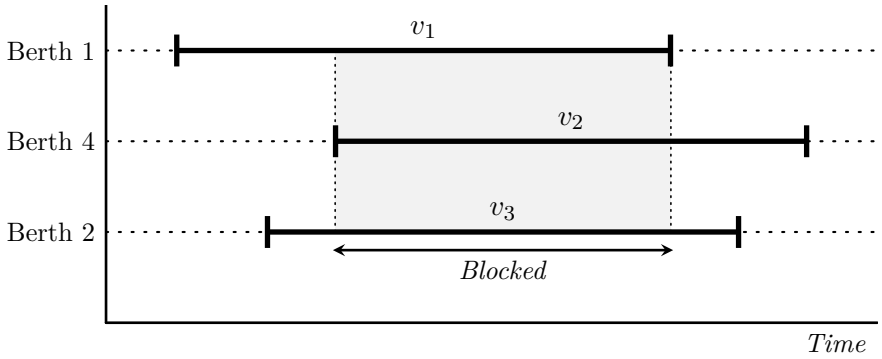


Figure 8.3: Example of a blocking situation. Berth 2 is blockable by berths 1 and 4 for this combination of vessels. Vessel v_3 cannot berth in or depart from berth 2 within the shaded period.

8.3.3 Parameters

The following data on the terminal, the vessels, and the restrictions define an instance of the problem:

- Berths:
  - Set of berths: B. $N_B = |B|$
  - Set of berths that are opposite each other:
    $B^o = \{(f,k) \in B \times B \mid \text{berths } f \text{ and } k \text{ are opposite each other}\}$
  - Set of adjacent berths: $B^a = \{(f,k) \in B \times B \mid \text{berths } f \text{ and } k \text{ are adjacent}\}$
  - For each berth $k \in B$, we know its release time: rel_k
  - For each $(f,k) \in B^o$, we know:
    - Berth separation in width: d^o_{fk}
    - Required inter-ship clearance: c^o_{fk}
  - For each $(f,k) \in B^a$, we know:
    - Berth separation in length: d^a_{fk}
    - Required inter-ship clearance: c^a_{fk}
- Vessels:
  - Set of vessels: V. $N_V = |V|$
  - For each ship $i \in S$, the following data are known:
    - Length: l_i
    - Beam: w_i
    - Expected arrival time: a_i
    - Desired departure time: s_i
    - Set of compatible berths: $B_i \subseteq B$

  - Estimated handling time in berth $k \in B_i$: h_i^k
  - Cost per waiting unit time for berthing after the expected arrival time: C_i^w
  - Cost per delay unit time after the desired departure time: C_i^d

- Set of tuples of vessels and berths representing mooring restrictions by decision of the terminal operator:
  $D = \{(b_1, v_1, \ldots, b_n, v_n) \mid v_1, \ldots, v_n \in V; (b_1, \ldots, b_n) \in B_{v_1} \times \cdots \times B_{v_n} :$
  the operator forbids the concurrent occupation of b_1 by v_1, b_2 by $v_2, \ldots, b_n$ by $v_n\}$
  These relations are not necessarily symmetrical.

- Set of tuples of vessels and berths representing blocking restrictions:
  $F = \{(b_1, v_1, \ldots, b_n, v_n) \mid v_1 \ldots v_n \in V; (b_1, \ldots, b_n) \in B_{v_1} \times \cdots \times B_{v_n} :$
  b_1 can be blocked for v_1 by the concurrent occupation of b_2 by $v_2, \ldots, b_n$ by $v_n\}$

8.4 A mixed integer linear model

The following mixed integer linear model is proposed to formulate the BAP in terminals with irregular layouts previously described.

8.4.1 Precalculated sets

In order to avoid generating unnecessary variables and constraints, the following sets are defined, based on the input data:

- Set of tuples of vessels and adjacent berths incompatible with respect to their length:
  $A = \{(f, i, k, j) \mid i, j \in V; f \in B_i; k \in B_j; i < j; f \neq k; \frac{l_i}{2} + \frac{l_j}{2} + c_{fk}^a > d_{fk}^a\}$

- Set of tuples of vessels and opposite berths incompatible with respect to their width:
  $O = \{(f, i, k, j) \mid i, j \in V; f \in B_i; k \in B_j; i < j; f \neq k; w_i + w_j + c_{fk}^o > d_{fk}^o\}$

- Set of tuples used to avoid pairs of vessels being served concurrently at the same berth: $C = \{(k, i, k, j) \mid i, j \in V; k \in B_i; k \in B_j; i \neq j\}$

- Set of all the tuples that represent incompatible mooring of specific ships on specific berths: $I = C \cup A \cup O \cup D$

- Set of pairs of vessels in which vessel i can be moored on a berth blockable by another berth which in turn can admit ship j:
  $P = \{\, (i, j) \in V \times V \mid \exists (b_1, v_1, \ldots, b_n, v_n) \in F, \exists k \in \{2, \ldots, n\} : i = v_1, j = v_k \,\}$

8.4.2 Variables

In this problem a berth, a berthing time, and a departure time are to be assigned to each calling vessel. Thus the decision variables are:

t_i = berthing time of vessel i

r_i = departure time of vessel i

u_i = delay incurred by vessel i relative to its desired departure time

$$m_i^k = \begin{cases} 1, & \text{if vessel } i \text{ is moored on berth } k \\ 0, & \text{otherwise} \end{cases}$$

$$\sigma_{ij} = \begin{cases} 1, & \text{if vessel } i \text{ is completely to the left of vessel } j \text{ in the time} \\ & \text{diagram, that is, vessel } i \text{ is handled completely before vessel } j \\ 0, & \text{otherwise} \end{cases}$$

$$\gamma_{ij} = \begin{cases} 1, & \text{if } t_i \leq t_j, \text{ where } i, j \text{ are vessels} \\ 0, & \text{otherwise} \end{cases}$$

$$\phi_{ij} = \begin{cases} 1, & \text{if } r_i \geq r_j, \text{ where } i, j \text{ are vessels} \\ 0, & \text{otherwise} \end{cases}$$

8.4.3 Objective and constraints

The objective function (8.1) is the minimization of the total assignment cost, which is the sum of the cost of waiting before berthing and the delay cost for each vessel. Constraints (8.2) ensure that the berthing of each vessel occurs on or after its arrival, while constraints (8.3) ensure that it is assigned to a compatible berth. The release times of the berths are considered in constraints (8.4). Constraints (8.5) establish that the departure of each vessel must occur at the time its handling finishes or later, and constraints (8.6) define the delays. Constraints (8.7)–(8.9) define σ, γ, and ϕ variables. Constraints (8.10) prevent both overlaps in time for tuples of vessels assigned to the same berth (set C) and moorings related to adjacent berths (set A), opposite berths (set O) or berths and vessels subject to special considerations (set D). In particular, given a tuple that represents a mooring restriction, if all the vessels in the tuple are assigned to their corresponding berths in the tuple (right-hand side of the inequality), at least one pair of vessels cannot be served concurrently (left-hand side). The same applies to tuples representing blocking restrictions in constraints (8.11) and (8.12), although in this case the blockage can be avoided if the vessel assigned to the blockable berth (v_1) is moored earlier (summation of γ) and departs later (summation of ϕ). Thus, by constraint (8.8), the berthing time of v_1 must be less than or equal to the berthing time of at least one of the vessels concurrently being processed at the blocking berths, and likewise its departure time must be greater than or equal to the departure time of at least one of those vessels, by constraint (8.9). Finally, constraints (8.13)–(8.16) define

the type of the variables. In this formulation, the constant M is an upper bound on the total time required to service all the vessels calculated by means of the procedure proposed in the following section.

$$Min \sum_{i \in V} (C_i^w (t_i - a_i) + C_i^d u_i) \tag{8.1}$$

s. t.

$$t_i \geq a_i, \qquad \forall i \in V \tag{8.2}$$

$$\sum_{k \in B_i} m_i^k = 1, \qquad \forall i \in V \tag{8.3}$$

$$t_i \geq \sum_{k \in B_i} m_i^k rel_k, \qquad \forall i \in V \tag{8.4}$$

$$r_i \geq t_i + \sum_{k \in B_i} m_i^k h_i^k, \qquad \forall i \in V \tag{8.5}$$

$$u_i \geq r_i - s_i, \qquad \forall i \in V \tag{8.6}$$

$$t_j \geq r_i - M(1 - \sigma_{ij}), \qquad \forall i, j \in V, i \neq j \tag{8.7}$$

$$t_i \leq t_j + M(1 - \gamma_{ij}), \qquad \forall (i,j) \in P \tag{8.8}$$

$$r_i \geq r_j - M(1 - \phi_{ij}), \qquad \forall (i,j) \in P \tag{8.9}$$

$$\sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} \sigma_{v_i v_j} \geq \sum_{i=1}^{n} m_{v_i}^{b_i} - n + 1, \qquad \forall (b_1, v_1, \ldots, b_n, v_n) \in I \tag{8.10}$$

$$\sum_{j=2}^{n} \gamma_{v_1 v_j} + \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} \sigma_{v_i v_j} \geq \sum_{i=1}^{n} m_{v_i}^{b_i} - n + 1, \qquad \forall (b_1, v_1, \ldots, b_n, v_n) \in F \tag{8.11}$$

$$\sum_{j=2}^{n} \phi_{v_1 v_j} + \sum_{i=1}^{n} \sum_{j=1, i \neq j}^{n} \sigma_{v_i v_j} \geq \sum_{i=1}^{n} m_{v_i}^{b_i} - n + 1, \qquad \forall (b_1, v_1, \ldots, b_n, v_n) \in F \tag{8.12}$$

$$\sigma_{ij} \in \{0,1\}, \qquad \forall i, j \in V, i \neq j \tag{8.13}$$

$$\gamma_{ij}, \phi_{ij} \in \{0,1\}, \qquad \forall (i,j) \in P \tag{8.14}$$

$$m_i^k \in \{0,1\}, \qquad \forall i \in V, \forall k \in B_i \tag{8.15}$$

$$t_i, r_i, u_i \geq 0, \qquad \forall i \in V \tag{8.16}$$

In order to reinforce the formulation, the following valid inequalities are also included:

$$\gamma_{ij} \geq \sigma_{ij}, \quad \forall (i,j) \in P \tag{8.17}$$

$$\phi_{ij} \geq \sigma_{ji}, \quad \forall (i,j) \in P \tag{8.18}$$

It is worth noting that this model can be applied in a rolling-horizon strategy by using the release time of the berths. In such strategies, the problem is decomposed into several time windows, so the berth planning is performed on the vessels arriving within

each time window independently. The release time can be used to connect consecutive time windows, thereby indicating that a vessel assigned at the end of a time window continues occupying a berth at the beginning of the next one.

Furthermore, this model can easily be extended to address a continuous BAP with multiple quays affected by similar restrictions on mooring and departing. To achieve this, it can be combined with the model proposed in Chapter 3, so that variable m_i^k refers to the assignment of vessel i to quay k, variable p_i determines the position of vessel i on its assigned quay, and variables δ_{ij} establish the relative position (above/below) of each pair of vessels i and j, according to the non-overlapping constraints considered in that model. Likewise, the objective function could include costs resulting from the assignment of vessels to quays.

8.4.4 Calculating upper bounds

A feasible solution for the problem can be easily obtained by sorting the vessels in non-decreasing arrival time and assigning them one after another in time to their corresponding most efficient berth, taking into account the release times of the berths (Algorithm 6). Thus no pair of vessels is served concurrently and all the constraints are trivially satisfied. The cost of this solution is an upper bound that may help solution methods based on branch-and-bound strategies. Moreover, this algorithm allows us to calculate a value of M adjusted to each instance. In particular, M is set to the maximum departure time among the vessels in the solution thus constructed. Algorithm 6 also returns this value.

Algorithm 6 Construction of a trivial feasible solution

Input: instance data
Output: a feasible solution
Output: M

1: Sort V by non-decreasing arrival time
2: $prevDeparture \leftarrow 0$ ▷ Auxiliary variable
3: **for** $i \in V$, according to the ordering **do**
4: Sort B_i lexicographically by 1) non-decreasing release time and 2) non-decreasing $h_i^{k \in B_i}$
5: $b \leftarrow$ first element in B_i ▷ Auxiliary variable
6: $m_i^b \leftarrow 1$
7: $t_i \leftarrow \max\{rel_b, a_i, prevDeparture\}$
8: $r_i \leftarrow t_i + h_i^b$
9: $prevDeparture \leftarrow r_i$
10: **end for**
11: $M \leftarrow prevDeparture$

8.5 Computational experiments

In order to evaluate the approach adopted, the model was implemented on a solver and several computational experiments were conducted over sets of instances created with different motivations.

8.5.1 Instance sets and implementation issues

Three sets of instances were generated: *Realistic*, *Realistic-Week*, and *Random*. The *Realistic* set was generated from real-world data kindly provided by a tank terminal located at the port of Antwerp. This set was designed to test the performance of the model in the context of a real terminal. It consists of 100 instances, 10 for each number of vessels considered: $N_V \in \{10, 20, \ldots, 100\}$. Each instance has the same data on the 11 berths in the terminal and their characteristics. All the berths are available from the beginning of the planning period ($rel_k = 0$). There are 10 pairs of opposite berths and 5 pairs of adjacent berths. The clearances required are $c_k^a = 10$ and $c_k^o = 30$ metres in all cases. The characteristics of the vessels (length, beam, draft, and deadweight tonnage) were calculated applying a random deviation obtained from $U[-20, 20]$ to the characteristics corresponding to 20 vessels that call regularly at the terminal. The arrival times were obtained from historical data, selecting a day at random and obtaining the next N_V arrivals. For each vessel, the compatible berths were precalculated according to the characteristics of both the vessel and each berth, including length, beam, draft, deadweight tonnage, and type of vessel. The handling times in the compatible berths were calculated applying a percentage of random deviation obtained from $U[-20, 20]$ to the difference between the historical arrival and departure times of the vessel. The desired departure time was generated in the same way, adding the arrival time. The cost coefficients were set to $C_i^w = 1$ and $C_i^d = 2$ for all the vessels. The set of tuples representing mooring restrictions (D) was generated according to a set of rules provided by the terminal. Rules may involve any characteristic of the vessels and the berths known in advance, such as the length or the deadweight tonnage. An example of a rule is: *"Any pair of vessels i and j cannot stay concurrently in berths 6 and 7, respectively, if: $l_i \geq 135\,\mathrm{m}$ and $l_j \geq 150\,\mathrm{m}$."* Similarly, the set of tuples representing blocking restrictions (F) was generated by means of rules such as: *"Any vessel i cannot berth on or depart from berth 2 if: $l_i \geq 110\,\mathrm{m}$ and either berth 3 or 4 are occupied."* A total of 23 rules were translated this way, 18 relating to mooring restrictions and 5 relating to blocking restrictions.

The previous set of instances was also used to generate another set, called *Realistic-Week*, so that in each instance vessels arrive within a time horizon of one week. This set makes it possible to evaluate the performance of the model under increasing vessel congestion. All the instances in the set *Realistic* were copied, but only those in which the arrival time of any vessel exceeded 168 hours were transformed. In those instances, the arrival time of each vessel i was changed according to: $new\_a_i = prev\_a_i \cdot prop$, where $prop = 168/\max_{j \in V}(prev\_a_j)$, rounded to the nearest integer. The desired departure time was also changed according to: $new\_s_i = prev\_s_i \cdot prop$, rounded to the nearest

integer. A total of 33 instances were changed this way: 2 instances with 60 vessels, 4 with 70 vessels, 8 with 80 vessels, 9 with 90 vessels, and all the instances with 100 vessels.

An additional set of instances, called *Random*, was generated considering random vessels and random adjacency, oppositional, and blocking restrictions in a terminal of a reasonable size. This set was aimed at assessing the performance of the model in a broader scope. It consists of 60 instances, 10 per $N_V \in \{10, 20, 30, 40, 50, 60\}$. Each instance contains 15 berths with release times equal to 0. In each one, four pairs of berths are randomly chosen to be adjacent, with separations in metres also randomly generated from $U[50, 400]$. This selection is performed preventing the formation of circular relations and considering that at most three berths can form a chain of adjacency relations. Likewise, four pairs of berths are selected randomly to be opposite berths, with separations obtained from $U[50, 200]$ and preventing both berths in the pair from being adjacent. All the adjacent berths require a clearance of 10 metres, while opposite berths require a clearance of 30 metres. Each instance also has four blocking restrictions, each with a different blockable berth randomly chosen. The set of blocking berths in each of these restrictions is also generated randomly, selecting a number of berths obtained from $U[1, 4]$ and preventing mutual blocking relations, that is, those in which a blockable berth is also a blocking berth of any of its blocking berths. Blocking relations involve all the combinations of vessels in the instance, so they do not refer to specific vessels. Each vessel i is also generated randomly, with its length in metres obtained from $U[30, 430]$ and its beam calculated as the nearest integer resulting from: $l_i^{2/3} + rand$, where $rand$ is obtained from $U[0, 5]$. Each vessel has a number of compatible berths obtained from $U[2, 8]$, which are chosen randomly. The arrival time (hour) was obtained from $U[0, 168]$; the processing time in each compatible berth, from $U[5, 20]$ hours; and the desired departure time is equal to $a_i + 1.25 \cdot \min_{k \in B_i}(h_i^k)$. The cost coefficients are the same for all the vessels: $C_i^w = 1, C_i^d = 2$, so no priorities are set between them.

The model was implemented in Java 1.7 and CPLEX 12.6 and run on an Intel Core i7 2600 at 3.4 GHz with 31.4 GiB of RAM running the operating system Ubuntu 14.04 GNU/Linux 3.13. CPLEX was configured with emphasis on optimality ($MIP\ emphasis = 2$). Moreover, for each instance the value of M and the upper bound were determined by means of Algorithm 6. In the next section, the results obtained on each set of instances are presented and discussed.

8.5.2 Results

The model was run on each instance for one hour. The results obtained in each set of instances are shown in Tables 8.1, 8.2, and 8.3. For each group of instances with the same number of vessels, these tables show the average computation time in seconds, including the time required to construct the model; the number of instances solved; the number of instances solved to optimality; the average and maximum gaps in percentage between the lower and the upper bounds obtained by the solver; and the average cost of the best solutions.

Table 8.1: Results on the set *Realistic*.

Vessels	Time	Solved	Optim.	Avg. gap	Max. gap	Avg. cost
10	0.2	10	10	0.0	0.0	12.0
20	1	10	10	0.0	0.0	34.1
30	732.4	10	8	7.3	49.1	185.2
40	1863.5	10	5	20.5	76.6	442.2
50	2751.9	10	3	45.0	91.0	595.5
60	3605.0	10	0	58.3	96.3	999.2
70	3293.2	10	1	64.5	96.9	2075.4
80	3615.2	9	0	74.9	96.3	1875.4
90	3622.9	10	0	79.3	97.1	3311.4
100	3637.6	10	0	83.9	98.7	4322.4

The results on the set *Realistic* (Table 8.1) show that the model is capable of solving optimally most of the real-world instances with up to 30 vessels and half of the instances with 40 vessels. Larger instances are occasionally solved to optimality, but in the cases in which an optimum is not obtained, both the average and the maximum gap are quite high. Nevertheless, the model can obtain feasible solutions in all the instances, except in one with 90 vessels. Instances with fewer than 30 vessels are solved in very short times, while larger instances require increasing computation effort, as was expected.

Table 8.2: Results on the set *Realistic-Week*.

Vessels	Time	Solved	Optim.	Avg. gap	Max. gap	Avg. cost
10	0.2	10	10	0.0	0.0	12.0
20	1	10	10	0.0	0.0	34.1
30	732.4	10	8	7.3	49.1	185.2
40	1863.5	10	5	20.5	76.6	442.2
50	2751.9	10	3	45.0	91.0	595.5
60	3605.0	10	0	62.1	96.3	1020.9
70	3609.3	10	0	73.5	96.9	2315.8
80	3614.9	10	0	85.3	96.3	3996.7
90	3623.4	8	0	93.8	97.1	8568.3
100	3638.3	6	0	95.8	97.6	15574.8

In Table 8.2, the results obtained on the set *Realistic-Week* show that the congestion resulting from the arrival of more than 60 vessels within one week increases the difficulty of the problem. The average gap increases and in six instances with 90 and 100 vessels the model could not obtain a feasible solution. The average cost also increases, doubling in the instances with more than 70 vessels.

Table 8.3: Results on the set *Random*.

Vessels	Time	Solved	Optimum	Avg. cost
10	0.2	10	10	0.4
20	2.2	10	10	2.9
30	35.7	10	10	7.1
40	419.3	10	10	14.4
50	533	9	9	14
60	348	1	1	20

Table 8.3 shows the results obtained on the set *Random*. The columns reporting the gaps are omitted because all the instances for which a solution was obtained were solved optimally. In one instance with 50 vessels and 9 instances with 60 vessels, the solver could not construct the model due to lack of memory, as a consequence of the great number of variables and constraints. The average computation times reported do not consider those instances. In general, the results are better than in the other sets of instances, so this set seems less difficult to solve. The model was able to achieve optimal solutions for most instances with up to 50 vessels in less than 10 minutes, which shows that the specific relations between berths (randomly chosen) do not increase the difficulty of the problem. The difference in performance between the set *Realistic* and the set *Random* seems to be due to the proportion of restrictions between berths relative to the number of berths available for the vessels, which is greater in the set *Realistic*. Further experiments will be conducted in the future to better determine the contribution of each factor to the complexity of the problem.

In summary, the model proposed is a good method for solving realistic instances with up to 40–50 vessels arriving within one week. Instances with up to 30 vessels can be solved in a few minutes, while larger instances may require up to one hour.

8.6 Concluding remarks

In this chapter, a new variant of the BAP arising in terminals with irregular layouts has been addressed. In particular, the problem has been formulated by means of a new mixed integer linear model, which is capable of representing a great variety of restrictions on vessel mooring and departure, including those derived from adjacency, oppositional and blocking relations between berths. This model can be easily extended to represent other discrete, continuous, and hybrid BAPs involving restrictions of this kind. Moreover, it can be integrated into rolling-horizon strategies without requiring additional adjustments.

The results obtained on realistic and randomly-generated instances show that in less than one hour the model obtains optimal solutions for instances with up to 40–50 vessels arriving within a one-week horizon, while instances with up to 30 vessels can be solved optimally in a few minutes. Therefore, the model is a good approach to tackling the BAP considering various restrictions involving the berths and the vessels.

Chapter 9

Conclusions and further research directions

In this thesis, different variants of the Berth Allocation Problem and the Berth Allocation and Quay Crane Assignment Problem have been tackled considering well-known and novel real-world constraints. Given that these combinatorial optimization problems are known to be $\mathcal{NP}$-hard, both exact and heuristic solution methods have been proposed to deal with realistic-size instances thereof. This chapter summarizes the main contributions (Section 9.1), proposes further research directions (Section 9.2) and presents the projects within which this thesis has been performed and the derived works (Section 9.3).

9.1 Contributions

In Chapter 3, the dynamic Berth Allocation Problem considering multiple continuous quays was studied for the first time at the operational level. A terminal with several quays is a problem for the continuous version, because in this variant of the problem the terminal is considered as one straight line on which vessels can be berthed according to their length and the positions of the other vessels. In a terminal with multiple quays, in addition to the berthing time and position on the quay, each vessel has to be assigned to one of the quays. Besides the usual costs of waiting to be berthed and of delay with respect to the ideal departure time, two additional costs were considered: the cost of assigning vessels to quays (which can be used to forbid some vessel-quay combinations) and a deviation cost with respect to the ideal position of the vessel on the quay to which it is actually assigned. This problem was tackled by means of a new mixed integer linear model, several priority rules, and a Genetic Algorithm based on ordered lists of vessels, for which four constructive algorithms (two heuristics and two matheuristics) and a Local Search procedure were proposed. The experiments conducted on well-known and new comprehensive sets of instances showed that in less than one hour the model can obtain optimal solutions on instances with up to 40 vessels arriving within one week, while the Genetic Algorithm is able to solve instances with up to 200 vessels and outperforms other heuristics proposed for the continuous BAP with a single quay. Furthermore,

an additional experiment showed that the complexity of the problem increases with increasing number of vessels, vessel length, vessel handling time, or congestion factor, while it decreases with increasing number of quays.

Chapter 4 addressed the deep integration of the dynamic continuous Berth Allocation Problem and the time-invariant Quay Crane Assignment Problem (BACAP). In this integrative problem, in addition to the berthing time and position on the quay, each vessel is to be assigned a number of cranes, assuming that it is kept fixed throughout the handling of the vessel. This version of the problem produces fewer crane changes between vessels than the variable-in-time crane assignment variant and therefore leads to more reliable berth plans. The problem was formulated by means of a new mixed integer linear model, and several families of valid inequalities were proposed to reinforce the initial formulation. In contrast to other state-of-the-art formulations, neither the quay length nor the lengths of the vessels were discretized in the model; instead, a continuous variable for the berthing positions of the vessels was used, thereby making the model truly continuous. Consequently, its performance does not depend on the discretization factor used for the lengths. The computational experiments conducted showed that the model outperforms other approaches proposed in the literature and is able in less than one hour to obtain optimal or near-optimal solutions on instances with up to 50 vessels arriving within one week.

This model is extended in Chapter 5 to solve the Berth Allocation and Specific Quay Crane Assignment Problem (BACASP). In this problem, instead of a number of cranes, a set of specific cranes is to be assigned to each vessel, assuming that it cannot be changed during the handling of the vessel and cranes cannot cross each other. Unlike the problem addressed in the previous chapter, this problem does not rely on the use of postprocessing algorithms that assign the specific cranes to the vessels, which produce additional crane changes between vessels and thus may lead to less reliable plans. A new cutting plane algorithm based on the BACAP model was also proposed in addition to the BACASP model. According to the results of the computational experiments, in less than one hour the model is capable of optimally solving instances with up to 35 vessels arriving within one week, while the cutting plane algorithm can optimally solve instances with up to 40 vessels requiring short computation times, thereby outperforming other state-of-the-art approaches. Both methods show stable behaviour regardless of the discretization factor applied to the lengths of the quays and the vessels.

In order to address the increasing demand for fast methods for solving these problems under the pressure of increasing vessel traffic and congestion, the previous exact approaches were complemented with new heuristic methods. In particular, in Chapter 6 a Biased Random-Key Genetic Algorithm with Memetic characteristics and several Local Search procedures were proposed to obtain good solutions on large BACAP instances in short times. Moreover, a matheuristic based on the BACAP model was developed to deal with the cases in which the solutions obtained by the Genetic Algorithm were difficult to improve on using ruin-and-recreate heuristics. These algorithms were adjusted and evaluated through extensive experiments and statistical analysis. The results were compared to those achieved by the exact methods and showed that in less than 5 minutes the Genetic Algorithm is able to achieve near-optimal solutions on instances

with up to 50 vessels, while it is able to obtain good solutions for instances of up to 100 vessels. The matheuristic Local Search proved to be the best option for improving the solutions obtained by the Genetic Algorithm.

In Chapter 7, these algorithms were extended to address the BACASP and achieved similar results in analogous experiments. Given that the continuous time-invariant BACASP assumes more real-world characteristics and constraints than other Berth Allocation Problems, the Genetic Algorithm for this problem is more practical and closer to the demands posed by container terminals than other state-of-the-art approaches.

Finally, Chapter 8 addressed for the first time a novel hybrid BAP arising in terminals with irregular layouts. This BAP takes into account restrictions on vessel mooring and departure depending upon the relations between the berths or special measures determined by the terminal operator. This includes restrictions posed by adjacent and opposite berths and others involving the potential blockage of a berth when certain vessels are moored in other berths. The problem was formulated by means of a new mixed integer linear model that can easily be adapted to represent other discrete, continuous or hybrid BAPs and integrated into rolling-horizon strategies. The experiments conducted in real-world and randomly-generated instances showed that in less than one hour the model proposed can obtain optimal solutions for instances with up to 40-50 vessels arriving within one week, while instances with up to 30 vessels can be solved optimally in a few minutes.

In conclusion, the mathematical programming models and the heuristic algorithms proposed in this thesis proved to be effective and efficient methods for solving realistic instances of the berth allocation problems confronted in port terminals. These methods outperform other state-of-the-art approaches to well-known versions of the problems and are capable of dealing with novel variants arising in real-world terminals.

9.2 Further research directions

This research can be further continued to address other aspects and variants of the problems.

The continuous Berth Allocation Problem with multiple quays studied in Chapter 3 could be extended to consider also the assignment of quay cranes, thereby leading to the novel continuous BACAP and BACASP with multiple quays. Moreover, besides the time-invariant crane assignment tackled in this thesis, new methods could be developed to obtain reliable plans for the variable-in-time version of the problem. This would involve minimizing the number of crane movements and the time thus spent, since they are major sources of schedule unreliability. Additionally, the moving range of cranes could also be taken into account, thus addressing the limitations posed by the characteristics of some terminals or the specific operational restrictions established by terminal operators.

These problems could also be tackled at a tactical level, following the previous studies on this line. Instead of considering a list of actual vessels, with their specific characteristics, the problems would consider shipping lines and sets of vessels cyclically calling at the terminal. Those ships would have to be assigned to several quays, while a number of cranes, or even specific sets of cranes, would have to be reserved for their handling.

All these problems could also be studied considering the kind of restrictions on vessel mooring and departure arising in terminals with irregular quays, as we saw in Chapter 8. Indeed, as the integer model proposed can be naturally extended to solve the continuous BAP with multiple quays, additional experiments could be conducted to assess its performance under such assumptions. Likewise, further experiments are required to determine the contribution of each factor (number of berths, number of restrictions of each type, etc.) to the complexity of the problem. The model proposed could also be complemented with a heuristic approach to obtain better solutions for large instances.

Another possibility would be to study these problems jointly with the Quay Crane Scheduling Problem to devise a general solution method for seaside operational planning. Thus the assignment of cranes to each vessel would also consider the various alternatives of assigning specific loading and unloading tasks to each crane, which may lead to better global solutions.

Furthermore, instead of combining all the costs into a single objective function, conflicting objectives could be studied separately by developing multi-objective procedures, thereby providing terminal operators with a set of non-dominated solutions that could also be useful for managerial decision making.

In the light of the novel developments in the field, it also would be interesting to introduce stochastic factors in order to improve the robustness of the solutions when faced with uncertainty and unexpected contingencies, such as crane breakdowns or adverse weather conditions. Moreover, fuel consumption and emissions could be considered as additional costs incurred by the terminal and the vessels, thus addressing the challenges posed by global warming and our heavy dependency on fossil fuels.

Besides the extension of these problems, the field needs to address the lack of standard and well-designed sets of instances. New investigations focusing on developing comprehensive benchmark instances are required to better assess and compare the approaches proposed by researchers. These studies should broadly cover the variability of the main parameters and provide statistical analysis to reveal how they affect the complexity of the problem.

All these efforts would enhance the methods aimed at improving the efficiency and quality of service in port terminals and in so doing they would provide new insights and solution strategies for combinatorial optimization problems similar to those tackled in this investigation.

9.3 Projects and derived works

The research presented in this thesis was conducted in the ambit of various projects from October 2013 to October 2016 and has already given rise to some international publications.

9.3.1 Research projects

This investigation was part of the following projects:

- **Modelos y Algoritmos para problemas de Optimización Combinatoria**
  Funded by Conselleria d'Educació, Formació i Ocupació, Generalitat Valenciana
  Reference: GV-PROMETEO/2013/049
- **Optimización de procesos en terminales marítimas de contenedores**
  Funded by Ministerio de Economía y Competitividad, Gobierno de España
  Reference: DPI2014-53665-P

9.3.2 International collaborations

An international collaboration was established with Prof. Dr. Greet Vanden Berghe and the Combinatorial Optimisation and Decision Support research group (CODeS) through a four-month research visit in 2016 to KU Leuven in Ghent, Belgium. During this visit I analysed a special Berth Allocation Problem arising in a tank terminal located at the port of Antwerp. I then proposed and developed an exact approach for this problem, which is described in Chapter 8. This research was possible thanks to the grant BEFPI-2016 of the Conselleria d'Educació, Formació i Ocupació of the Generalitat Valenciana (Spain).

9.3.3 Derived works

Some of the main contributions appearing in this thesis have been submitted or accepted for publication. The corresponding bibliographical references are enumerated below.

- Correcher, J.F.; Alvarez-Valdes, R. 2017. **A Biased Random-Key Genetic Algorithm for the Berth Allocation and Quay Crane Assignment Problem**. Technical report, Department of Statistics and Operations Research, University of Valencia, Spain, March 2017. Submitted.
  URL: `http://www.optimization-online.org/DB_HTML/2017/03/5927.html`
- Correcher, J.F.; Alvarez-Valdes, R.; Tamarit, J.M. 2017. **A New Mixed Integer Linear Model for the Berth Allocation and Quay Crane Assignment Problem**. Technical report, Department of Statistics and Operations Research, University of Valencia, Spain, March 2017. Submitted.
  URL: `http://www.optimization-online.org/DB_HTML/2017/03/5890.html`
- Froján, P.; Correcher, J.F.; Alvarez-Valdes, R.; Koulouris, G.; Tamarit, J.M. 2015. **The Continuous Berth Allocation Problem in a Container Terminal with Multiple Quays**. *Expert Systems With Applications*, 42(21), pp. 7356–7366.

Another relevant problem has been addressed during the preparation of this thesis. Although it is not directly related to berth planning, it is also of interest for the container transportation field. The publication derived from this investigation is presented here for reference purposes.

- Correcher, J.F.; Alonso M.T.; Parreño, F.; Alvarez-Valdes, R. 2017. **Solving a large multicontainer loading problem in the car manufacturing industry**. *Computers & Operations Research*, 82, pp. 139–152.

9.3.4 Conference contributions

Some of the main contributions were also presented in the following conferences:

- Correcher, J.F.; Alvarez-Valdes, R.; Tamarit, J.M. 2016. Un modelo lineal entero para el problema de la asignación de atraques y grúas en terminales de contenedores. *XXXVI Congreso Nacional de Estadística e Investigación Operativa, SEIO 2016*. Universidad de Castilla la Mancha, Toledo, Spain.

- Correcher, J.F.; Alvarez-Valdes, R. Tamarit, J.M. 2016. A New Random-Key Genetic Algorithm for the Berth Allocation and Quay Crane Assignment Problems. *28th European Conference of Operations Research, EURO 2016*. Poznań University of Technology, Poznań, Poland.

- Correcher, J.F.; Alvarez-Valdes, R. 2016. The Berth Allocation and Quay Crane Assignment Problem: a new integer linear model. *15th lnternational Conference on Project Management and Scheduling, PMS 2016*. Universitat de València and Universitat Politècnica de València, València, Spain.

- Correcher, J.F.; Alvarez-Valdes, R.; Tamarit, J.M.; Lescaylle, A. 2015. Modelos y algoritmos para el problema de la asignación de atraques y grúas en las terminales de contenedores. *3er Workshop sobre Metaheurísticas Inteligentes en la Planificación Logística, MIPL, en Congreso de la Asociación Española de Inteligencia Artificial, CAEPIA 2015*. Universidad de Castilla la Mancha, Albacete, Spain.

- Correcher, J.F.; Alvarez-Valdes, R.; Tamarit, J.M. 2015. Models and Algorithms for the Berth Allocation and Time-invariant Quay Crane Assignment Problem. *International Workshop on Cutting, Packing and Related Topics, IWCPRT 2015*. Guimarães, Portugal.

- Correcher, J.F.; Alvarez-Valdes, R.; Tamarit, J.M. 2015. A new mixed integer linear model for berth allocation and time-invariant quay crane assignment problems. *27th European Conference of Operations Research, EURO 2015*. University of Starthclyde, Glasgow, United Kingdom.

- Correcher, J.F.; Froján, P.; Alvarez-Valdes, R.; Koulouris, G.; Tamarit, J.M. 2015. Modelos y algoritmos para el problema de la asignación de atraques en una terminal de contenedores. *Actas del X Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, MAEB 2015*, Universidad de Extremadura, Merida, Spain, pp. 289–296.

- Correcher, J.F.; Alvarez-Valdes, R.; Tamarit, J.M. 2014. The Continuous Berth Allocation Problem considering Multiple Quays: Models and Algorithms. *The International Conference on Logistics and Maritime Systems, LOGMS 2014*. Erasmus School of Economics, Rotterdam, Netherlands.

Bibliography

Barros, V. H., Costa, T. S., Oliveira, A. C. M., and Lorena, L. A. N. 2011. Model and heuristic for berth allocation in tidal bulk ports with stock level constraints. *Computers and Industrial Engineering* 60:606–613.

Bean, J. C. 1994. Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing* 6:154–160.

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

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

Blazewicz, J., Cheng, T. C. E., Machowiak, M., and Oğuz, C. 2011. Berth and quay crane allocation: a moldable task scheduling model. *Journal of the Operational Research Society* 62:1189–1197.

Bridi, I., Rodrigues, G., Rosa, R. D. A., Gomes, T. C., and Ribeiro, G. M. 2016. Mathematical model for the Berth Allocation Problem in ports with cargo operation limitations along the pier. *Gestão & Produção* 23:771–786.

Brinkmann, B. 2011. Operations Systems of Container Terminals: A Compendious Overview. *In* J. W. Böse (ed.), Handbook of Terminal Planning, chapter 2. Springer Science.

Carlo, H. J., Vis, I. F. A., and Roodbergen, K. J. 2014a. Storage yard operations in container terminals: Literature overview, trends, and research directions. *European Journal of Operational Research* 235:412–430.

Carlo, H. J., Vis, I. F. A., and Roodbergen, K. J. 2014b. Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. *European Journal of Operational Research* 236:1–13.

Carlo, H. J., Vis, I. F. A., and Roodbergen, K. J. 2015. Seaside operations in container terminals: literature overview, trends, and research directions. *Flexible Services and Manufacturing Journal* 27:224–262.

Caserta, M., Schwarze, S., and Voss, S. 2011. Container Rehandling at Maritime Container Terminals. *In* J. W. Böse (ed.), Handbook of Terminal Planning, chapter 13. Springer Science.

Caserta, M. and Voss, S. 2010. Metaheuristics: Intelligent Problem Solving. *In* Matheuristics. Hybridizing Metaheuristics and Mathematical Programming, chapter 1. Springer.

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

Chen, D. S., Batson, R. G., and Dang, Y. 2011. Applied Integer Programming: Modeling and Solution. Wiley.

Chen, J. H., Lee, D. H., and Cao, J. X. 2012. A combinatorial Benders' cuts algorithm for the quayside operation problem at container terminals. *Transportation Research Part E: Logistics and Transportation Review* 48:266–275.

Cheong, C. Y., Tan, K. C., Liu, D. K., and Lin, C. J. 2010. Multi-objective and prioritized berth allocation in container ports. *Annals of Operations Research* 180:63–103.

Chung, C.-c. and Chiang, C.-h. 2011. The Critical Factors: An Evaluation of Schedule Reliability in Liner Shipping. *International Journal of Operations Research* 8:3–10.

Conforti, M., Cornuéjols, G., and Zambelli, G. 2014. Integer Programming. Springer.

Cordeau, J.-F., Laporte, G., Legato, P., and Moccia, L. 2005. Models and Tabu Search Heuristics for the Berth-Allocation Problem. *Transportation Science* 39:526–538.

Du, Y., Chen, Q., Quan, X., Long, L., and Fung, R. Y. K. 2011. Berth allocation considering fuel consumption and vessel emissions. *Transportation Research Part E: Logistic and Transportation Review* 47:1021–1037.

Elwany, M. H., Ali, I., and Abouelseoud, Y. 2013. A heuristics-based solution to the continuous berth allocation and crane assignment problem. *Alexandria Engineering Journal* 52:671–677.

Ericsson, M. and Pardalos, P. M. 2002. A Genetic Algorithm for the Weight Setting Problem in OSPF Routing. *Journal of Combinatorial Optimization* 6:299–333.

Ernst, A.T., Oğuz, C., Singh, G., and Taherkhani, G. 2017. Mathematical models for the berth allocation problem in dry bulk terminals. *Journal of Scheduling*. In press. doi:10.1007/s10951-017-0510-8.

Esmer, S. 2008. Performance Measurements of Container Terminal Operations. *Dokuz Eylul University Journal of Graduate School of Social Sciences* 10:238–255.

Gass, S. I. and Assad, A. A. 2005. An Annotated Timeline of Operations Research. Springer.

Gass, S. I. and Fu, M. C. (eds.) 2013. Encyclopedia of Operations Research and Management Science. Springer, 3rd edition.

Gendreau, M. and Potvin, J.-Y. (eds.) 2010. Handbook of Metaheuristics. Springer.

Gharehgozli, A. H., Roy, D., and de Koster, R. 2016. Sea container terminals: New technologies and OR models. *Maritime Economics & Logistics* 18:103–140.

Gonçalves, J. F. and Resende, M. G. C. 2012. A parallel multi-population biased random-key genetic algorithm for a container loading problem. *Computers and Operations Research* 39:179–190.

González, M. M. and Trujillo, L. 2008. Efficiency measurement in the port industry: A survey of the empirical evidence. *Journal of Transport Economics and Policy* 43:157–192.

Goodchild, A., Zhao, W., and Wygonik, E. 2011. Decision Problems and Applications of Operations Research at Marine Container Terminals. *In* J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith (eds.), Wiley Encyclopedia of Operations Research and Management Science. Wiley.

Graeber, D. 2012. Debt: The First 5000 Years. Melville House.

Grammenos, C. T. (ed.) 2010. The Handbook of Maritime Economics and Business. Lloyd's list, 2nd edition.

Grubišić, N., Hess, S., and Hess, M. 2014. A solution of berth allocation problem in inland waterway ports. *Tehnički vjesnik* 5:1135–1141.

Guan, Y., Xiao, W. Q., Cheung, R. K., and Li, C. L. 2002. A multiprocessor task scheduling model for berth allocation: heuristic and worst-case analysis. *Operations Research Letters* 30:343–350.

Gudelj, A., Krcum, M., and Twrdy, E. 2010. Models and methods for operations in port container terminals. *Promet - Traffic & Transportation* 22:43–51.

Han, X., Gong, X., and Jo, J. 2015. A new continuous berth allocation and quay crane assignment model in container terminal. *Computers and Industrial Engineering* 89:15–22.

Hansen, P., Oğuz, C., and Mladenović, N. 2008. Variable neighborhood search for minimum cost berth allocation. *European Journal of Operational Research* 191:636–649.

Hansen, P. and Oğuz, C. 2003. A Note on Formulations of Static and Dynamic Berth Allocation Problems. *Les Cahiers du GERAD* 30:1–17.

Harris, M. and Johnson, O. 2007. Cultural Anthropology. Pearson, 7th edition.

He, J. 2016. Berth allocation and quay crane assignment in a container terminal for the trade-off between time-saving and energy-saving. *Advanced Engineering Informatics* 30:390–405.

Heaver, T. D. 2012. The Evolution of Maritime Economics. *In* W. K. Talley (ed.), The Blackwell Companion to Maritime Economics, chapter 2. Wiley-Blackwell.

Hendriks, M., Laumanns, M., Lefeber, E., and Udding, J. T. 2010. Robust cyclic berth planning of container vessels. *OR Spectrum* 32:501–517.

Hendriks, M. P. M., Armbruster, D., Laumanns, M., Lefeber, E., and Udding, J. T. 2012. Strategic allocation of cyclically calling vessels for multi-terminal container operators. *Flexible Services and Manufacturing Journal* 24:248–273.

Hillier, F. S. and Lieberman, G. J. 2015. Introduction to Operations Research. McGraw-Hill Education, 10th edition.

Hu, Q.-M., Hu, Z.-H., and Du, Y. 2014. Berth and quay-crane allocation problem considering fuel consumption and emissions from vessels. *Computers & Industrial Engineering* 70:1–10.

Hu, Z. H. 2015. Heuristics for solving continuous berth allocation problem considering periodic balancing utilization of cranes. *Computers and Industrial Engineering* 85:216–226.

Imai, A., Nishimura, E., Hattori, M., and Papadimitriou, S. 2007. Berth allocation at indented berths for mega-containerships. *European Journal of Operational Research* 179:579–593.

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

Imai, A., Nishimura, E., and Papadimitriou, S. 2013. Marine container terminal configurations for efficient handling of mega-containerships. *Transportation Research Part E: Logistics and Transportation Review* 49:141–158.

Iris, Ç., Pacino, D., Ropke, S., and Larsen, A. 2015. Integrated Berth Allocation and Quay Crane Assignment Problem: Set partitioning models and computational results. *Transportation Research Part E: Logistics and Transportation Review* 81:75–97.

Islam, S. and Olsen, T. L. 2013. Operations Research (OR) at ports: An update. *In* Proceedings of the 22nd National Conference of the Australian Society for Operations Research (ASOR 2013), pp. 91–97.

Journal of Commerce 2014. Berth productivity: The Trends, Outlook and Market Forces Impacting Ship Turnaround Times. Technical Report July, Journal of Commerce Group.

Karam, A. and Eltawil, A. B. 2016. Functional integration approach for the berth allocation, quay crane assignment and specific quay crane assignment problems. *Computers and Industrial Engineering* 102:458–466.

Kim, K. H. and Lee, H. 2015. Container Terminal Operation: Current Trends and Future Challenges. *In* C.-Y. Lee and Q. Meng (eds.), Handbook of Ocean Container Transport Logistics, chapter 2. Springer.

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

Kordić, S., Davidović, T., Kovač, N., and Dragović, B. 2015. Combinatorial approach to exactly solving discrete and hybrid berth allocation problem. *Applied Mathematical Modelling* 40:8952–8973.

Korte, B. and Vygen, J. 2012. Combinatorial Optimization. Theory and Algorithms. Springer, 5th edition.

Kuhn, T. S. 1992. The Copernican Revolution. Harvard University Press.

Laguna, M. and Martí, R. 2013. Heuristics, pp. 695–703. *In* S. I. Gass and M. C. Fu (eds.), Encyclopedia of Operations Research and Management Science. Springer, 3rd edition.

Lalla-Ruiz, E., Expósito-Izquierdo, C., Melián-Batista, B., and Moreno-Vega, J. M. 2016. A Set-Partitioning-based model for the Berth Allocation Problem under Time-Dependent Limitations. *European Journal of Operational Research* 250:1001–1012.

Lalla-Ruiz, E., González-Velarde, J. L., Melián-Batista, B., and Moreno-Vega, J. M. 2014. Biased random key genetic algorithm for the Tactical Berth Allocation Problem. *Applied Soft Computing* 22:60–76.

Lalla-Ruiz, E., Melián-Batista, B., and Marcos Moreno-Vega, J. 2012. Artificial intelligence hybrid heuristic based on tabu search for the dynamic berth allocation problem. *Engineering Applications of Artificial Intelligence* 25:1132–1141.

Le, M., Wu, C., and Zhang, H. 2012. An integrated optimization method to solve the berth-QC allocation problem. *In* Proceedings of the 2012 8th International Conference on Natural Computation (ICNC 2012), pp. 753–757. IEEE.

Le, M. N., Ong, Y. S., Jin, Y., and Sendhoff, B. 2009. Lamarckian memetic algorithms: Local optimum and connectivity structure analysis. *Memetic Computing* 1:175–190.

Lee, D. H., Chen, J. H., and Cao, J. X. 2010. The continuous Berth Allocation Problem: A Greedy Randomized Adaptive Search Solution. *Transportation Research Part E: Logistics and Transportation Review* 46:1017–1029.

Lee, D. H., Jin, J. G., and Chen, J. H. 2012. Terminal and yard allocation problem for a container transshipment hub with multiple terminals. *Transportation Research Part E: Logistics and Transportation Review* 48:516–528.

Lehnfeld, J. and Knust, S. 2014. Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *European Journal of Operational Research* 239:297–312.

Levinson, M. 2008. The Box: How the Shipping Container Made the World Smaller and the World Economy Bigger. Princeton University Press.

Li, C.-l., Cai, X., and Lee, C.-y. 1998. Scheduling with multiple-job-on-one-processor pattern. *IIE Transactions* 30:433–445.

Li, W., Wu, Y., and Goh, M. 2015. Planning and Scheduling for Maritime Container Yards. Springer.

Liang, X., Li, W., Zhao, W., and Li, B. 2012. Multistage collaborative scheduling of berth and quay crane based on heuristic strategies and particle swarm optimization. *In* Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 913–918. IEEE.

Lim, A. 1998. The berth planning problem. *Operations Research Letters* 22:105–110.

Maniezzo, V., Stützle, T., and Voss S. (eds.) 2010. Matheuristics. Hybridizing Metaheuristics and Mathematical Programming. Springer.

Maxwell, S. E. and Delaney, H. D. 2004. Designing Experiments and Analyzing Data: a Model Comparison Perspective. Lawrence Erlbaum Associates, 2nd edition.

Meersman, H. and de Voorde, E. V. 2010. Port Management, Operation and Competition: A Focus on North Europe. *In* C. T. Grammenos (ed.), The Handbook of Maritime Economics and Business, chapter 30. Lloyd's list, 2nd edition.

Meersmans, P. and Dekker, R. 2001. Operations Research Supports Container Handling. *European Journal of Operational Research* 22:1–22.

Meisel, F. 2009. Seaside Operations Planning in Container Terminals. Springer.

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

Meisel, F. and Bierwirth, C. 2013. A Framework for Integrated Berth Allocation and Crane Operations Planning in Seaport Container Terminals. *Transportation Science* 47:131–147.

Naredo, J. M. 2015. La economía en evolución. Historia y perspectivas de las categorías básicas del pensamiento económico. Siglo XXI, 4th edition.

Nishimura, E., Imai, A., and Papadimitriou, S. 2001. Berth allocation planning in the public berth system by genetic algorithms. *European Journal of Operational Research* 131:282–292.

Notteboom, T. 2012. Container Shipping. *In* W. K. Talley (ed.), The Blackwell Companion to Maritime Economics, chapter 12. Wiley-Blackwell.

Notteboom, T. E. 2006. The Time Factor in Liner Shipping Services. *Maritime Economics & Logistics* 8:19–39.

Panayides, P. M., Maxoulis, C. N., Wang, T.-F., and Ng, K. Y. A. 2009. A Critical Analysis of DEA Applications to Seaport Economic Efficiency Measurement. *Transport Reviews: A Transnational Transdisciplinary Journal* 29:183–206.

Park, K. T. and Kim, K. H. 2002. Berth scheduling for container terminals by using a sub-gradient optimization technique. *Journal of the Operational Research Society* 53:1054–1062.

Park, Y. and Kim, K. 2003. A scheduling method for berth and quay cranes. *OR Spectrum* 25:1–23.

Polanyi, K. 2001. The Great Transformation: The Political and Economic Origins of Our Time. Beacon Press, 2nd edition.

Qin, T., Du, Y., and Sha, M. 2016. Evaluating the solution performance of IP and CP for berth allocation with time-varying water depth. *Transportation Research Part E: Logistics and Transportation Review* 87:167–185.

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

Rashidi, H. and Tsang, E. P. K. 2013. Novel constraints satisfaction models for optimization problems in container terminals. *Applied Mathematical Modelling* 37:3601–3634.

Ribeiro, G. M., Mauri, G. R., De Castro Beluco, S., Lorena, L. A. N., and Laporte, G. 2016. Berth allocation in an ore terminal with demurrage, despatch and maintenance. *Computers and Industrial Engineering* 96:8–15.

Robenek, T., Umang, N., Bierlaire, M., and Ropke, S. 2014. A branch-and-price algorithm to solve the integrated berth allocation and yard assignment problem in bulk ports. *European Journal of Operational Research* 235:399–411.

Rodriguez-Molins, M., Salido, M. A., and Barber, F. 2014a. A GRASP-based metaheuristic for the Berth Allocation Problem and the Quay Crane Assignment Problem by managing vessel cargo holds. *Applied Intelligence* 40:273–290.

Rodriguez-Molins, M., Salido, M. A., and Barber, F. 2014b. Robust Scheduling for Berth Allocation and Quay Crane Assignment Problem. *Mathematical Problems in Engineering* 2014:1–17.

Salim, A. M. 2015. Review on the Role of Ports in the Development of a Nation. *Aquatic Procedia* 4:295–301.

Sciomachen, A., Acciaro, M., and Liu, M. 2009. Operations research methods in maritime transport and freight logistics. *Maritime Economics & Logistics* 11:1–6.

Shang, X. T., Cao, J. X., and Ren, J. 2016. A robust optimization approach to the integrated berth allocation and quay crane assignment problem. *Transportation Research Part E: Logistics and Transportation Review* 94:44–65.

Siarry, P. (ed.) 2016. Metaheuristics. Springer.

Solís, C. and Sellés, M. 2013. Historia de la ciencia. Espasa.

Sörensen, K. and Glover, F. 2013. Metaheuristics, pp. 960–970. *In* S. I. Gass and M. C. Fu (eds.), Encyclopedia of Operations Research and Management Science. Springer, 3rd edition.

Stahlbock, R. and Voss, S. 2008. Operations research at container terminals: A literature update. *OR Spectrum* 30:1–52.

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

Stopford, M. 2009. Maritime Economics. Routledge, 3rd edition.

Suárez-Alemán, A., Morales Sarriera, J., Serebrisky, T., and Trujillo, L. 2016. When it comes to container port efficiency, are all developing regions equal? *Transportation Research Part A: Policy and Practice* 86:56–77.

Talley, W. K. (ed.) 2012. The Blackwell Companion to Maritime Economics. Wiley-Blackwell.

Tang, L. C., Low, J. M. W., and Lam, S. W. 2011. Understanding Port Choice Behavior-A Network Perspective. *Networks and Spatial Economics* 11:65–82.

Theofanis, S., Boile, M., and Golias, M. M. 2009. Container Terminal Berth Planning. *Transportation Research Record: Journal of the Transportation Research Board* 2100:22–28.

Ting, C.-J., Wu, K.-C., and Chou, H. 2014. Particle swarm optimization algorithm for the berth allocation problem. *Expert Systems with Applications* 41:1543–1550.

Tongzon, J. and Heng, W. 2005. Port privatization, efficiency and competitiveness: Some empirical evidence from container ports (terminals). *Transportation Research Part A: Policy and Practice* 39:405–424.

Türkoğullari, Y. B., Taşkin, Z. C., Aras, N., and Altinel, I. K. 2014. Optimal berth allocation and time-invariant quay crane assignment in container terminals. *European Journal of Operational Research* 235:88–101.

Türkoğullari, Y. B., Taşkin, Z. C., Aras, N., and Altinel, I. K. 2016. Optimal berth allocation, time-variant quay crane assignment and scheduling with crane setups in container terminals. *European Journal of Operational Research* 254:985–1001.

Umang, N., Bierlaire, M., and Vacca, I. 2013. Exact and heuristic methods to solve the berth allocation problem in bulk ports. *Transportation Research Part E: Logistics and Transportation Review* 54:14–31.

UNCTAD 1976. Review of Maritime Transport. United Nations Conference on Trade and Development.

UNCTAD 2012. Review of Maritime Transport. United Nations Conference on Trade and Development.

UNCTAD 2015. Review of Maritime Transport. United Nations Conference on Trade and Development.

UNCTAD 2016. Review of Maritime Transport. United Nations Conference on Trade and Development.

UNCTAD 2017. UNCTAD Stat Website. URL: http://unctadstat.unctad.org.

Ursavas, E. and Zhu, S. X. 2016. Optimal policies for the berth allocation problem under stochastic nature. *European Journal of Operational Research* 255:380–387.

Vallada, E. and Ruiz, R. 2011. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research* 211:612–622.

van Marle, G. 2015. Measuring port performance. *The Loadstar Long Read* 1.

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

Wang, S., Meng, Q., and Liu, Z. 2013. A note on "Berth allocation considering fuel consumption and vessel emissions". *Transportation Research Part E: Logistics and Transportation Review* 49:48–54.

Woo, S.-H., Pettit, S. J., Beresford, A. K. C., and Kwak, D.-W. D.-W. 2012. Seaport Research: A Decadal Analysis of Trends and Themes Since the 1980s. *Transport Reviews* 32:351–377.

Woo, S. H., Pettit, S. J., Kwak, D. W., and Beresford, A. K. C. 2011. Seaport research: A structured literature review on methodological issues since the 1980s. *Transportation Research Part A: Policy and Practice* 45:667–685.

Xiao, L. and Hu, Z. H. 2014. Berth allocation problem with quay crane assignment for container terminals based on rolling-horizon strategy. *Mathematical Problems in Engineering* 2014:1–11.

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

Xuelian, C. and Zhiying, Y. 2012. An Algorithm for Continuous Berth Allocation with Quay Crane Dynamic Allocation. *In* Proceedings of the 2012 Fifth International Conference on Intelligent Computation Technology and Automation, pp. 541–544. IEEE.

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

Zhang, C., Zheng, L., Zhang, Z., Shi, L., and Armstrong, A. J. 2010. The allocation of berths and quay cranes by using a sub-gradient optimization technique. *Computers & Industrial Engineering* 58:40–50.

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

Zhen, L., Chew, E. P., and Lee, L. H. 2011. An Integrated Model for Berth Template and Yard Template Planning in Transshipment Hubs. *Transportation Science* 43:483–504.

Zhen, L., Jiang, X., Lee, L. H., and Chew, E. P. 2013. A Review on Yard Management in Container Terminals. *Industrial Engineering & Management Systems* 12:289–304.

Zhicheng, B., Chai, J., Mi, C., Shen, Y., and Xu, Z. 2013. Hybrid berth allocation problem with fake berths in busy coal terminal. *Information Technology Journal* 12:4610–4617.