UNIVERSITÀ DELLA CALABRIA
Dipartimento di ELETTRONICA, INFORMATICA E SISTEMISTICA

UNIVERSITÁ DELLA CALABRIA

DIPARTIMENTO DI ELETTRONICA INFORMATICA E SISTEMISTICA

Ph.D. Course in Operations Research (S.S.D. MAT/09)
XXII CYCLE

Ph.D. Thesis

Modelling and solving management problems in container and automotive maritime hub terminals

Gregorio Sorrentino

Advisors
M. Flavia Monaco
Luigi Moccia

Supervisor
Lucio Grandinetti

A.A. 2009-2010

a Teresa, Nino e Salvatore

Contents

Chapter 1

Maritime Hub Terminals, Operations and Optimization Problems

In the last decades maritime transportation of goods is increased enormously and, actually, accounts for at least 90 percent of global trade. Containerization has dramatically reduced the cost of freight handling. Further reduction of transportation costs are obtained by increasing the economies of scale building ever larger containerships for long-haul routes. In order to efficiently handle these big ships, mega-terminals, with appropriate facilities and technologies, are required. Due to the resulting network topology this system is known as hub and spoke. Deep-sea vessels operate between a limited number of transshipment terminals called hubs. Smaller feeder vessels link the hubs with the other ports which are the spokes of the system.

The transportation of finished vehicles has also grown impressively during the last years. Maritime automotive transportation is developing along the same lines of container transportation, therefore hub and spoke arrange-

ment is widely adopted (Mattfeld, 2006). This network topology results in the consolidation of capacity along the routes connecting the transshipment ports, and in the growth of their importance. Deep-sea car carriers have a capacity of up to 6000 vehicles whereas the capacity of ships deployed on short-sea segments can attain 1000 vehicles. Therefore, automotive transshipment terminals manage large flows of incoming and outgoing cars.

A hub terminal, either container or automotive, is an highly complex system that involves a lot of equipments, operations, and handling steps. Resources allocations and operations scheduling become crucial planning issues in such a system. According to the considered planning horizon, the decision problems arising in hub terminals can be of strategic, tactical and operational level. A time horizon in decisions for the strategic, tactical, and operational level covers one to several years, weeks to months, and one or few days, respectively.

This work faces some operational problems typical of maritime hub terminals, both container and automotive. The following sections describe the main key aspects of container and automotive hub terminals and the differences among these type of terminal.

1.1 Container terminal operations

A container port terminal provides transfer facilities for containers between a sea vessel and a truck or rail or between vessels, the last one is a single-mode transportation which is called *transshipment*. A container terminal can be viewed as an open system with two interfaces: the quayside, where ships load and unload containers, and the landside, where containers are loaded to and unloaded from trucks and trains.

To facilitate exchange between these interfaces a yard is used as a buffer where to stack containers. A ship arriving at the terminal is assigned to a

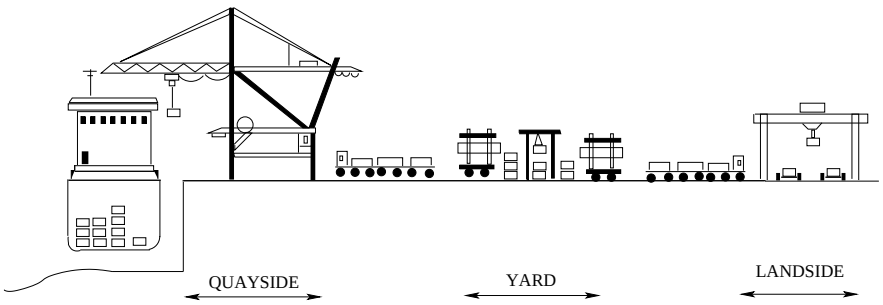

Figure 1.1: Schematic view of a container terminal system

berth, which is a portion of quay, equipped with cranes for loading and discharging containers.

The unloaded (import) containers are transported to their assigned yard positions by internal transportation equipment. Analogously, export containers are moved from yard positions to the quay in order to be loaded by quay cranes. From an operational point of view, we can think of a terminal as divided into two areas, the quayside and the yard, linked by an internal transportation system. Therefore there are decisional problems related to each of these areas and to the transportation system. Now we briefly describe some of these problems.

1.1.1 Berth planning

The berth allocation problem consists of assigning incoming ships to berthing positions. Berth planners have to decide where and when the ships have to be moored. There are technical constraints to consider in deciding where to moor: the length, the tonnage and the draft of the ship. Other constraints are related to particular equipment required by the ship. In fact, some new very wide ships, known as Post-Panamax ships, require appropriate cranes (Post-Panamax cranes) to load and discharge their containers.

The technical characteristics of the quay, as its length, the dock draft and the kind of cranes deployed, are fixed at the strategical level. A common criterion used by the terminal manager to allocate ships to berths is that the whole distance travelled by internal equipments to move containers from the yard to the quay (and vice versa), should be minimized (Steenken et al., 2004). On the other hand, the satisfaction of the ship owner can be reached by allocating the ship to berth in order to minimize turnaround time, that is the total time the ship stays in the port (Cordeau et al., 2005).

1.1.2 Crane split and scheduling

Solving the crane split and scheduling problem consists in allocating cranes to a ship and in deciding in which order the ship sections, called also ship bays, will be handled by cranes in such a way that all required transshipment of containers can be accomplished. The optimization aims either at minimizing the ships delay or at maximizing the crane utilization, i.e., minimizing the crane idle times. A feasible berth plan and a set of available cranes are input data. Furthermore, for each vessel, the number of containers to be discharged and/or loaded, as well as the maximum number of cranes allowed to simultaneously work on it, are known. The problem can be approached as a two phases process: the first one consists in assigning cranes to vessel and the second one in scheduling cranes operations (Bierwirth and Meisel, 2010). The quay crane assignment is not a difficult problem and it is generally solved by rules of thumb. The Quay Cranes Scheduling Problem (QCSP), on the contrary, is much more complex: it is usually formulated as a scheduling problem, where the quay cranes operating on a single vessel are parallel machines, performing a set of tasks, representing the loading/discharging operations. Of course, each task has to be processed once, and each crane can execute a task at a time. Precedence constraints between tasks are given,

to ensure, for instance, that discharge tasks precede loading ones, and the sequence of operations on the deck and in the hold are performed correctly. Some spatial constraints, aimed to avoid crossing of cranes and to ensure a safety distance between adjacent cranes, are also considered (Sammarra et al., 2007). A solution to the problem is a feasible schedule of tasks to be executed by each crane; the objective function, to be minimized, is the overall completion time for the service at the berthed vessel (makespan).

1.1.3 Stowage planning

Designing a stowage plan for a containership consists of two sequential processes (Steenken et al., 2004; Alvarez, 2006). The first step is executed by the shipping lines, whose planners have a complete view of both all containers that have to be loaded or discharged during the vessel trip, and the vessel structure. They provide stowage plans for each port of the vessel port rotation. The shipping-line objectives are to minimize the number of on-board shifts during the port rotation and to maximize the ship utilization. The result of this process is a list of documents, which are sent to the terminal planners of all ports the containership will visit. These documents act as guidelines (*Prestow Plans*) for the terminal planners, which are involved in the second step of the stowage plan design, sometimes referred as *ship loading* or *load sequencing* problem. As regards the objectives the terminal planners have to pursue, they are quite different from the shipping-line ones. In order to speed up the ship loading process, it is possible to minimize the transportation time of containers from the yard to the quay, or minimize the yard reshuffles. Reshuffles, or yard shifts, are time consuming unproductive moves, which occur whenever some containers on the top of a yard stack have to be removed (and then re-stacked) in order to pick up a container on the bottom of the same stack.

Yard management

The yard can be considered the heart of a container terminal. Most of the processes start and terminate at the yard. With the growing of the traffic, storage space is becoming a scarce resource. Yard management is influenced by many factors: technological configuration of the internal transportation equipment (Direct Transportation System vs Indirect Transportation System), type of container traffic (import, export and transshipment) and level of decisions (strategic, tactical and operational). The yard is organized in blocks, each block is made up of rows divided in bays, generally suitable to store either a 20 or a 40 feet container. Containers are stacked, and the maximum number of tiers in a stack depends on the transportation system adopted (straddle carriers or gantry cranes), and on the type of container (empty, reefer, dangerous, etc.). At the operational level, the yard management must decide where to allocate each container to be stored, but, whenever needed, also the reallocation of entire groups of containers (housekeeping) should be considered, in order to speed up the vessel handling (Moccia et al., 2009).

Yard-Quay-Yard connection

The containers have to be transported from the quay to the yard and vice versa. The vehicles used for the horizontal transportation of containers can be either active vehicles or passive vehicles. The active vehicles are able to lift and transfer containers, while passive vehicles, as trucks with trailers and AGVs (automated guided vehicles traveling along a predefined path), need pick up equipments (yard crane). The transport capability of a truck is related to the number of trailers (generally each trailer can carry 2 TEU), while an AGV is capable to move either 1 or 2 TEU.

Straddle carriers, forklifts and reachstackers are active vehicles. Straddle carriers, capable to transport either 1 or 2 TEU, are the most important ma-

chines among the horizontal transportation equipments. The importance of straddle carrier relies in their flexibility, since they can span all over the yard, and, moreover, they can stack containers on, at most, four other ones. The kind of transportation equipments to adopt is a strategical decision, whereas at the tactical level the main problem is determining the proper number of vehicles. Finally, at the operative level, assigning vehicles to cranes, routing and scheduling vehicles for loading/discharging and housekeeping operations are the major tasks.

1.2 Automotive terminal operations

Terminal operations in vehicle transshipment can differ significantly from container transshipment. Container flows are strongly fragmented and planning is done considering containers as single entity. Vehicles flow is much similar to bulk cargos; vehicles can be grouped and the group of cars is the planning entity. Unlike containers, cars are considered to be fragile objects that require careful and consequently labour-intensive handling. For example, cars cannot be stacked, which results in larger yards compared with container terminals. In our study we consider a transshipment automotive terminal in which cars must be parked in a yard made up of rows of varying lengths. Whereas containers may be relocated several times during their stay in a terminal, cars, once assigned to their parking rows, remain in the same yard position for the duration of their stay in order to reduce the risk of damage. This "no-relocation" rule, combined with the low density of the yard, increases the importance of optimal yard assignment. Cars are transferred from the quay to their parking slot by drivers, who are grouped in teams and are transported by a mini-bus that brings them back to their starting point. For drivers, the avoidance of damage is of top priority. Nevertheless, efficient operations have to be ensured. The transport distance within the

terminal, and consequently the sum of working hours for the driving personnel, is a crucial issue for planning. The best plan assigns cars to parking slots minimizing the sum of storage and retrieval distance. However, planners may accept a longer driving distance for storage operation in periods of small manpower utilization. This strategy could result in an advantage because most favorable positions that are closer to the quay, are left free and can be used in a forthcoming period of congested manpower utilization. These findings make yard management and loading/discharging operations in an automotive hub terminal different from those in a container hub terminal.

Some other decisional problems in automotive terminals are very similar to the analogous ones in container terminals, and can be approached in the same way. One of these problems is, for instance, the berth allocation problem.

1.3 The port of Gioia Tauro

The problems covered in this work arise at the Gioia Tauro terminal, one of the most important hub ports in the Mediterranean Sea. Its barycentric position in the Mediterranean Sea makes this port very attractive as a hub terminal (see Monaco et al. (2009) for a discussion about the Gioia Tauro container terminal). At the Gioia Tauro port there are both a container terminal and an automotive terminal.

The container terminal, managed by MCT (Medcenter Container Terminal Contship Italia Group), started its activity with the first ship berthed in 1995 and in just five years reached the leadership position among the ports operating in the Mediterranean Sea with a throughput of 2.652 millions of TEUs. After some years of fluctuating performance, the latest available statistics (ESPO, 2010), show a throughput of 3.468 millions of TEUs in 2008 and, due to the world economic crisis, in 2009 the TEUs handled have decreased

Figure 1.2: The port of Gioia Tauro

to 2.857 millions. Today, the worst part of economic crisis seems to be over, and forecasts are for a new period of growth. At the Gioia Tauro container terminal, for the yard-quay-yard transportation, is adopted a Direct Transfer System DTS, implemented by means of a fleet of straddle carriers, mobile cranes that move and stack containers; reachstackers are also used to handle empty containers which are arranged in taller stacks.

Recently, at the Gioia Tauro port, an automotive hub terminal, carried on by ICO BLG AUTOMOBILE LOGISTICS ITALIA S.P.A, has been established. The automotive terminal has an area of 320.000 m^2, a yard capacity of 15.000 vehicles, and equipments to technically process vehicles if necessary. A siding track, 3 head ramps and RoRo pontoons are available for the loading/discharging operations. The automotive terminal handles about 75000 cars per year. There is an area dedicated to transshipment traffic made up of 374 parking rows with lengths varying from 50 to 70 meters. The problem addressed by this study is related to the transshipment flow.

1.4 Summary of the thesis

This research is motivated by the cooperation between the LogiLab (the Logistics Laboratory affiliated to the University of Calabria) and both the terminal management companies operating at the Gioia Tauro port. The common aim of the joint research projects is to devise useful decision support system for the optimal management of the operational activities. In particular, this work studies three management problems. The first two problems, arising in transshipment container terminals, are: the Ship Stowage Planning Problem (SSPP) and the Straddle Carriers Routing Problem. The third one is the Yard Assignment Problem at an Automotive Transshipment Terminal.

Chapter 2 tackles the Ship Stowage Planning Problem (SSPP) from the point of view of the terminal manager. SSPP consists in assigning a specific container to each slot of a ship, according to a prestow plan and a loading list provided by the shipping company, satisfying constraints related to ship stability. The objective is the minimization of the total ship berthing time. Since we can assume as input data the shipping-line prestow plan, the number of cranes allocated to the containership and the sequence of ship bays each of them has to handle (as output of the QCSP), the total loading time of containers can be considered to be constant. Therefore the objective function reduces to the sum of two terms: the time to transport the containers to be loaded from the yard to the assigned slots, and the time needed to perform additional reshuffle moves. We present two mathematical model and a heuristic based on the Tabu search algorithm. The chapter concludes with results coming from a set of tests on real instances from the terminal container of Gioia Tauro.

Chapter 3 deals with Scheduling and Dispatching Models for Routing Straddle Carriers at a Transshipment Container Terminal. We analyze routing strategies for horizontal transportation equipment at a transshipment

container terminal. In particular we consider a port with an extensive yard where straddle carriers are used to move containers from the yard to the quay and vice versa. On the basis of the information available at the beginning of the planning horizon, we derive both scheduling and dispatching models for straddle carriers operating in "pooling"modality (a group of SCs shared by two or more quay cranes). The former yield optimal solutions to the problem, but apply when all the information related to the loading/discharging process can be considered to be known. On the other hand, the dispatching model does not require such a strong assumption, and thus it provides an useful online optimization tool, but with some loss in the solutions' quality. To validate the proposed models, we present also some numerical experience related to a case study, coming from the Gioia Tauro Container Terminal.

Chapter 4 addresses the Yard Assignment Problem at an Automotive Transshipment Terminal. The problem consists in assigning cars that arrive by vessels, to parking rows minimizing the total handling time. For an automotive terminal, yard management is a crucial issue. Furthermore, since that cars are transported by driver, some issues related to manpower planning need to be also considered. We present two integer linear programming models, and some extensions that take into account rolling horizon scenario, and manpower leveling objective. In order to solve the models a metaheuristic based on the ALNS framework applied to a rectangle packing problem is proposed. In fact the problem can be viewed as a variant of the two-dimensional rectangle packing problem, where an assignment of a group of cars to a set of yard rows is represented by a rectangle with the duration of stay as height and the number of occupied rows as width. Extensive experiments are provided in order to prove the effectiveness of the proposed solution methods.

Chapter 5 contains conclusions and possible future developments.

1.5 Sommario della tesi

L'attivitá di ricerca oggetto di questa tesi si colloca all'interno di una piú ampia e consolidata collaborazione tra il LogiLab (il Laboratorio di Logistica affiliato all'Universitá della Calabria) e le compagnie che gestiscono il terminal container e il terminal automobilistico nel porto di Gioia Tauro. Obiettivo condiviso dai progetti di ricerca congiunta é lo sviluppo di sistemi di supporto alle decisioni utili ai fini di una gestione ottimale delle attivitá operative. In particolare, in questo lavoro sono analizzati e discussi tre problemi gestionali. I primi due sono classici problemi in terminal di transshipment per container: il Problema della Pianificazione dello Stivaggio di Navi (PPSN) e il Problema dell' Instradamento di Straddle Carrier. Il terzo problema riguarda l'Assegnamento del Piazzale in un Terminal Automobilistico di Transshipment.

PPSN consiste nell'assegnamento di uno specifico container a ogni slot della nave, in accordo con il piano di prestivaggio e la lista di imbarco, forniti dalla compagnia di navigazione e nel rispetto di vincoli legati alla stabilitá della nave. L'obiettivo é la minimizzazione del tempo totale di servizio della nave. Essendo noti in input: il piano di pre-stivaggio e la lista di imbarco, il numero di gru operanti sulla nave e la sequenza delle operazioni che ciascuna di esse dovrá compiere (come output del problema di scheduling delle gru di banchina), il tempo totale di imbarco dei container puó considerarsi costante. Pertanto la funzione obiettivo si riduce alla somma di due termini: il tempo di trasporto dei container da imbarcare dal piazzale alla banchina e il tempo necessario per eseguire movimentazioni aggiuntive di riordino (reshuffle). Sono presentati due modelli matematici e una euristica basata sull'algoritmo Tabu search. Il capitolo si chiude con la presentazione dei risultati ottenuti eseguendo una serie di test su istanze reali relative al terminal container di Gioia Tauro.

Il Capitolo 3 si occupa del problema dell'instradamento degli straddle carrier. Sono analizzate strategie di instradamento in un porto con piazzale estensivo in cui si utilizzino gli straddle carrier per trasportare i container dalla banchina al piazzale, e viceversa. Al fine di minimizzare la distanza totale percorsa dai veicoli, ed in particolare quella percorsa "a vuoto" (senza container), si considera la modalitá operativa "pooling", che consiste nella condivisione di un gruppo di straddle carrier tra due o piú gru operanti sulla medesima nave. Sulla base delle informazioni disponibili all'inizio del periodo di pianificazione, vengono formulati un modello di scheduling ed uno di dispatching. Il primo modello consente di ottenere soluzioni ottime per il problema, ma puó essere utilizzato solo quando sono disponibili tutte le informazioni che riguardano il processo di imbarco/sbarco. D'altro canto, il modello di dispatching non necessita di una assunzione cosí restrittiva, e fornisce quindi un utile strumento di ottimizzazione online, ma con qualche perdita in termini di qualitá della soluzione. Per validare i modelli proposti, sono presentati esperimenti numerici relativi ad un caso di studio ricavato dal terminal container di Gioia Tauro.

Il Capitolo 4 affronta il Problema dell'Assegnamento del Piazzale in un Terminal Automobilistico di Transshipment. Il problema consiste nell'assegnare automobili che arrivano via nave a righe di piazzale, con l'obiettivo di minimizzare il tempo totale di movimentazione. In un terminal automobilistico, la gestione del piazzale é un aspetto cruciale. Inoltre, dato che le macchine sono movimentate da autisti, occorre tenere conto di aspetti legati alla pianificazione della forza lavoro. Sono proposti due modelli di programmazione lineare e alcune loro estensioni per l'adattamento a scenari con orizzonte rullante e che consentono di ottenere un desiderato andamento nel tempo dei livelli di utilizzo della forza lavoro. Per la soluzione di questi modelli, é presentata una metaeuristica basata sul framework ALNS applicato ad un problema di impacchettamento di rettangoli. Infatti, il problema

in esame puo essere visto come una variante del problema di impacchettamento di rettangoli in due dimensioni, in cui l'assegnamento di un gruppo di automobili ad un insieme di righe di piazzale é rappresentato da un rettangolo con altezza pari alla durata di permanenza delle automobili nel piazzale e con larghezza pari al numero di righe occupate. Viene documentata un'ampia sperimentazione per la valutazione delle prestazioni della metodologia risolutiva proposta.

Il Capitolo 5 contiene conclusioni e possibili sviluppi futuri.

Chapter 2

The Ship Stowage Planning Problem: Optimization Models and Heuristics

2.1 Problem Description

Among all processes arising at a container terminal, the loading of containers, commonly considered the most time-consuming, is the only one which involves, in some way, both the shipping lines and the terminal operators. The former provide loading (discharging) instructions, that is a list of all containers to be loaded (discharged) and their positions within the containership. The latter realize these instructions, by allocating sufficient terminal resources in order to speed up the whole process. The instructions for loading a containership are known as *stowage plans*.

Before discussing how a vessel stowage plan is drawn up, it is instructive to describe, briefly, the structure of a modern containership. We can think of a containership berthed at the quay as a box in a three dimensional space,

with the larger side parallel to the quay. It consists of several smaller boxes, which are called slots and are capable of holding one TEU. A slot, as well as the position of a container within the containership, can thus be identified by three coordinates: *bay*, *row*, *tier*.

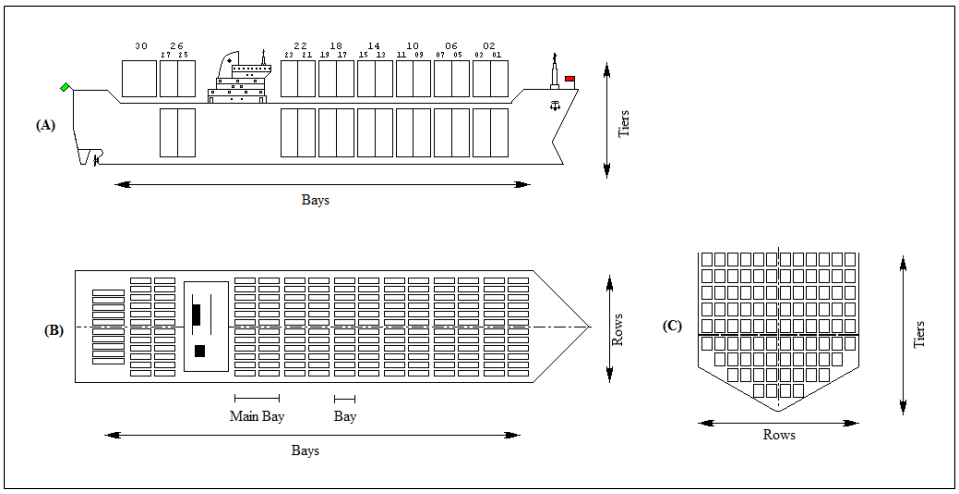

Figure 2.1: Views of a vessel.

The bay is the longitudinal coordinate of a slot, while the row and the tier represent, respectively, the transversal and vertical ones. Two adjacent slots, sharing the same row and tier coordinates, are used for storing a two TEUs container. In this case the bay coordinate is also referred as main bay. Bay and tier coordinates take positive increasing numbers from the bow to the stern and from the bottom to the top of the containership, respectively.

In order to differentiate main bays from bays the following coding is adopted: the main bay takes an even value b and the corresponding two bays are indexed by the odd numbers $b-1$ (F-bay) and $b+1$ (A-bay). In a similar way, the tier index takes even values from 02 (indicating the lowest position) to the maximum allowed for slots located below the deck (hold), while positions on the deck are labeled starting from 80. Decks and holds are separated

by steel plates, called hatch covers. Finally the row index, is a positive odd number for rows on the quay-side of the vessel, while rows on the sea-side are indexed by even numbers (the row 0 is the central one). These issues are highlighted in Figure 2.1, where the possible views of a containership are shown.

The basic Ship Stowage Planning Problem (SSPP in the following) is then the problem of assigning a slot to each container to be shipped by a vessel, while satisfying several constraints mainly related to the capacity and stability of the vessel.

Determining the arrangement of containers within a vessel is a very complicated task, and it requires full information of containers and vessel characteristics. Vessels are provided with a Cargo Securing Manual, which illustrates the maximum permissible load on each slot of the vessel, the lashing pattern, the stack weight and eight. For these reasons, the design of a stowage plan is made up, generally, by two consecutive steps (Steenken et al., 2004; Alvarez, 2006). The first step is executed by the shipping line planners, which have a complete view of both all containers to be loaded and discharged at each port the containership calls during its journey, and the cellular structure of the vessel. Containers are first classified on the basis of some attributes, like the dimension (standard, 45-footer, high-cube, oversized), the weight class (the basic ones are light, medium, and heavy), the type (reefer, open-top), the load (dangerous, perishable), and, last but not least, the port of destination (POD). Then, homogeneous containers, i.e. container of the same class, are planned to be stowed into adjacent slots.

The result of the first planning phase is either a rough stowage plan, more or less detailed, or an operative stowage plan. In the following we will call *pre-stow plan* the output of the first step. In the former case the shipping line planners indicate for each slot just the container class and the detail levels depend on the definition of the classes. For example in Figure 2.2, where a

rough plan for a vessel bay is shown, the containers are classified only by the POD, the dimension, and the type.

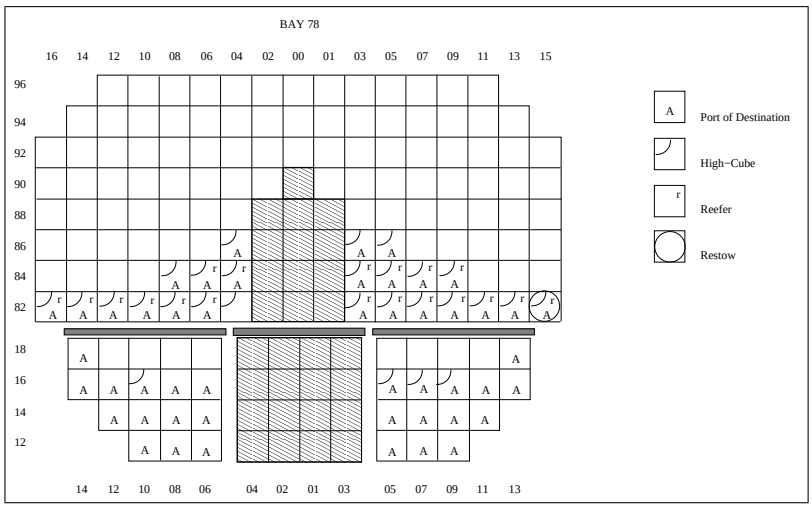

Figure 2.2: Class based stowage plan for bay 78 of a vessel.

On the contrary, an operative stowage plan, provided by the shipping line planners, indicates the exact container to be stowed in each slot, as in Figure 2.3. The details which can be retrieved by this stowage plan are (from the left to right and from the top to the bottom of each slot): the destination port (A), the loading port (B), the container code, the weight, the operating temperature for reefer containers, the type, and, finally, the slot coordinates.

In any case, shipping line planners have to take into account a lot of constraints in order to guarantee static and dynamic equilibrium of the ship. This is done by distributing weights in an uniform way, as indicated in the Cargo Security Manual. The shipping-line objectives are to minimize the number of on-board shifts, called re-stows, during the port rotation and to maximize the ship utilization.

Then the pre-stow plan is sent, via an EDI File (Electronic Data Interchange File), to the terminal planners of all ports the containership will visit.

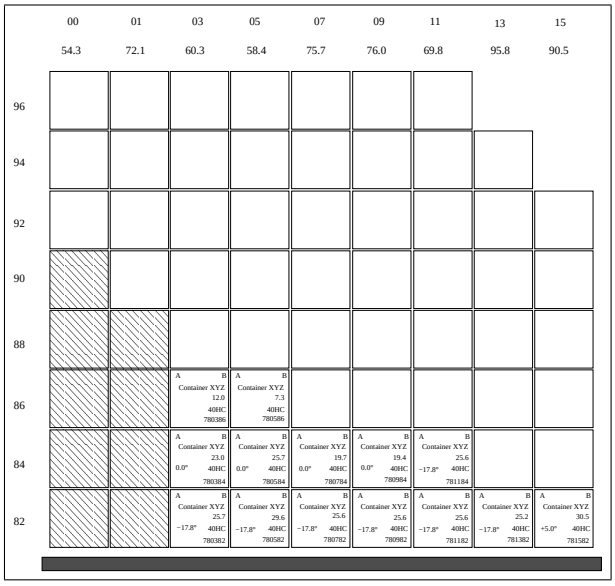

Figure 2.3: Detail of the stowage plan for quay-side rows of bay 78.

It acts as an input datum for the terminals' planners, which are involved in the second step of the stowage planning, sometimes referred as *ship loading* or *load sequencing* problem. In this phase the basic constraint planners have to comply with is that the weight of container stacks must decrease from the hold to the deck and it cannot exceed a predefined amount.

As regards the objectives the terminal planners have to pursue, they are quite different from the shipping-line ones. In order to speed up the ship loading process, it is possible to minimize the transportation time of containers from the yard to the quay, or minimize the yard reshuffles. Reshuffles, or yard shifts, are very time consuming unproductive moves, which occur whenever some containers on the top of a yard stack have to be removed (and then restacked) in order to pick up a suitable container on the bottom of the same stack.

We want to underline that when the pre-stow plan is an operative plan,

as above defined, the second phase can be skipped. Reshuffles can be minimized, and thus a high loading efficiency achieved, by resorting to premarshalling the export containers (Lee and Hsu, 2007; Lee and Chao, 2009), that is by reallocating the containers in yard locations near the export berths so that no further useless container moves will be done during the loading process. As pointed out in Moccia et al. (2009), yard areas close to the quay represent a strategic terminal resource and, at the same time, managerial practice is aimed at reducing container reallocations. This amounts to say that often terminal planners do not make use of pre-marshalling, designing stowage plans with respect to the terminal objectives, although they are provided with an operative plan.

This chapter addresses the ship stowage planning problem as it arises at a transshipment container terminal. The strong interaction between the problem under consideration and the yard layout and equipment is evident. We will consider an extensive yard, where containers are moved by a fleet of straddle carriers (*Direct Transfer System - DTS*). In such a context, yard stacks are almost three/four containers high.

Looking at the problem from the point of view of the terminal manager, the objective to minimize will be the total ship berthing time. The contribution to the ship berthing time related to the loading process is given by two terms: the total transportation time of containers from yard to quay, including the time wasted by reshuffles, and the total loading time. Since the shipping-line prestow plan is an input datum, as well as the number of cranes allocated to the containership and the sequence of ship bays each of them has to handle, the time the cranes will take to load all the containers into the ship can be considered a constant. It is worth noting that the stowage planning is an offline optimization process, in the sense that the stowage plan is generated by the ship planner before ship loading starts. Hence, the main modelling difficulty, in the case of many cranes working on the same vessel, relies on the

reshuffles estimation, since the yard stack configurations vary dynamically during the loading process.

Here we will reduce our attention to the minimization of the transportation times and of the yard reshuffles, by formulating the stowage planning problem as an integer linear programming problem. In Section 2.2 we provide an analysis of related literature. Two Linear Integer Models for the stowage problem are presented and discussed in Section 2.3. Section 2.4 describes a heuristic algorithm and is followed by numerical results in Section 2.5.

2.2 Literature Review

The SSPP has received a lot of attention by researchers. Nevertheless in some of the previous studies the distinction between the two phases is not so clear, which makes difficult the application in a real context of the models and algorithms developed.

Avriel et al. (1998) have studied the SSPP from the point of view of the shipping companies. The authors propose an Integer Linear Model, where the objective is the minimization of on-board shifts, and a heuristic algorithm to get feasible solutions to the problem.

In the paper of Wilson and Roach (1999) the whole stowage planning problem is considered. First, the authors propose a Branch-and-Bound algorithm for the pre-stow planning. Then the second phase is performed by a Tabu Search algorithm. Similar studies are reported in Wilson and Roach (2000); Wilson et al. (2001).

Dubrovsky et al. (2002) propose a genetic algorithm for the problem as described in Avriel et al. (1998), but taking into account constraints on the ship stability.

In Kang and Kim (2002) the SSPP is addressed by splitting the problem

into two subproblems, one for assigning container groups into the holds and one for determining a loading pattern of containers assigned to each hold, with the objective of minimizing the on-board shifts and the crane movements at all ports of the port rotation. The authors adopt a greedy heuristic for the first subproblem, while the second one is tackled by a tree search method.

Ambrosino and Sciomachen (2003) analyze the changes in the time needed to load a containership, when different picking strategies of containers from the yard stacks are considered.

In Ambrosino et al. (2004) the ship planning problem at the first port of the port rotation is described. The authors propose a Mixed Integer Program and a three-step heuristic, with the aim of minimizing the total loading time. Different algorithms for the same problem are given in Ambrosino et al. (2006, 2009).

The paper of Kim et al. (2004) considers the loading-sequencing problem at a port where a fleet of trucks moves containers between the yard and the quay, while stacking and retrieving of containers in the yard is performed by transfer cranes. The problem consists in determining a pick-up sequence of containers for each transfer crane, and then, for each container, a loading sequence. The authors present a Nonlinear Integer Model and a Beam Search algorithm. Similarly, Steenken et al. (2001) propose a just-in-time approach combining stowage planning and the selection of loading and transport sequences of containers by a fleet of straddle carriers.

In Alvarez (2006) the author studies the SSPP at a port where containers are moved by reachstackers and analyzes the problem from the point of view of the terminal managers. The author provides an Integer Program, whose objective function to minimize is a linear combination of the yard reshuffles and of the total distance travelled by the reachstackers. The problem is solved via a Tabu Search algorithm. Similarly in Ning and Weijian (2009)

the authors consider not only yard reshuffling but also take into account the moving frequency of yard cranes, the probability of wait by quay cranes and the feasibility of several yard cranes feeding one quay crane during the loading process. The authors provide a multi-objective linear integer model and a genetic algorithm.

Sciomachen and Tanfani (2007) combines the design of optimal stowage plans and the maximization of the terminal productivity.

Gumus et al. (2008) address the problem of assigning containers of various types to specific storage locations in a vessel at each port of the port rotation with the aim of maximizing the efficiency of the cranes and minimizing the shipping costs. The authors propose a decomposition heuristic.

By the analysis of the literature, the SSPP seems have been addressed, in the past, prevalently optimizing shipping lines' objectives, which are basically related to the ships' completion times. However, it should commonly agreed that the terminal managers have an active role in minimizing the completion time of the ships, by optimizing the transportation of the export containers from the yard to the quay. In this context the SSPP we face with differs from the previous ones, since it is mainly oriented to the terminal managers objectives, but it is not in contrast with the objectives of the shipping lines companies.

2.3 Mathematical Models

In this Section we describe two Linear Integer Models for the SSPP, which share, as input data, the following:

1. the configuration of the ship in terms of bays, rows, and tiers;

2. the class based pre-stow plan;

3. the set K of cranes allocated to serve the ship ($|K| = m$); the crane split and scheduling, that is sequence of bays processed by each crane and, for each bay, the operational modality of the crane (sea-to-shore, shore-to-sea, row-wise, stack-wise);

4. the set N of all containers to be loaded ($|N| = n_C$). For each container $i \in N$, we are given its class (c_i), weight (w_i), code, attributes, and position in the yard.

Thanks to assumptions (1) and (3), the set of slots available for stowing the export containers can be sorted by means of positional indexes. Since for each slot the pre-stow plan indicates the class of container which can be stowed in that slot, the problem reduces to find a matching between compatible containers and slots-positions, subject to side constraints. In the following we describe two alternative sorting strategies, leading to two Mathematical Models.

2.3.1 Model $\mathscr{F}_1$

A first partial sorting of the slots can be achieved by mapping the slot (b,r,t) into a sequence position indexe p of a crane k, if the slot (b,r,t) is the p-th one in the sequence of slots assigned to the crane k. Let P^k be the set of slots-positions pertaining to the crane k ($P^k = n^k \quad \forall k \in Q$). We indicate by c_p, $\forall k \in K, p \in P^k$ the class of the slot-position p, as reported in the pre-stow plan.

Each sequence P^k describes the relative positions of two slots in time. Analogously, slots can be related also in space. To this aim we define the following matrices

$$\Phi^k_{n^k \times n^k} = \left\{ \phi^k_{pq} \right\} \qquad \forall k \in Q \tag{2.1}$$

where $\phi^k_{pq} = 1$ if and only if slots related to the position indexes p and q satisfy

the conditions:

$$b_p = b_q, \;\; r_p = r_q, \;\; t_p < t_q \tag{2.2}$$

otherwise $\phi^k_{pq} = 0$. Practically $\phi^k_{pq} = 1$ indicates that slots p and q belong to the same stack and p is under q.

In a similar way, the relative position of the export containers in the yard can be described by the matrix $\Gamma_{n_c \times n_c} = \{\gamma_{ij}\}$, where $\gamma_{ij} = 1$ if containers i and j are in the same yard stack and i lies below j; otherwise $\gamma_{ij} = 0$.

By introducing the decisional variables

$$x^k_{ip} \in \{0,1\} \;\; \forall i \in N, k \in Q, p \in P^k \tag{2.3}$$

where $x^k_{ip} = 1$ if and only if the container i is assigned to the slot-position p of the sequence P^k;

$$z_{ij} \in \{0,1\} \;\; \forall i, j \in N \tag{2.4}$$

where $z_{ij} = 1$ if and only if container i precedes container j in some sequence P^k

and defining

- τ^k_{ip} as the time needed to transport the container i from its location in the yard to the bay b corresponding to the slot-position p of the crane k;

- σ as the time needed to perform a reshuffle of a container in the yard

we can formulate the SSPP as follows (Model $\mathscr{F}_1$):

$$\min \sum_{i\in N}\sum_{k\in K}\sum_{p\in P^k} \tau_{ip}^k x_{ip}^k + \sigma \sum_{i\in N}\sum_{j\in N} \gamma_{ij} z_{ij} \tag{2.5}$$

$$\sum_{k\in K}\sum_{p\in P^k} x_{ip}^k = 1 \qquad \forall i \in N \tag{2.6}$$

$$\sum_{i\in N} x_{ip}^k = 1 \qquad \forall k \in K, p \in P^k \tag{2.7}$$

$$\sum_{i\in N} w_i x_{ip}^k - \phi_{pq}^k \sum_{j\in N} w_j x_{jq}^k \geq 0 \qquad \forall k \in K, p, q \in P^k \tag{2.8}$$

$$\tag{2.9}$$

$$z_{ij} \geq x_{ip}^k + \sum_{\substack{q\in P^k \\ q>p}} x_{jq}^k - 1 \qquad \forall k \in K, p \in P^k, p \neq n^k, i, j \in N, \;\; \gamma_{ij} = 1 \tag{2.10}$$

$$x_{ip}^k = 0 \qquad \forall i \in N, k \in q, p \in P^k, c_i \neq c_p \tag{2.11}$$

$$x_{ip}^k \in \{0,1\} \qquad \forall i \in N, k \in K, p \in P^k \tag{2.12}$$

$$z_{ij} \in \{0,1\} \qquad \forall i, j \in N \tag{2.13}$$

In the model $\mathscr{F}_1$ constraints (2.6) and (2.7) are the matching constraints between containers and available slot-positions. Constraints (2.8) ensure that the containers stowed in the same ship stack satisfy the weight-decreasing rule. The yard reshuffles are computed by constraints (2.10): when i is located below j in the same yard stack, but i must be loaded before j by the same crane k, then a yard reshuffle occurs and $z_{ij} = 1$. In all the other cases $z_{ij} = 0$. As regards constraints (2.10) one could observe that they fail to take into account yard reshuffles when containers i and j are loaded by different cranes. This means that Model $\mathscr{F}_1$ returns, actually, a lower bound on the total number of yard reshuffles. On the other hand, yard stacks, usually, are homogeneous at least with respect to the port of destination of the con-

tainers. Since the crane scheduling, which we assume as an input datum, is done so that each crane handles entirely groups of homogeneous containers (Sammarra et al., 2007), it is quite unusual that two containers belonging to the same yard stack are handled by different cranes. Variable settings (2.11) avoid containers to be assigned to unsuitable slot-positions. Finally the objective function (2.5), we minimize, sums the transportation and the reshuffling times.

2.3.2 Model $\mathscr{F}_2$

If we assume that

- (1) and (3) hold, as for the Model $\mathscr{F}_1$;
- no idleness occurs in the crane work (this can be achieved considering a suitable number of straddle carriers);
- cranes have the same productivity;

then we can associate to each slot-position $p \in P^k$ a starting handling time, that is the time when the crane k starts to load a container in to the slot (b,r,t). Therefore, the sets P^k can be merged in the set $P = \{1,\dots,P\}$ ($|P| = \sum_{k=1}^{n} n^k$), and the latter set can be sorted with respect to the starting handling times of the slots, breaking ties arbitrarily.

In order to formulate an alternative Model to $\mathscr{F}_1$, we extend our notation. To this aim we define

- $\theta_p \quad \forall p \in P$ the starting handling time of the slot p;
- $\tau_{ip} \quad \forall i \in N, p \in P$, the time needed to transport the container i from its yard location to the bay b corresponding to the slot $p = (b,r,t)$;
- $L = \{1,\dots,|L|\}$ the set of ship stacks;

- $\Psi(l)\ \ \forall l \in L$ the set of slot positions belonging to ship stack l;

- $W_l\ \ \forall l \in L$ the maximum permissible weight for containers in the ship stack l;

- $N(p),\ \ \forall p \in P$ the set of containers $i \in N$ such that $c_i = c_p$;

- $P(i), \forall i \in N$ the set of slot positions $p \in P$ such that $c_i = c_p$;

- $\Delta(i)\ \ \forall i \in N$ the set of containers that are located above and in the same yard stack of container i;

- $\pi(p)$ the slot-position immediately above and in the same ship stack of the slot p, i.e. if p is the slot (b,r,t), then $\pi(p)$ corresponds to the slot $(b,r,t+1)$.

By means of the decisional variables

$$x_{ip} \in \{0,1\}\ \ \forall i \in N, p \in P \tag{2.14}$$

where $x_{ip} = 1$ if and only if the container i is assigned to the slot-position $p \in P$;

$$z_{ij} \in \{0,1\}\ \ \forall i \in N, j \in \Delta(i) \tag{2.15}$$

where $z_{ij} = 1$ if and only if container i is picked up by a straddle carrier before the container j, i.e. an additional move (reshuffle) is needed, then the Model $\mathscr{F}_2$ for the SSPP is the following

$$\min \sum_{i\in N}\sum_{p\in P(i)} x_{ip}\tau_{ip} + \sigma \sum_{i\in N}\sum_{j\in\Delta(i)} z_{ij} \tag{2.16}$$

$$\sum_{p\in P(i)} x_{ip} = 1 \qquad \forall i \in N \tag{2.17}$$

$$\sum_{i\in N(p)} x_{ip} = 1 \qquad \forall p \in P \tag{2.18}$$

$$\sum_{i\in N(p)} x_{ip} w_i - \sum_{j\in N(\pi(p))} x_{j\pi(p)} w_j \geq 0 \qquad \forall p \in P \tag{2.19}$$

$$\sum_{p\in\Psi(l)}\sum_{i\in N(p)} x_{ip} w_i \leq W_l \qquad \forall l \in L \tag{2.20}$$

$$\sum_{q\in P(j)} x_{jq}(\theta_q - \tau_{jq}) - \sum_{p\in P(i)} x_{ip}(\theta_p - \tau_{ip}) \leq z_{ij} M \qquad \forall i \in N, \forall j \in \Delta(i) \tag{2.21}$$

$$x_{ip} \in \{0,1\} \qquad \forall i \in N, \forall p \in P(i) \tag{2.22}$$

$$z_{ij} \in \{0,1\} \qquad \forall i \in N, \forall j \in \Delta(i) \tag{2.23}$$

In this model the objective function is the same as in Model $\mathscr{F}_1$, and the constraints (2.17), (2.18), (2.19) act as the corresponding constraints (2.6), (2.7), (2.8) of Model $\mathscr{F}_1$. Constraints (2.20) ensure that the maximum permissible weight on each ship stack is not exceeded. Constraints (2.21) set the correct value of the z variables whenever a yard reshuffling occurs. Actually the term $(\theta - \tau)$ represents the time when a container is picked-up from a yard stack. When i and j are located in the same yard stack (i is below j), then the left-hand-side of the constraint (2.21) is not negative if i is picked up before j. Therefore z_{ij} must be equal to one. In the other case (j is picked up before i), the left-hand-side of (2.21) is negative, but the objective function forces the corresponding z_{ij} variable to be 0. We note that, with respect to constraints (2.10), in this case we are able to evaluate correctly a yard reshuffling between a pair of containers, without requiring that those containers have to be han-

dled by the same crane. Nevertheless the number of reshuffling returned by this model is always a lower bound, due to the assumptions we made at the beginning of this Section.

2.4 Solution Algorithm

The ship stowage planning problem is, in general, $\mathscr{NP}$-hard (Avriel, 2000). All the solution methods proposed in the papers cited in Section 2.2 are based on heuristic procedures, except for the algorithm in Wilson and Roach (1999), where the SSPP is split up into two subproblems, and the first subproblem is solved by a Branch-and-Bound algorithm.

In order to get feasible solutions to the Model $\mathscr{F}_2$, we devised a solution algorithm based on the Tabu Search paradigm (Gendrau and Potvin, 2005). It is well known that Tabu Search (TS in the following) is an iterative local search heuristic provided with a memory mechanism. At each iteration h, TS explores the neighborhood of the current solution x^h in order to find an improving one. When the algorithm fails to find such an improving solution, then it has reached a local optimum. Nevertheless, TS is allowed to explore the solution space by moving towards non-improving solutions, with the hope of finding, later, a better one. In order to prevent TS reaching previously visited local optima, it keeps track, in some way, of the search history, forbidding the search to move towards the previously (*tabu*) solutions. However the tabu status of a solution may be revoked when moving towards a tabu solution yields a new incumbent (*aspiration*). This strategy implements the short term memory mechanism.

TS is also provided with a long term memory tool, which keeps track of the overall most visited solutions. The long term memory is used to lead the search towards little explored regions of the solution space (*diversification*). By means of the short and long term memory mechanisms, TS performs a

wide exploration of the solution space and, therefore, it is able to return much better solutions than the standard local search algorithm.

For our TS algorithm the neighborhood of a solution is obtained by swapping pairs of homogeneous containers in to compatible slots. The size of the neighborhood of a solution is bounded from above by $n_c \times n_c$.

2.4.1 Overall description of the Algorithm

According to Glover et al. (2007), we use attributive memory structures, meaning that we represent a solution by its attributes. If x is a solution then the set of attributes, identifying x, is the set

$$A(x) = \{(i, p) \mid \text{container } i \text{ is assigned to the slot } p\} \tag{2.24}$$

By this way the moving between solutions is performed by modifying the attribute set of a solution, i.e adding or deleting attributes. If, at a given iteration, an attribute (i, p) is removed from $A(x)$ then its re-insertion is forbidden for the next $\bar{h}$ iterations (*tabu tenure*). According to the attributive memory strategy, a move is declared tabu if at least one of the attributes describing the solution obtained by that move is tabu. However a tabu move is allowed when it yields an improving solution (*aspiration criterion*). For each attribute (i, p), in the following we denote by

- α_{ip}, the aspiration level of the attribute;
- β_{ip}, the number of iterations for which the attribute is tabu (*recency*);
- δ_{ip} the number of times the attribute has been used to identify a solution (*frequency*);

Recency, as its name suggests, keeps track of solutions attributes that have changed during the recent past. Frequency typically takes into account how

many times a certain attribute has changed, and it is used, as mentioned above, to diversify the search, that is to force the algorithm searching unexplored regions of the solution space. This is achieved penalizing a non-improving solution by the frequencies of the attributes added to describe that solution.

Moreover, we define:

- x a feasible solution, x^* the best known solution (*incumbent*);
- $c(x)$ the cost of x, that is the value of the objective function (2.16) corresponding to x;
- $F(x)$ the neighborhood of x;
- $\bar{F}(x) \subseteq F(x)$ the set of solutions obtained by x performing non tabu moves;
- $h_{\max}$ the maximum number of iteration performed by the algorithm;
- $\bar{h}$ the number of iterations for which an attribute is tabu.

The algorithm is shown in Table Algorithm 1. In lines (1) to (6) is initialized the data used by the algorithm. The set of non tabu solutions is generated throghout line (8) to (13), while the local search is performed from (14) to (21). Lines (22) to (24) and (25) to (27) manage the short term and long term memories, respectively. In lines (28) to (30) the incumbent is . The aspiration level of the attributes describing the best non tabu solution in the current neighborhood is updated by steps (31) to (33). Finally instructions (34) make the algorithm to move towards the new solutions and to iterate.

2.4.2 Computing a starting feasible solution

The Algorithm 1 can be successfully used to construct an initial feasible solution. To this aim, let x be a class based (perfect) matching between export

Algorithm 1 Tabu Search Algorithm

$x^* = x$, $h = 1$
for all (i,p)
  $\alpha_{ip} = \infty,\ \beta_{ip} = 0,\ \delta_{ip} = 0$
for all $(i,p) \in A(x)$ **do**
  $\alpha_{ip} = c(x)$
end for
repeat
  $\bar{F}(x) = \emptyset$
  for all $\bar{x} \in F(x)$ **do**
    for all $(i,p) \in A(\bar{x}) \setminus A(x)$ such that $\beta_{ip} < h$ **or** $c(\bar{x}) < \alpha_{ip}$ **do**
      $\bar{F}(x) = \bar{F}(x) \cup \{\bar{x}\}$
    end for
  end for
  for all $\bar{x} \in \bar{F}(x)$ **do**
    if $c(\bar{x}) \geq c(x)$ **then**
      $$g(\bar{x}) = c(\bar{x}) + \gamma \times c(x) \times \sum_{(i,p) \in A(\bar{x}) \setminus A(x)} \left(\frac{\delta_{ip}}{h} \right)$$
    else
      $g(\bar{x}) = c(\bar{x})$
    end if
  end for
  $\hat{x} = \operatorname{argmin} \{ g(\bar{x}) \mid \bar{x} \in \bar{F}(x) \}$
  for all $(i,p) \in A(x) \setminus A(\hat{x})$ **do**
    $\beta_{ip} = h + \bar{h}$
  end for
  for all $(i,p) \in A(\hat{x}) \setminus A(x)$ **do**
    $\delta_{ip} = \delta_{ip} + 1$
  end for
  if $c(\hat{x}) < x^*$ **then**
    $x^* = \hat{x}$
  end if
  for all $(i,p) \in A(\hat{x})$ **do**
    $\alpha_{ip} = \min\{\alpha_{ip}, c(\hat{x})\}$
  end for
  $x = \hat{x},\ c(x) = c(\hat{x}),\ h = h + 1$
until $h \leq h_{max}$

containers and available slots, that is a vector satisfying constraints (2.17) and (2.18), and not necessarily constraints (2.19) and (2.20). In particular we make sure that constraints (2.19) are satisfied for portions of containers stacks having the same Port Of Destination, type and dimension attributes. As regards relations (2.21), since they represent together with constraints (2.23) the definition of the z, they can be always satisfied whenever a x vector satisfying constraints (2.17) to (2.20) is given. In other words, constraints (2.23) can be relaxed without affecting the feasible region of the x variables.

Let us consider the following quantities:

$$\lambda_p(x) = \max\left\{ -\sum_{i \in N(p)} x_{ip} w_i + \sum_{j \in N(\pi(p))} x_{j\pi(p)} w_j \; ; \; 0 \right\} \qquad \forall p \in P \qquad (2.25)$$

$$\mu_l(x) = \max\left\{ \sum_{p \in \Psi(l)} \sum_{i \in N(p)} x_{ip} w_i - W_l \; ; \; 0 \right\} \qquad \forall l \in L \qquad (2.26)$$

and the function

$$v(\lambda(x), \mu(x)) = \sum_{p \in P} \lambda_p + \sum_{l \in L} \mu_l \qquad (2.27)$$

Basically the λ and μ measure the amount by which the constraints (2.19) and (2.20) are violated by a matching x and the function (2.27) represents the total amount of the violation. Clearly, if at least a feasible solution for the model $\mathscr{F}_2$ exists, the minimum value of (2.27) is zero, which means that all the constraints (2.19) and (2.20) are satisfied.

In order to find such a vector returning a zero value for the function v, we apply the Algorithm 1, using as starting point the above defined vector x and minimizing the function $v(\lambda(x), \mu(x))$. The steps of the algorithm are shown in the Table Algorithm 2.

Algorithm 2 Starting Solution Algorithm

1: Find the minimum cost perfect matching between export containers and available slots

$$x_R^* = \text{argmin} \left\{ \sum_{i \in N} \sum_{p \in P(i)} x_{ip} \tau_{ip} \mid \text{subject to 2.17, 2.18, 2.22} \right\}$$

2: **if** x_R^* satisfies constraints (2.19) and (2.20) **then**
3: STOP
4: **else**
5: Set $x = x_R^*$, $c(x) = v(\lambda(x), \mu(x))$
6: Call the Algorithm 1
7: **end if**

2.5 Computational Results

We have tested the Model $\mathscr{F}_2$ and the TS on a set of 20 real stowage planning instances provided by Medcenter Container Terminal, the company managing the Gioia Tauro port. The model has been implemented using ILOG Concert 2.9 and solved by ILOG CPLEX 12.1 and the TS Algorithm was coded in C++. The numerical experiments were run on a workstation equipped with one Intel Xeon $3GHz$ processor and 4 GB of RAM. We have let CPLEX running for at most one hour and adopting default settings, while the parameters of the TS Algorithm have been chosen as follows:

- maximum number of iterations $h_{max} = 500$;
- tabu tenure $\bar{h} = 10$;
- diversification penalty $\gamma = 0.005$.

The results, together with the main characteristics of the instances, are summarized in the Table 2.1. We report the lower bounds (LB) returned by CPLEX which are used to evaluate the quality (GAP) of the feasible solutions provided by the terminal planners as well as those found by solving the Model $\mathscr{F}_2$ and by running the TS Algorithm. Gaps are computed

as $Gap = (UB - LB)/UB$. Moreover since the number of reshuffles is a very important parameter for evaluating the efficiency of the yard operations, we report explicitly these values in the Table 2.1 in order to have a prompt comparison of the three solutions. Finally we report the computation times for both CPLEX and TS Algorithm.

			Terminal Planners' Solution		CPLEX				TS 500		
Instance	Cnts	Classes	GAP %	Reshuffles	LB	GAP %	Reshuffles	Time (s)	GAP %	Reshuffles	Time (s)
S438863	231	2	4,8	29	592292	0,0	0	19	0,0	0	31
S438173	337	2	5,6	46	809661	0,0	0	2259	0,0	0	51
S438031	633	2	8,6	164	1742350	—	–	3600	0,0	0	183
S438905	84	3	8,4	22	240233	0,0	0	16	0,0	0	4
S438130	103	3	8,2	12	147869	0,0	0	3	0,0	0	5
S438744	212	3	5,3	32	578702	0,0	0	52	0,0	0	24
S438422	123	4	4,9	17	332478	0,0	0	2	0,0	0	5
S438950	394	4	8,1	8	173140	0,0	0	1208	0,3	0	35
S438877	700	4	0,1	0	995208	—	–	3600	0,0	0	122
S438034	499	6	12,7	47	444200	0,0	0	1681	0,1	0	91
S437423	549	6	3,6	55	1487410	0,0	0	1182	0,0	0	57
S437412	523	7	2,1	20	917320	—	–	3600	0,0	0	126
S416573	552	7	10,6	137	1088190	0,0	35	3256	0,1	35	44
S438190	617	7	8,2	88	1105290	0,0	0	2368	0,1	0	78
S438174	209	8	4,4	12	259849	0,0	0	3	0,0	0	7
S437698	360	8	7,0	67	894175	0,1	1	3600	0,0	0	33
S438932	375	8	7,2	70	927493	0,0	0	481	0,0	0	24
S437997	289	10	0,9	6	650122	0,0	0	523	0,0	0	22
S437976	516	10	5,1	73	1369310	0,0	0	1277	0,0	0	39
S417950	296	18	4,1	16	317858	0,0	4	27	0,0	4	19

Table 2.1: Computational results ("–"means that the corresponding values are not available.)

From the analysis of Table 2.1 it is clear that both our approaches are able to improve remarkably the number of reshuffles with respect to the solution provided by the terminal planners, although with different efficiency levels (computation times). As regards the comparison between CPLEX and TS, we note that CPLEX has not been able to optimally solve four out of 20 instances within the imposed time limit. In particular in three cases (instances S438031, S438877, and SS437412) it has just returned a lower bound. On the contrary TS, in the same four cases, has returned the optimal solution within a computation time of at most three minutes. CPLEX can be considered efficient, and thus the Model $\mathscr{F}_2$ effective, in solving seven instances with computation times ranging from a minimum of two to a maximum of 52 seconds. In

the remaining cases CPLEX is poorly competitive, since it takes at least eight minutes of computation time.

As for CPLEX, also TS fails to find the optimumal solution in four cases (instances S438950, S438034, SS4173, S438190). However TS returns very small gaps and takes computation times in the range $[35, 78]$ seconds, while CPLEX takes 20 to 55 minutes for solving the same instances. Overall, the TS Algorithm seems to be very effective and efficient, since it is able to find optimal or quasi-optimal solutions with a little computational burden.

Chapter 3

Scheduling and Dispatching Models for Routing Straddle Carriers at a Transshipment Container Terminal

3.1 Problem description

On the basis of the containers' flow, container terminals can be classified in *import, export, import/export*, and *transshipment*. Another distinction can be made with respect to the vehicles adopted to transport containers from the quay to the yard and vice versa during the discharge/loading process of the berthed vessels (*quayside transport*). In particular this task can be performed either by *active* or by *passive* vehicles, where the attribute means whether or not the vehicle can load the containers by itself. Together with forklifts and reachstackers, the Straddle Carriers (SCs) belong to the family of active vehicles and are capable to transport either one or two TEUs. They

are flexible, because they potentially can span all over the yard, and, moreover, they can stack containers as well as the Yard Cranes (YCs), although at a lower stacking height (at most a straddle carrier can stack a container over two or three other ones). Terminals adopting active vehicles for the quayside transport are also known as *Direct Transfer System* (DTS) terminals. On the contrary *passive* vehicles, as trucks, trailers, and AGVs (Automated Guided Vehicles traveling along a predefined path), need cranes, both Yard Cranes and Quay Cranes, for loading/discharging containers. The transport capability of a trailer is related to the number of trailers (generally each trailer can carry two TEUs), while an AGV is capable to move either one or two TEUs. Those terminals, where the quayside transport is performed by passive vehicles, are also referred as *Indirect Transfer System* (ITS) terminals.

Adopting active or passive vehicles for horizontal transportation of containers is a strategic decision involving, therefore, several factors. Discussing about these issues goes beyond the aim of this work. Here we just want to underline that, basically, this decision depends on the yard width and on its target storage capacity. Actually at those terminals with limited stacking spaces the yard stacks must be higher than three containers to guarantee the terminal works at a profitable capacity. Therefore these terminals have necessarily to adopt the ITS technology. On the contrary the DTS terminals have wide yard and, as a consequence, the target capacity can be reached by stacking containers at lower height, which make possible the use of SCs.

We consider a transshipment DTS terminal, where one can observe a bidirectional flow of containers: from the quay to the yard and in the opposite direction, and we assume that the quay cranes (QCs), allocated to handle a given ship, perform alternately loading and discharging of container batches as in the usual practice. Under these assumptions, the management of the horizontal transportation equipment at the quayside becomes very complicated with respect to the case of import or export ports, where the containers

flow along one direction.

When SCs are used to perform the quayside transport of containers, a limited space under each crane portal works as *crane buffer*, meaning that QCs likewise SCs pick up/release containers from/into these buffers. Generally the crane buffers are capable to store (temporarily) up to five containers. The management of SCs is then aimed at maintaining as much as possible not full the buffers of cranes which are performing the discharging phase and not empty those related to cranes which are loading containers. This reflects in maximizing the cranes' productivity.

In order to state the problem addressed in this chapter, we start by analyzing the most widely used operational modalities for them: *dedicated* (or *gang*) modality and *shared* modality (or *pooling*).

In the dedicated modality a given number of SCs is allocated to a given QC, following its working phases. Therefore the SCs will move containers from the quay to the yard when the QC is in the discharging phase, and from the yard to the quay when it is in the loading phase. It is clear that the synchronism between straddle carriers and cranes makes their productivities mutually dependent. Actually, a decrease of cranes' productivity, for example during the removing of the hatch covers, causes unavoidable idle times for the straddle carriers, as well as a decreasing in the productivity of the SCs, due to the traffic congestion inside the yard, slackens the working rate of the cranes. Moreover, in the dedicated modality, every time an SC moves a container it has also to perform an unfruitful move of the same length, but in the opposite direction (*empty travel*).

On the contrary, in the pooling modality a group of SCs is shared by two or more QCs, working on the same ship or on adjacent berthed ships. It is easy to show that this modality allows to reduce the total distance traveled by the SCs, reducing the unproductive travels. Let us consider the following example, where we are given two cranes (QC1 and QC2) operating on a ship

and a pool C of SCs shared by them. We assume that QC1 is in the loading phase, while QC2 is performing the discharging of containers (see Figure 3.1).

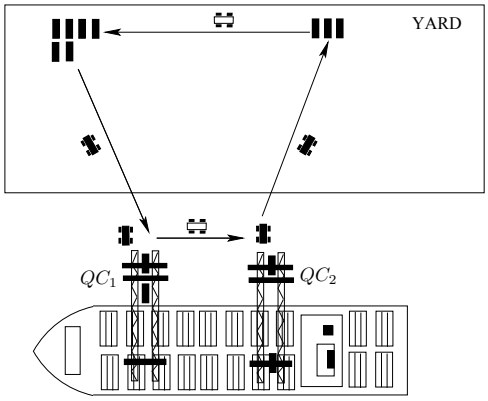

Figure 3.1: The Pooling Modality

To this aim the straddle carriers of the pool C alternately move containers from QC2 to the yard and then from the yard to QC1. Each straddle carrier $c \in C$, during its operative cycle, has necessarily to travel some empty distances. At the quay the empty distance corresponds to the distance between QC1 and QC2; in the yard it corresponds to the distance between the position of the container just moved by c and the position of the next container c has to transport to QC1. The latter distances are liable to be optimized, choosing for c the most suitable container among those which must be loaded by QC1. The same can not be done when the dedicated modality is used, since in this case the empty traveled distances are constant.

However, the reduction of the empty traveled distances is not the only benefit which derives from adopting the pooling modality. Since straddle carriers are shared by cranes, the pooling modality allows to cope with all those circumstances giving rise to sharp decays in the productivity of the

crane system. Actually, when a crane is not working, the straddle carriers of a pool can speed up operations of the other cranes, and thus a more constant value of the productivity, both of cranes and straddle carriers, can be achieved.

A further cost saving for the terminal manager is related to the reduction of the SCs fleet size allocated to the quayside transport. In fact, thanks to the more rational use of the involved resources, the same workload could be completed, in the same time, with a smaller number of SCs.

Therefore in this work we deal with the problem of the management of the SCs when they work in the pooling modality. The *Straddle Carriers Pooling Problem* (SCPP) then consists in a dynamic assignment of container moves to SCs in order to speed up the loading/discharging operations. Based on information available at the beginning of the planning horizon, the SCPP can be formulated either as a dynamic scheduling problem on parallel machines, or as a Real-Time Assignment Problem.

In the following we refer to the Gioia Tauro Container Terminal, in the Southern Italy, although the models described in this work could be applied to each transshipment port. Gioia Tauro is the main hub port in the Mediterranean Sea, with a traffic volume of about 3.5 Million TEUs in 2007, and a valuable number of feeder services connecting the terminal to some less than 60 spoke ports (Monaco et al., 2009). It is characterized by an extensive yard, where the quayside transport is performed by a fleet of staddle carriers working, at the present, in gang modality. On the contrary, the yard-to-yard transportation of containers (*housekeeping*) is usually performed by multitrailers. Due to the high operational and upkeep costs of the straddle carriers, the focus of the quayside transport optimization is the reduction of the empty travels, although the crane productivity can not be overlooked.

The outline of the chapter is the following. Section 3.2 is dedicated to a survey of the literature. Scheduling based models, where we assume to

know the sequence of containers to be loaded/discharged by each crane, are presented in Section 3. Then we address the pooling problem when some data are not available at the beginning of the planning horizon. This leads to devise a Real-Time model, which is presented in Section 4. The results of the computational experience on both classes of models are discussed in Section 5. Finally, some conclusions are drawn in Section 6.

3.2 Literature Review

The scientific production on the quayside transport optimization at container terminals is very wide: inspired by the application context in the main ports around the world, different transportation equipment have been considered, including automated guided vehicles (AGVs), yard trailers or trucks (YTs) and straddle carriers (SCs). For a complete and up-to-date literature review in this field the reader can refer to Steenken et al. (2004) and Stahlbock and Voß (2008).

Researchers have devoted a lot of attention to the case of AGVs, since their inherent lack of flexibility (no drivers experience and insight, routes on fixed paths), their low speed, and their large number, require complex management systems in order to speed up the deliveries and to synchronize the work of QCs and YCs. Evers and Koppers (1996) propose a control system using semaphores for the AGV traffic control problem. Vis et al. (2001) devise a minimum flow algorithm to determine the fleet size of AGVs in a semi-automated container terminal, assuming that containers are available for transport at known time instants. Bish (2003) discusses the problem of dispatching vehicles to containers, while determining storage location for each container and scheduling loading and unloading operations on the cranes, so as to minimize the maximum service time for a given set of ships. The dispatching problem for multi-load AGVs, i.e. automated guided vehicles

which can carry more than one container at a time, has been addressed in Grunow et al. (2004), where a flexible priority rule based approach has been suggested, and in Grunow et al. (2006), which presents a simulation study of AGV dispatching strategies.

Kim and Bae (2004) discuss the dispatching problem for single-capacity AGVs as the tool for synchronizing operations of QCs and YCs. They propose a mixed-integer linear programming model, whose objective function is a linear combination of the total delays of the QCs and the total travel time of AGVs, and a heuristic algorithm for solving the mathematical model. The main ideas presented in the paper of Kim and Bae (2004) have been afterward extended by Nguyen and Kim (2009), where the authors present a mathematical model and a heuristic algorithm for the dispatching problem of Automated Lifting Vehicles (ALV), which are special AGVs capable of lifting containers from the ground by themselves. The authors consider also constraints on the crane buffer capacity and suggest procedures to convert them into time windows constraints on the moves of containers.

A similar context but a different problem is considered in Bish et al. (2005): the fleet of vehicles moving containers between the ship area (QCs) and the yard (YCs) are, more generally, vehicles with a one container capacity. Again the port under consideration is a transshipment terminal, but more restrictive assumptions are made: the dispatching problem concerns a fleet of vehicles assigned to the QCs working on a single ship. Authors assume that the job sequence of each crane always starts with the containers to be discharged, followed by the containers to be loaded onto the ship and propose easily implementable dispatching policies in order to minimize the ship makespan.

On the contrary, in the paper of Vis et al. (2005), the depicted application context is quite different and a bit surprising: the authors consider a container terminal with lifting vehicles (with one container capacity) travelling between QCs and YCs, and limited-capacity buffer areas for the QCs. The

vehicles, they say, could be SCs or AGVs, but it is assumed that they travel on tracks in loop between yard stacks and ship, and each crane has its own track. The mix of peculiarities of the AGVs and SCs is really unusual. Due to the buffer area, each container has a time window, defined as the interval between release time and due date, in which the transportation should start. The problem is determining the minimum vehicle fleet size under the time window constraints.

We note that the dispatching problem for AGVs sensibly differs from the same problem in the case in which the containers are moved by SCs. In the first case, an AGV needs to be available by the QC throughout the loading and unloading operations, since the container to load or just discharged is picked up/put on the vehicle directly by the crane. Since no buffer space for the containers can be used and the QCs are the bottleneck resource in port container terminals, the vehicle fleet size grows, with consequent possible traffic congestion; moreover a lot of other parameters need to be neglected (as, for example, those related to the YCs) and synchronization of the AGVs routing and QCs operations becomes crucial. This to say that methods developed for the AGVs are not suitable for the SCs, although sometimes some similarity can be found, as in the case of the problem described in Kim and Bae (2004).

The routing problem for port terminals where SCs are used to move containers, has been studied by Steenken et al. (1993), Kim and Kim (1999a), Kim and Kim (1999b), Das and Spasovic (2003), Bose et al. (2000) . Steenken et al. (1993) have discussed the routing of SCs which have to perform internal and landside moves within time windows. The problem is modeled as a multiple rural postman problem and two heuristic algorithms are proposed for solving it. Kim and Kim (1999a) and Kim and Kim (1999b) focus on the routing problem, respectively, for a single SC and many SCs, during loading operation of export containers, the objective being to minimize the total travel

distance (or time) of the SCs on the yard. The latter paper generalizes the problem introduced in the former one, on the same underlying assumptions: only export containers are considered, which are handled by a combination of SCs and yard trucks or yard trailers, assigned, in a dedicated way, to the quay cranes. In order to transfer a loading container from the storage area to the containership, a straddle carrier picks up the container in the yard bay, moves it to the end of the yard bay and loads it onto a yard truck; then the YT delivers the container to the corresponding quay crane. In Kim and Kim (1999b) the problem is comprised of the container allocation problem and the carrier routing problem. In the first subproblem, containers in the yard are divided in multiple classes, each of which will be loaded by a quay crane and thus transferred by a SC: this is modeled as a transportation problem. Then, a beam search procedure is suggested in order to determine the number of containers to be picked up at each yard bay, as well as the travel route of each straddle carrier.

In Das and Spasovic (2003) port terminals where SCs are used to move containers from trucks to the storage area and vice versa are considered. The paper presents an SC scheduling procedure driven by an assignment algorithm that dynamically matches SCs and trucks as each becomes available. The objective is to minimize the empty travels of SCs, while minimizing any delays in servicing customers.

The paper by Bose et al. (2000) is, to our knowledge, the first and only work in which a routing problem very similar to the one we are considering is discussed. The authors explicitly refer to pooling policy for a fleet of SCs, assuming that no other vehicles are used for moving containers between the storage area and the quay cranes, in a transshipment container terminal. Taking into account, for each quay crane, the limited capacity buffer and the sequence of containers to be handled, their aim is to maximize the productivity of the cranes, i.e. to minimize the delays in delivery or pick up at the

crane buffer caused by the SCs. No mathematical model for the described problem is presented; the authors report, instead, some numerical experience with simple dispatching rules, showing that the pooling policy of SCs can improve productivity.

The optimal routing of yard-trailers is discussed in Nishimura et al. (2005). The authors focus on what they call "dynamic routing"of trailers, as opposite to the dedicated assignment of trailers to specific quay cranes, which is exactly the pooling strategy for their vehicles. They consider two different situations, depending on the capacity of the vehicles: the yard trailers are capable of carrying only one container (single-trailer), or more than one (multi-trailer). They propose mathematical models for both cases, the objective being to minimize the total travel distance, and develop a heuristic solution algorithm for the multi-trailer version of the problem, using a genetic algorithm. The problem is described as a special case of the vehicle routing problem with pick up and delivery; no attention is paid to the time-dimension.

In Imai et al. (2007) a different problem, again falling in the class of vehicle routing ones, is introduced: the Vehicle Routing Problem with Full Container load (VRPFC). It arises in picking up and delivering full container load from/to an intermodal terminal and consists in finding the optimal assignment of the own and chartered fleets to a set of delivery and pick up point pairs, in order to minimize the total distribution cost. The authors propose a Lagrangean relaxation-based heuristic for solving the problem.

In a more general setting, Hartmann (2004) proposes a genetic algorithm for dispatching various handling equipment and manpower in container terminals; Meersmans (2002) addresses the simultaneous scheduling problem for QCs, AGVs and YCs, proposing a branch and bound algorithm and a beam search algorithm for scheduling handling activities.

3.3 Scheduling Based Models

In a scheduling framework (Pinedo, 1995), we can associate straddle carriers to *machines* and containers to *jobs*. For each job a minimum starting time for its processing (transportation) is defined (*release date)*. Since we want to avoid cranes' idle times, we also associate to each container an upper bound on its completion time (*due date*). Moreover, the time an SC needs to perform an empty travel between two consecutive container moves can be interpreted, in a natural way, as a *setup* time. Therefore the SCPP can be formulated as a scheduling problem on identical parallel machines, with release and due dates on jobs and setup times.

In order to optimise the quayside transport, different objective functions could be defined, depending on which resource is considered as the most critical one, on the emphasis given by the terminal manager to the cost or to the efficiency of the process, and so on. The interested reader can refer to the seminal work of De Monie (1987) and to the already cited review papers Steenken et al. (2004), Stahlbock and Voß (2008) on the topics related to the port performance indices and to the impact of the quayside transport on these indices. Here we just note that the quay cranes are the most expensive equipment in a container terminal, so that, with a large agreement, a common aim is to enhance their productivity, which strongly affects the efficiency and the competitiveness of the port. In the following we will propose three different objective functions, and thus as many different scheduling models for the SCPP.

The scheduling based models rely on the following assumptions:

A1 the pool of SC serves a group of cranes which are adjacent on the quay, meaning that either they are allocated to the same ship or to adjacent berthed ships;

A2 the cranes work performing alternately sequences of loading and dis-

charging operations;

A3 for each crane, the sequence of containers to be handled is known a priori;

A4 for each outgoing container the position in the yard from which it will be picked up (*starting position*) is known;

A5 for each ingoing container the position in the yard where it will be stored (*final position*) is known;

A6 due to assumption A3, the starting position of ingoing containers and the final position of outgoing containers are also known and correspond to the positions of cranes which have to handle them;

A7 straddle carriers are identical machines, in the sense they have the same average speed and the same pick up and release times.

Now we introduce the notation that will be adopted throughout the chapter. We denote by C the pool of straddle carriers, $|C| = m$; by Q the set of adjacent cranes which are served by the pool C, $|Q| = k$; by J the set of containers to be handled by the quay cranes in Q, $|J| = n$. The set J is partitioned into the set of containers to be *discharged* J^d (ingoing containers) and the set of containers to be *loaded* J^l (outgoing containers): $J = J^d \cup J^l$.

For each container $j \in J$ we define:

- p_j, the processing time of j; it is the time an SC takes to move j from its starting position to its final position;
- r_j, the release date of j;
- d_j, the due date of j;
- $\sigma_{jl} \quad \forall l \in J \setminus \{j\}$, the time an SC takes to reach the starting position of container l from the final position of container j (*setup time*);

- $s_j^c \quad \forall c \in C$, the time the SC c takes to reach the starting position of j, if j is the first container moved by c (*starting setup time*);

- $e_j^c \quad \forall c \in C$, the time the SC c takes to reach its destination starting from the final position of j, if j is the last container moved by c (*ending setup time*);

As regards the cranes $q \in Q$ we define:

- t_q^0, the earliest availability time for q;

- $\bar{t}_q$, the average time q takes to handle a container, that is the time needed to perform a *crane cycle* (locking, lifting up, lifting down, unlocking);

- b_q, the capacity of the buffer of q;

- J_q^l, the set of containers to be loaded by q;

- J_q^d, the set of containers to be discharged by q;

Let us consider a QC $q \in Q$. Since q picks up (releases) containers from (into) the buffer, to make the crane working as long as possible uninterruptedly, each outgoing container $j \in J_q^l$ must be available in the buffer when q is ready to load it. This means that each $j \in J_q^l$ must be delivered by an SC of the pool to the buffer of q not later than the handling of all containers preceding j has been completed by q. Analogously each ingoing container must be transferred from the buffer to the yard before the buffer becomes full. By the assumption A3, if j is the i-th container handled by $q \in Q$, then the due date of j can be set as follows:

$$d_j = t_q^0 + (i-1)\bar{t}_q \qquad \forall j \in J_q^l \tag{3.1}$$

$$d_j = t_q^0 + (i+b_q-1)\bar{t}_q + p_j \qquad \forall j \in J_q^d \tag{3.2}$$

Note that (3.2) is derived by considering the worst case for the QC's buffer: $d_j - p_j$ is the latest time for the ingoing container j to be taken away from the buffer, when all its $(b_q - 1)$ successors in the crane sequence are ingoing containers too.

For what concerns the release dates, we note that each outgoing container can be moved to the crane which has to load it, at any moment in the planning horizon, since it is ready in the yard. On the contrary, each ingoing container can not be moved to the yard before it has been discharged. Therefore we have:

$$r_j = 0 \qquad \forall j \in J_q^l \tag{3.3}$$

while, if $j \in J_q^d$ is the i-th container handled by the QC q,

$$r_j = t_q^0 + i\bar{t}_q \qquad \forall j \in J_q^d \tag{3.4}$$

For convenience we also introduce two dummy containers 0 and $n+1$, which correspond to the starting and final states of the machines (SCs). For these dummy jobs we set $p_0 = p_{n+1} = 0$, $r_0 = r_{n+1} = 0$, $d_0 = d_{n+1} = K >> 0$. We note that the previously defined s_j^c, e_j^c $\forall j \in J, c \in C$ are, respectively, the setup times σ_{0j} and σ_{jn+1} $\forall j \in J$, with different possible values on different machines. This choice allows to correctly model both the case of a common depot for the SCs of the pool and the multi-depot case. The complete set of containers will be denoted by $\bar{J} = J \cup \{0, n+1\}$.

Finally we define the following decision variables:

- $t_j \geq 0 \quad \forall j \in \bar{J}$, the starting time for the move of container j;
- $w_{jl}^c \in \{0,1\} \qquad \forall j,\ l \in \bar{J},\ j \neq l,\ c \in C$, where $w_{jl}^c = 1$ if the SC c moves container l immediately after j; 0 otherwise.

Now we are ready to derive three scheduling models for the pooling prob-

lem, which share the above defined variables and the following constraints.

$$\sum_{l=1}^{n+1} w^c_{0l} = 1 \qquad \forall c \in C \tag{3.5}$$

$$\sum_{j=0}^{n} w^c_{jn+1} = 1 \qquad \forall c \in C \tag{3.6}$$

$$\sum_{c\in C} \sum_{\substack{l=1 \\ l\neq j}}^{n+1} w^c_{jl} = 1 \qquad \forall j \in J \tag{3.7}$$

$$\sum_{\substack{l=1 \\ l\neq j}}^{n+1} w^c_{jl} - \sum_{\substack{l=0 \\ l\neq j}}^{n} w^c_{lj} = 0 \qquad \forall j \in J\ ,\ \forall c \in C \tag{3.8}$$

$$\left(1 - \sum_{c\in C} w^c_{jl}\right) M + t_l - t_j \geq \sigma_{jl} + p_j \qquad \forall l, j \in J,\ \ j \neq l \tag{3.9}$$

$$(1 - w^c_{0l})\, M + t_l \geq s^c_l \qquad \forall l \in J\ ,\ \forall c \in C \tag{3.10}$$

$$\left(1 - w^c_{jn+1}\right) M + t_{n+1} - t_j \geq e^c_j + p_j \qquad \forall j \in J\ ,\ \forall c \in C \tag{3.11}$$

$$t_j \geq r_j \qquad \forall j \in \bar{J} \tag{3.12}$$

$$w^c_{jl} \in \{0,1\} \qquad \forall j,l \in \bar{J},\ c \in C \tag{3.13}$$

Constraints (3.5) to (3.8) properly define the sequences of containers to be moved by each SC: constraints (3.5) and (3.6) impose, respectively, that for each SC c, exactly one container is the first, and exactly one container is the last one in its own sequence. Note that, including in the summations the indices $l = n+1$ in (3.5) and $j = 0$ in (3.6), allows to consider feasible solutions with less than $|C|$ working straddle carriers: if, for some $c \in C$, $w^c_{0n+1} = 1$, then c is not used. Constraints (3.7) ensure that each container is moved by exactly one SC. Constraints (3.8) impose that, if the container j is moved by the SC c, then it must have exactly one predecessor and one successor in the sequence of containers assigned to c. Constraints (3.9) define the variables $t_j \ \ \forall j \in J$ as functions of the variables w^c_{jl}, taking into account the setup times σ_{jl}: since M

is a big number, it is easy to see that (3.9) imposes

$$t_l \geq t_j + p_j + \sigma_{jl}$$

if and only if, for some $c \in C$, $w^c_{jl} = 1$ (otherwise it is redundant). In a similar way, constraints (3.10) enforces

$$t_l \geq s^c_l$$

if and only if l is the first container moved by c, while (3.11) ensures

$$t_{n+1} \geq t_j + p_j + e^c_j$$

if and only if j is the last container moved by c. We note that constraints (3.10) and (3.11) can be included in constraints (3.9), with $j \in \bar{J}$, when the starting and final setup times are independent from the machine c. Finally, (3.12) are the ready time constraints.

The above constraints define the feasible set of a scheduling problem belonging to the class $P|r_j, seq.dep.|\bullet()$, where $\bullet()$ is a regular function of the jobs' completion times.

On the other hand, looking at the same problem as to a Vehicle Routing Problem (see Desrosiers et al. (1995)), it is well known that the w components of each feasible solution to constraints (3.5) - (3.13), define $m = |C|$ acyclic paths in the weighted directed graph $G = (\bar{J}, \bar{J} \times \bar{J})$, from the origin 0 to the destination $n+1$ obtained by suitably splitting the single depot or the multiple depot nodes. What remains to be done is defining the objective function to optimise. To this aim, let $\mathscr{P}$ be the polyhedron

$$\mathscr{P} = \left\{ w \in \mathbb{R}_+^{(n+2)^2 \times m} \, , \, t \in \mathbb{R}_+^{n+2} \mid (3.5) \text{ to } (3.12), \text{ hold} \right\} \tag{3.14}$$

Minimization of the total tardiness - Model $\mathscr{F}_1$

If the quay cranes productivity is the focus of the optimization problem, then it is natural to define the objective function in order to minimize the idle times in the cranes activity, caused by delays in the containers' pick up and/or delivery.

We recall that the the delay or *tardiness* T_j of job j is defined to be the lateness $L_j = t_j + p_j - d_j$ whenever $L_j > 0$, and zero otherwise:

$$T_j = \max\{0,\ t_j + p_j - d_j\} \quad \forall j \in \bar{J} \tag{3.15}$$

Therefore one can choose to minimize either the number of tardy jobs ($\sum_{j\in\bar{J}} U_j$, where $U_j = 1$ if and only if $T_j > 0$, otherwhise $U_j = 0$) or the maximum tardiness ($\max_{j\in\bar{J}}\{T_j\}$) or, finally, the total tardiness of the jobs ($\sum_{j\in\bar{J}} T_j$). There is no doubt that in this context minimizing the total tardiness is the most meaningful objective function, since this value clearly reflects the cranes' productivity. Actually minimizing the number of tardy jobs could result in few but too long cranes' idle times. On the contrary minimizing the maximum tardiness could give schedules with shorter but too many idle times.

By linearisation of (3.15), the model $\mathscr{F}_1$ that minimizes the total tardiness is the following:

$$\min \sum_{j\in\bar{J}} T_j \tag{3.16}$$

$$T_j \geq t_j + p_j - d_j \qquad \forall j \in \bar{J} \tag{3.17}$$

$$T_j \geq 0 \qquad \forall j \in \bar{J} \tag{3.18}$$

$$(w,\ t) \in \mathscr{P} \tag{3.19}$$

$$w^c_{jl} \in \{0,1\} \qquad \forall j,l \in \bar{J}, c \in C \tag{3.20}$$

This model guarantees high cranes' productivities, but it does not explic-

itly take care of the unfruitful moves of SCs. In passing we note also that its feasible set $\Omega(\mathscr{F}_1)$ is always not empty.

Models $\mathscr{F}_2$ and $\mathscr{F}_3$ - minimization of the setup times and of the makespan

Alternative models to $\mathscr{F}_1$ can be derived by considering the due dates d_j as *deadlines* $\bar{d}_j$ and so requiring that the SCs perform container moves without delays. Then constraints (3.17) become

$$t_j + p_j \leq \bar{d}_j \quad \forall j \in \bar{J} \tag{3.21}$$

Including the above constraints into the model, different objective functions can be considered, which are relevant to the transport optimization at the quayside. A first objective, directly concerning the straddle carriers, is to minimize the empty travels. In our notation, this is equivalent to minimize the sum of the setup times: the related model $\mathscr{F}_2$ is the following

$$\min \sum_{c\in C}\sum_{l\in J} s^c_l w^c_{0l} + \sum_{c\in C}\sum_{j,l\in J} \sigma_{jl} w^c_{jl} + \sum_{c\in C}\sum_{j\in J} e^c_j w^c_{jn+1} \tag{3.22}$$

$$t_j + p_j \leq \bar{d}_j \qquad \forall j \in \bar{J} \tag{3.23}$$

$$(w\ ,\ t) \in \mathscr{P} \tag{3.24}$$

$$w^c_{jl} \in \{0,1\} \qquad \forall j,l \in \bar{J}\ ,\ \forall c \in C \tag{3.25}$$

The above model corresponds to the classical *Multi Traveling Salesman Problem with Time Windows* (Desrosiers et al., 1995).

A second significant aim, more strictly related to the efficiency of the terminal, is to minimize the berthing times of the containerships served by the pool. In the machine scheduling framework, this is equivalent to consider as objective function the makespan $C_{\max}$, i.e. the time when the move of the last

container to be loaded/discharged ends. Since $C_{\max} = \max_{j \in \bar{J}} \{t_j + p_j\} = t_{n+1}$, we get the model $\mathscr{F}_3$

$$\min\ t_{n+1} \tag{3.26}$$

$$t_j + p_j \leq \bar{d}_j \qquad \forall j \in \bar{J} \tag{3.27}$$

$$(w\ ,\ t) \in \mathscr{P} \tag{3.28}$$

$$w^c_{jl} \in \{0,1\} \qquad \forall j,l \in \bar{J}\ ,\ \forall c \in C \tag{3.29}$$

We have included $\mathscr{F}_2$ and $\mathscr{F}_3$ in the same subsection since they model very closely related problems. Actually in absence of release times and due dates, minimizing the makespan with setup times on a single machine reduces to a Traveling Salesman Problem, while the same problem on parallel machines is equivalent to a Multi Traveling Salesman Problem with min-max objective (Gendreau et al., 2001).

Due to the min-max structure of the objective, the model $\mathscr{F}_3$ should force, whenever possible, to share the workload in a balanced way between the straddle carriers of the pool: this is a secondary, but not negligible, aim of the planner.

Note that, as a consequence of constraints (3.21), every feasible solution to $\mathscr{F}_2$ and $\mathscr{F}_3$ is also optimal for $\mathscr{F}_1$. By the last observation and since models $\mathscr{F}_2$ and $\mathscr{F}_3$ take into account both crane productivities and unfruitful travels of SCs, they should be preferred to $\mathscr{F}_1$, whenever feasible. On the other hand, as it is easy to guess, they are more difficult to solve than the model $\mathscr{F}_1$.

3.4 A Real-Time Model

To derive the scheduling models we have assumed to know:

- the sequence of containers handled by each QC;

- the position assigned in the yard to each ingoing container.

Sometimes these assumptions are not realistic, especially for ingoing containers. Consider a containership which has been planned to tranship containers across some ports. The shipping company, owner of the containership, draws up the stowage plan which is sent to all terminals the containership has to visit. The stowage plan reports in detail, for each terminal, the positions of containers into the containership and it is constructed on the basis of container attributes, like the destination port, the weight and so on. Containers with the same attributes (*homogeneous containers*) are stowed in adjacent slots within a ship bay. On the basis of the stowage plan, a suitable number of quay cranes are allocated to handle the containership and the ship planners can determine the sequence of operations they will perform. It is clear that stowing a container into a slot which is different from that reported in the stowage plan, has a chain repercussion to next ports in the ship route. Different causes can lead to this situation. For example, to avoid cranes' work stoppage, the deckman can accept the loading of a container j before container l, although the stowage plan indicates that j should be stowed on the top with respect to l. At the next port, where containers j and l have to be discharged, the planners expect to discharge j before l. If the scheduling models are used to plan SCs moves, wrong release and due dates will be assigned to j and l, with the consequence that the moves of SCs could not respect the predefined schedule.

As far as regards the second assumption, usually the stowage plan is used to determine the blocks in the yard where to allocate groups of homogeneous containers (see Moccia and Astorino (2007) on these topics). The precise position within a block (identified by a triple: bay, row and tier) may be assigned to ingoing containers only after they have been really discharged and checked. This is to say that the information needed to make possible the use of the scheduling models is completely available only when containers are

into the crane buffers.

To derive dispatching models which do not use the discussed hypothesis, let $H = [0,h]$ be the planning horizon and $\tau \in H$ the time when an SC, say c, has finished the move of a previously assigned container. At this time a new move, among the available ones, must be decided for c. Of course, the best choice for c, minimizing its next empty travel, can be done. However, taking a decision only for a single SC could result in a wrong strategy, as it can be seen by the example in Figure 3.2. At time τ the SC c_1 has finished to move the container j_1, wile c_2 is still moving j_2. Suppose that the next containers to be transported to the crane buffers are j_3 and j_4 ($\sigma_{j_1 j_3} \leq \sigma_{j_1 j_4}$, $\sigma_{j_1 j_4} \leq \sigma_{j_2 j_4}$, and $\sigma_{j_1 j_3} \approx \sigma_{j_2 j_3}$) and that their due dates are sufficiently large so that no delay is incurred whatever SC will move them.

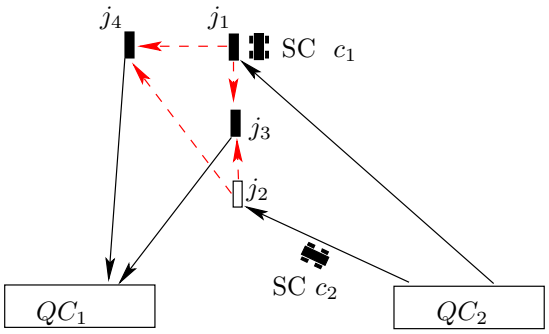

Figure 3.2: The lookahead policy.

Clearly the (locally) best choice for c_1 is to move j_3, while the solution minimizing the sum of the empty travels is obtained by assigning j_4 to c_1 and j_3 to c_2. It turns out that to avoid these myopic choices, the next move for c_1 must be decided contextually to possible next moves of all other SCs. This means that at time τ we are assigning certainly the next move to a SC and we are also trying to assign next operations to SCs which are still performing the moves previously decided for them (*Lookahead Policy*). These further de-

cisions could be updated, if needed, at the next breakpoint, i.e. at time $\tau_1 > \tau$ when another container move will be completed. One of the main benefits of this policy relies in a possible *dynamic* adjustment of all model parameters which have been assumed to be constant in the scheduling framework (crane productivity, average speed of SCs). By this way the dispatching model results to be more faithful to the real process.

To formalize these ideas, we have to extend our notation, suitably defining containers, buffers and SCs attributes varying with the time. Let $J_q^l(\tau)$, $J_q^d(\tau)$, be, respectively, the sets of outbound and inbound containers, in the sequence of the QC q, which are ready to be moved at time τ, and

$$J^l(\tau) = \cup_{q\in Q} J_q^l(\tau)$$
$$J^d(\tau) = \cup_{q\in Q} J_q^d(\tau)$$

More precisely, at the time τ, the containers in $J_q^l(\tau)$ are still in the yard, those in $J_q^d(\tau)$, already discharged, are still in the buffer of q. Therefore we can set:

$$r_j(\tau) = 0 \qquad \forall j \in J^l(\tau) \cup J^d(\tau) \tag{3.30}$$

Denoting by $b_q(\tau)$ the number of containers in the buffer of q at the time τ, we can also define the *dynamic due dates* of containers in $J^d(\tau)$ as follows:

$$d_j(\tau) = \tau + (b_q - b_q(\tau))\bar{t}_q + p_j \qquad \forall j \in J_q^d(\tau), q \in Q. \tag{3.31}$$

For the containers in $J^l(\tau)$, let j be the i-th container to be handled by q. Then we set

$$d_j(\tau) = \bar{\tau}_q + (i-k-1)\bar{t}_q \qquad \forall j \in J_q^l(\tau), q \in Q \tag{3.32}$$

where k is the position index, in the working sequence of crane q, of the last container handled by q, and $\bar{\tau}_q$ its completion time. We note that the dynamic due dates obtained by (3.31) and (3.32) are more reliable then those computed by the respective *static* definitions (3.2) and (3.1), since they include current values instead of estimated values of the process parameters. Furthermore, they reduce to the static due dates whenever nor delays neither unforeseen events occurred in the work of both cranes and straddle carriers, from the time 0 to the time τ.

Let us sort the containers in $J^l(\tau) \cup J^d(\tau)$ in increasing order of their due dates. At the time τ, the first $|C|$ containers in this ordered list define the set of "forthcoming containers": $J(\tau)$. Then we are able to define the following quantities:

- $\rho_c \geq 0$ the *residual time* the SC c needs to complete the move previously assigned to it (its *current move*);

- $\sigma_{cj} \quad \forall c \in C,\ j \in J(\tau)$, the time needed to the SC c for the empty travel between its final position (i.e. the position where c will end its current move) and the starting position of container j;

- $\gamma_{cj} \quad \forall c \in C,\ j \in J(\tau)$, the time when the move of container j would end if performed by the SC c (*completion time* of j if moved by c). It can be expressed as:

$$\gamma_{cj} = \tau + \rho_c + \sigma_{cj} + p_j \tag{3.33}$$

Introducing the decision variables $x_{cj} \in \{0,1\} \quad \forall c \in C,\ j \in J(\tau)$, where $x_{cj} = 1$ if the SC c performs the move of container j, and $x_{cj} = 0$ otherwise, the decision problem to be solved at the time τ can be modeled as the following

Assignment Problem with side constraints $AP(\tau)$:

$$\min \sum_{c \in C} \sum_{j \in J(\tau)} \sigma_{cj} x_{cj} \tag{3.34}$$

$$\sum_{j \in J(\tau)} x_{cj} = 1 \qquad \forall c \in C \tag{3.35}$$

$$\sum_{c \in C} x_{cj} = 1 \qquad \forall j \in J(\tau) \tag{3.36}$$

$$(\gamma_{cj} - d_j(\tau)) x_{cj} \leq 0 \qquad \forall c \in C, j \in J(\tau) \tag{3.37}$$

$$x_{cj} \in \{0,1\} \qquad \forall c \in C, j \in J(\tau) \tag{3.38}$$

Constraints (3.35) and (3.36) in the above model are standard assignment constraints; constraints (3.37) force the variable x_{cj} to be zero if the straddle carrier c is not able to complete the move of container j within the due date $d_j(\tau)$. The objective function is the sum of the empty travel times. Therefore, solving the problem $AP(\tau)$ is equivalent to find, at the time τ, for the forthcoming containers $J(\tau)$, the dispatching strategy which minimizes the empty travel times, while ensuring no idle times for the quay cranes. In other words, it is the "real time version"of the model $\mathscr{F}_2$.

As for the model $\mathscr{F}_2$, the due dates $d_j(\tau)$ are playing the role of deadlines on the completion time of the moves: this could lead, quite often, to unfeasible problems. For this reason, we prefer to relax the deadline constraints, putting them in the objective function and weighting their violation by suitable penalty coefficients, in a Lagrangian Relaxation fashion.

In order to simplify the notation, in what follows we will omit the dependence from the time if not strictly necessary: it should be clear that, in the context of a real time model, all parameters are dynamically computed or updated, and, in the problem $AP(\tau)$, their values are function of τ. Consider

the following constraints:

$$\sum_{c \in C} \gamma_{c\,j} x_{c\,j} \leq d_j \qquad \forall j \in J(\tau) \tag{3.39}$$

For binary $x_{c\,j}$ satisfying the assignment constraints (3.35), they are equivalent to constraints (3.37), eventhough they are weaker (in fact they are implied by (3.37) and (3.35)). Constraints (3.39) are suitable for dualization; for each choice of multipliers $\mu_j \geq 0 \quad \forall j \in J(\tau)$, we can define the Lagrangean Relaxation :

$$\begin{aligned}
&\min \sum_{c \in C} \sum_{j \in J(\tau)} \sigma_{c\,j} x_{c\,j} - \sum_{j \in J(\tau)} \mu_j (d_j - \sum_{c \in C} \gamma_{c\,j} x_{c\,j}) \\
&\sum_{j \in J(\tau)} x_{c\,j} = 1 && \forall c \in C \\
&\sum_{c \in C} x_{c\,j} = 1 && \forall j \in J(\tau) \\
&x_{c\,j} \in \{0,1\} && \forall c \in C, j \in J(\tau)
\end{aligned}$$

Thanks to constraints (3.36), on the feasible region the Lagrangean objective function can be rewritten as follows:

$$\min \sum_{c \in C} \sum_{j \in J(\tau)} \lambda_{c\,j} x_{c\,j} \tag{3.40}$$

where

$$\lambda_{c\,j} = \sigma_{c\,j} + \mu_j (\gamma_{c\,j} - d_j) \tag{3.41}$$

Note that $L_{c\,j} = \gamma_{c\,j} - d_j$ is the Lateness of the container j if it is moved by the SC c : thus, replacing, in the problem $AP(\tau)$, the original objective function (3.34) with the Lagrangean function (3.40), is equivalent to reward the assignment of container j to the SCs c which will move it on time, and to penalize

the assignment to those which will cause a delay. Therefore, at each breakpoint τ, the dispatching strategy will be obtained by solving the following relaxed assignment problem:

$$\begin{aligned} \min \quad & \sum_{c \in C} \sum_{j \in J(\tau)} \lambda_{cj} x_{cj} \\ & \sum_{j \in J(\tau)} x_{cj} = 1 && \forall c \in C \\ & \sum_{c \in C} x_{cj} = 1 && \forall j \in J(\tau) \\ & x_{cj} \in \{0,1\} && \forall c \in C, j \in J(\tau) \end{aligned} \tag{3.42}$$

where λ_{cj} are defined in (3.41).

As far as concerns the definition of the multipliers μ_j, we set:

$$\mu_j(\tau) = \frac{1}{d_j - \tau} \qquad \forall j \in J(\tau) : d_j > \tau \tag{3.43}$$

The above formula can be motivated by noting that:

- for fixed τ, the multipliers of containers $j \in J(\tau)$ are inversely proportional to their due dates d_j;
- for a fixed container j, the multiplier μ_j is increasing with respect to τ.

Of course, (3.43) can not be used if $d_j \leq \tau$, since the corresponding multiplier would result to be either undefined or negative. However, if the condition $d_j \leq \tau$ holds for some $j \in J(\tau)$, then those containers are definitely late, and what we can do at best, at the time τ, is to force the SCs of the pool to move them as soon as possible. For this reason, when the set of the "expired"containers at the time τ is not empty, i.e.:

$$E(\tau) = \{ j \in J(\tau) \mid d_j \leq \tau \} \neq \emptyset$$

we solve the assignment subproblem on $E(\tau)$, with cost coefficients γ_{cj} (so as to minimize their completion time).

3.5 Computational Experience

In this section we report on the computational experience, aimed at validating the proposed scheduling and dispatching models for the SCPP. We have based our analysis on the historical data retrieved from the database of the Gioia Tauro Container Terminal, where all information on containers' movements are stored. This allows us to compare performance of the gang modality adopted in the past for the SCs with the pooling modality proposed in our approach. Of course this comparison allows to highlight only the potential benefit of the new modality, since it is applied to the historical data of the containers' movements. We first describe the features of the test instances and how they have been constructed; then we discuss the numerical results.

3.5.1 The Test Problems

We have derived the test problems, with different and increasing dimensions. Among the ships which called the port in April 2005, we have selected, randomly, two feeder ships and two mother vessels, so that in each class there were both ships with *low* and *high* volume of handled containers. These values have been fixed up to 60 and 120 containers for feeder ships and up to 300 and 600 containers for mother vessels. The above query, on the terminal historical database, returned four basic instances (A, B, C, D); in Table 3.1 we report their characteristics and the number of SCs which have been used to perform the quayside transport. The dimension of each gang, or, equivalently, the number of SCs assigned to each QC, depends on the number of containers to be handled, but also on contractual clauses between the

terminal manager and the shipping line. Anyway, to avoid traffic congestion around the cranes, at most four SCs are assigned to each gang.

Instance	Ship	$\|J\|$	$\|Q\|$	SC
A	feeder	40	2	6
B	feeder	120	2	6
C	mother	200	4	12
D	mother	540	4	12

Table 3.1: Characteristics of the basic test problems.

In order to make the data available for the scheduling models, as described in Section 3.3, some parameters must be set beforehand. In particular we refer to the earliest availability time of cranes t_q^0, their work speed $\bar{t}_q$, and the capacity of their buffers b_q. Moreover, since we know the starting and final positions of each handled container, we must translate these distances into time values. To this aim we have set

$$p_j = \frac{|X_S(j) - X_F(j)| + |Y_S(j) - Y_F(j)|}{\bar{v}} + t_{pr} \qquad \forall j \in J \tag{3.44}$$

$$\sigma_{ij} = \frac{|X_F(i) - X_S(j)| + |Y_F(i) - Y_S(j)|}{\bar{v}} \qquad \forall i, j, i \neq j \in J \tag{3.45}$$

where $(X_S(), Y_S())$, $(X_F(), Y_F())$ are, respectively, the starting and final xy-coordinates of each container (available in the database), $\bar{v}$ is the average speed of a straddle carrier and t_{pr} its *pick up-release* time. The value of these parameters are summarized in Table 3.2.

Entries in Table 3.2 deserve some comments. First we note that there are two values for t_q^0; the null value is given to all cranes which start their work sequence with the unloading phase, while we let loading cranes to start at 360 s, in order to cope with the natural transient phenomena (some containers must be in the buffers, before cranes can start to load them). The value for $\bar{t}_q$ corresponds, in practice, to consider a working rate of 24 moves per hour,

Parameter	Value
t_q^0	0 or 360 s
$\bar{t}_q$	150 s
b_q	5
$\bar{v}$	4 m/s
t_{pr}	60 s

Table 3.2: Parameter values.

which is the average QCs productivity at Gioia Tauro, but also at many of the most efficient container terminals in the world. The average speed value for the SCs has been retrieved by the terminal database.

Finally, the adoption of the L_1 norm in the expressions (3.44) and (3.45), gives a good approximation of the real route travelled by the SCs, since they are constrained to move along straight lines, due to the yard layout.

The solution approach to the scheduling models, described in Section 3.3, is based on a *Rolling Horizon* strategy. This is equivalent to decompose each problem in several subproblems of smaller size, characterized by the same number of machines, but a smaller number of jobs. The reason for this choice is the inherent computational complexity of the problems under consideration (it is well known that they are $\mathscr{NP} - hard$) and the big size of the real instances (thousand of variables and constraints). However this approach is quite reasonable, since ship operation is a dynamic process in practice and, thus, demands online optimization. Therefore, taking decisions about container moves which are faraway to be performed, could be not only unnecessary but also poorly meaningful. Moreover, as for the real-time model, the rolling horizon strategy allows to update, whenever needed, the equipment working rates (average speed of SCs and average productivity of QCs), although in our experiments they have been kept constant.

The width h of the rolling horizon, that is the number of containers that

should be considered in each subproblem, has been assumed to be either equal or two times the number of straddle carriers in the pool. As far as concerns the pool size, we have conducted some computational experiments in order to verify the expected reduction, with respect to the gang modality, discussed in the Introduction. We have observed that a pool of SCs obtained as a simple merging of the gangs allocated to a containership is oversized: some of them would not be used. Therefore, we have defined $|C|$, starting from Table 3.1, so as to save a straddle carrier for each two cranes.

The above arguments lead to eight test instances, whose features are reported in Table 3.3, which have been used to validate the Rolling Horizon strategy on scheduling models $\mathscr{F}_1$ and $\mathscr{F}_2$. The dispatching model has been tested on the corresponding instances defined in Table 3.1, where we set the pool size ($|C|$) as in Table 3.3. The results are discussed in the following subsection. Here we want just to observe that the validation of the proposed routing strategies with respect to the gang modality, is based on a single performance index, i.e. the reduction of the empty travels:

$$\Delta ET = \frac{ET(gang) - ET(pool)}{ET(gang)} \tag{3.46}$$

where $ET(\bullet)$ is the total empty traveled distance in the corresponding operational modality for the SCs. There is a twofold reason for this choice: $ET(gang)$ is the only one indicator we are able to compute from the historical database which is not affected by aleatory phenomena (as in the case of time-based indicators) and, moreover, it is one of the most relevant indicator for the terminal manager.

3.5.2 Computational Results

We have coded the Rolling Horizon algorithm and the Dispatching heuristic in C++, adopting CPLEX 10.1 with Concert Technology to solve each

Instance	$\|J\|$	$\|Q\|$	$\|C\|$	h
A1	40	2	5	5
A2	40	2	5	10
B1	120	2	5	5
B2	120	2	5	10
C1	200	4	10	10
C2	200	4	10	20
D1	540	4	10	10
D2	540	4	10	20

Table 3.3: Features of the test problems.

scheduling subproblem as well as the relaxed assignment problems (3.42). We have run our experiments on a machine equipped with one Intel Xeon $3GHz$ processor and 4 GB of RAM, giving to CPLEX a time limit of two hours for solving each subproblem generated by the Rolling Horizon algorithm.

The results are summarized below in Tables 3.4 and 3.5. In both tables we report the reduction in empty travels with respect to the gang modality ($\Delta ET(\%)$); in Table 3.4, related to the scheduling models, the number of subproblems generated (N), the number of those optimally solved (N^*), and the average Gap ($\bar{G}$), are also shown.

The most evident result in Table 3.4 is the high computational complexity of the scheduling models, despite of the small dimension of the subproblems in which they have been decomposed: very often CPLEX fails to find the optimal solution within the wide time limit (see Instances C2 and D2). We have observed that when CPLEX returns a non optimal solution for the model $\mathscr{F}_1$, the GAP$((UB-LB)/UB)$ is always 100%; this is because, during the exploration of the Branch-and-Bound tree, it has not been able to improve the lower bound at the root node, which is, trivially, zero. As a consequence, even when the number of subproblems not solved at optimality is small, the resulting average GAP is relatively high.

		Model $\mathscr{F}_1$			Model $\mathscr{F}_2$		
Instance	N	N^*	$\bar{G}$	$\Delta ET(\%)$	N^*	$\bar{G}$	$\Delta ET(\%)$
A1	8	8	0	14	8	0	32
A2	4	4	0	16	3	15	32
B1	24	24	0	12	24	0	26
B2	12	12	0	16	10	10	24
C1	20	17	15	10	3	71	22
C2	10	7	30	10	0	75	22
D1	54	54	0	6	5	85	14
D2	27	16	41	2	0	87	14

Table 3.4: Comparison of computational results on scheduling models.

The entries in Table 3.4 show that model $\mathscr{F}_2$ outperforms $\mathscr{F}_1$, in terms of solution quality, also when for both models only feasible solutions have been found. As disclosed in Section 3.3, the superiority of $\mathscr{F}_2$ over $\mathscr{F}_1$ is due to the definition of the objective function. In the former we explicitly minimize the sum of the empty travels, while in $\mathscr{F}_1$ this is done implicitly via the minimization of the total tardiness. On the other hand, comparing the N^* columns, $\mathscr{F}_1$ seems to be "easier"to solve than $\mathscr{F}_2$. In both cases, however, a sensible reduction in the empty travels can be observed, although this reduction deteriorates for instances D1 and D2, because of the small number of optimally solved subproblems.

A part from highlighting the superiority of the pooling over the gang modality, the computational experience on the scheduling models also help us to evaluate the performance of the Real Time model, whose results are shown in Table 3.5. As expected, the reduction of the empty travels is lower than that obtained by the scheduling models. However it is fair to recall that the aim of a dispatching model is not to optimise a merit function, but to give simple and, in the same time, effective rules, in order to dynamically manage a complex system.

Instance	$\Delta ET(\%)$
A	12
B	8
C	6
D	4

Table 3.5: Dispatching Model - Computational Results.

We want to point out that the figures in the above tables are related to relatively small cases: the instance D, where ten straddle carriers are used to move 540 containers, corresponds to about a quarter of a working day. Therefore one should translate them in a more wide context, where thousand of container must be, daily, moved and a little unit saving gives rise to remarkable economic benefits to the terminal managers.

These arguments and the saving in the number of straddle carrier used to perform the quayside transport, should validate the reliability of the pooling strategy.

Chapter 4

Optimizing Yard Assignment at an Automotive Transshipment Terminal

4.1 Problem description

An automotive transshipment terminal manages large flows of incoming and outgoing cars. The cars arrive and depart by ships in large batches in a given planning period, and the yard planners have to dynamically assign incoming cars to the yard that is made up of rows of varying lengths. The main objective is the minimization of the total handling time. In order to reduce the risk of damage, once assigned to parking rows, the cars are not relocated inside the yard, i.e. their initial yard position is not modified during their duration of stay. This "no-relocation" rule, combined with the low density of the yard, increases the importance of optimal yard assignment. Cars are transported from the quay to their parking slot by drivers organized in groups of 5 or 6, dedicated to a taxi responsible for the relocation of the group. In the

following a set of cars that arrive and depart by the same vessel pair, and are of the same type (model and brand) will be called a *group*. To facilitate the yard management and the driver busing process, a group is allocated to a set of adjacent parking rows. The number of required rows depends on the car length and on the row length. Yard managers prefer not to share a row between different groups, which often results in partially empty rows.

This study was motivated by an application at the BLG Italia automotive transshipment terminal which operates in the port of Gioia Tauro located in southern Italy, on the West coast. The automotive terminal handles 75000 cars per year. The yard is spread over an area of 11 ha, and its 374 parking rows have lengths varying from 50 to 70 meters. Figure 4.1 provides an aerial view of the terminal and highlights the two main yard areas. Mother vessels unload cars while berthing at the quay on the right of Figure 4.1, and feeder vessels load cars while berthing at the quay on the left of Figure 4.1. The

Figure 4.1: Aerial view of the BLG Italia terminal, port of Gioia Tauro

remainder of this chapter is organized as follows. We present in Section 4.2 an optimization model for the yard allocation process, and we analyze the computational complexity of the problem. Two integer linear programming formulations and model extensions are discussed. The relationships between the yard assignment problem and other known problems are investigated in Section 4.3. We describe in Section 4.4 a metaheuristic algorithm for our

problem, while Section 4.5 presents computational experiments followed by some conclusions.

4.2 Optimization Model

We first introduce the notation used to derive integer linear programming formulations for our problem. We then discuss the computational complexity of the problem and some extensions.

4.2.1 Notation

The problem is defined on a time horizon discretized in $|T|$ time steps indexed by $t \in T = \{1, ..., |T|\}$. The set of groups to allocate during the time horizon is indicated by $K = \{1, ..., |K|\}$, and $R = \{1, ..., |R|\}$ is the set of parking rows. The data related to groups are:

- n^k, number of cars in group k;
- v_r^k, maximum number of cars of group k that can fit in row r;
- $a^k \in T$, arrival time of group k;
- b^k, departure time of group k;
- o^k, quay unloading position of group k;
- d^k, quay loading position of group k;
- c_a^k, largest admissible handling time when unloading group k;
- c_b^k, largest admissible handling time when loading group k.

The considered groups are those arriving inside the time horizon, and the departure time of a group may exceed the time horizon.

Rows are numbered in their filling direction, i.e. if row r is filled before row s then $r < s$. The row ordering is such that if rows r and s are adjacent and $r < s$, then $s = r + 1$. We will consider later in this section the case of an "ending-row" arising when a given row r does not have an adjacent row in the filling direction. In the following we assume that there always exists an adjacent row. For each group k we have to find a set of free adjacent rows of sufficient capacity. Since we consider parking rows of varying lengths, the number of required rows is variable as well. In other words, if r is the first row in the filling direction assigned to group k, then the last row will be $r + q_r^k - 1$, where q_r^k is the smallest positive integer value satisfying

$$\sum_{\alpha=0}^{q_r^k-1} v_{r+\alpha}^k \geq n^k.$$

The q_r^k value expresses the number of rows needed by the group k when the first row of the group is r, i.e. the group would occupy the row interval F_r^k defined as $F_r^k = \{r, r+1, ..., r+q_r^k-1\}$. Analogously, we denote by u_s^k the number of rows that group k will require if s is the *last* row of the group, i.e. u_s^k is the smallest positive integer value satisfying

$$\sum_{\alpha=0}^{u_s^k-1} v_{s-\alpha}^k \geq n^k.$$

Consequently, we have the row interval $B_s^k = \{s-u_s^k+1, s-u_s^k+2, ..., s\}$ which is equivalent to F_r^k whenever $r = s - u_s^k + 1$. Since the q_r^k and u_s^k values are related to the filling direction, we refer to them as "forward row request", and "backward row request", respectively. For notational compactness, we define the following sets:

- $T(k) = \{t \in T : a^k \leq t \leq b^k\}, \forall k \in K,$ the set $T(k)$ represents the duration of stay of group k in the planning horizon;

- $K(t) = \{k \in K : t \in T(k)\}, \forall t \in T$, groups that are in the terminal at time step t;
- $K_a(t) = \{k \in K : t = a^k\}, \forall t \in T$, groups that arrive at the terminal at time step t;
- $K_b(t) = \{k \in K : t = b^k\}, \forall t \in T$, groups that leave the terminal at time step t.

The set of feasible first row assignments for a group k is denoted by $R(k) \subseteq R$. The set $R(k)$ handles some aspects of the planning problem in a rolling horizon framework. In fact, the assignments must comply with rows occupied by groups already in the yard at the first time step. Therefore, these pre-assigned groups are taken into account in the definition of the sets $R(k)$.

The "ending-row" case is now treated by considering as infeasible an assignment of a group k to a first row r such that $q_r^k > 1$ and the set $\{r, r+1, ..., r+q_r^k-2\}$ contains an ending row. Let $\bar{R} \subset R$ be the set of ending rows. We define as $R(k)$ the subset of R such that there does not exist an intermediate ending row for any assignment of k to $r \in R(k)$, i.e. $R(k) = \{r \in R : F_r^k \setminus \{r+q_r^k-1\} \cap \bar{R} = \emptyset\}$.

With this notation we can characterize the assignment of a group by its assignment to a first row. Our decision variables are:

- $y_r^k \in \{0,1\}, k \in K, r \in R(k)$, $y_r^k = 1$ if the first row of the group k is r, i.e. the group occupies the row set $\{r, r+1, ..., r+q_r^k-1\}$.

The assignment of row r as the first row for group k, i.e. $y_r^k = 1$, forbids some assignments of groups to rows. The affected groups are those that are present in the yard during the stay of group k, i.e. groups h such that $T(k) \cap T(h) \neq \emptyset$. Any such group h cannot be assigned to any row s that interferes with group k. A forbidden row s for h is such that $F_r^k \cap F_s^h \neq \emptyset$. We define the set Φ as the

set of quadruples (k,r,h,s):

$$\Phi = \{(k,r,h,s) : k,h \in K, h > k, r \in R(k), s \in R(h), F_r^k \cap F_s^h \neq \emptyset, T(k) \cap T(h) \neq \emptyset\}.$$

A quadruple (k,r,h,s) belonging to Φ indicates that the variables y_r^k and y_s^h cannot be both equal to one.

We now introduce the data required for the objective function of our problem. Since we want to minimize the total handling time, we define as c_{vz} the handling time required to move a car between $v \in R \cup O$ and $z \in R \cup D$, where the set O represents the unloading positions $O = \bigcup_{k \in K}\{o^k\}$. Similarly we indicate by D the set of loading positions, $D = \bigcup_{k \in K}\{d^k\}$.

Our decision variables induce cost coefficients defined as follows:

- $c^k_{o^k r}$, unloading handling time for the group k when the first assigned row is r:
  $c^k_{o^k r} = \sum_{\alpha=0}^{q^k-2} c_{o^k, r+\alpha} v^k_{r+\alpha} + c_{o^k, r+q^k-1}(n^k - \sum_{\alpha=0}^{q^k-2} v^k_{r+\alpha})$.
- $c^k_{rd^k}$, loading handling time for the group k when the first row is r:
  $c^k_{rd^k} = \sum_{\alpha=0}^{q^k-2} c_{r+\alpha, d^k} v^k_{r+\alpha} + c_{r+q^k-1, d^k}(n^k - \sum_{\alpha=0}^{q^k-2} v^k_{r+\alpha})$.

These cost coefficients are used in the objective function. Observe that the loading handling time is defined for all groups, hence also for those leaving the terminal after the end of the planning horizon. Thus we account for a *future* loading handling time in the *current* planning horizon. Moreover, these cost coefficients are used to define the set of feasible assignments $R(k)$. A row r does not belong to $R(k)$ whenever $c^k_{o^k r} > c^k_a$ or $c^k_{rd^k} > c^k_b$. We observe that this models loading and unloading priorities. The c^k_a (respectively c^k_b) coefficient of a group k can be set to smaller values to ensure that the group k is assigned to rows closer to the unloading (respectively loading) quay position. This results in user-controlled parameters to specify group priorities, since closer rows mean shorter handling times.

4.2.2 Integer linear programming formulations

We can now formulate our problem, hence called the adjacent row dynamic assignment problem (ARDAP), by means of the following model $\mathscr{F}_1$:

$$\text{minimize} \sum_{k \in K} \sum_{r \in R(k)} (c^k_{o^k r} + c^k_{r d^k}) y^k_r \tag{4.1}$$

subject to

$$\sum_{r \in R(k)} y^k_r = 1 \qquad \forall k \in K, \tag{4.2}$$

$$y^k_r + y^h_s \leq 1 \qquad \forall (k, r, h, s) \in \Phi, \tag{4.3}$$

$$y^k_r \in \{0, 1\} \qquad \forall k \in K, \forall r \in R(k). \tag{4.4}$$

The objective function (4.1) minimizes the sum of the handling times. The constraints (4.2) state that each group k must be allocated to one and only one admissible first row r, since r must belong to $R(k)$. The feasibility of the assignment is guaranteed by constraints (4.3) which forbid pairs of incompatible assignments as defined by the set Φ.

The model uses $|K| \times |R|$ binary variables and the number of constraints is $O(|K| + |K|^2 \times |R|^2)$. We can obtain a more compact model $\mathscr{F}_2$ by replacing constraints (4.3) with

$$\sum_{k \in K(t)} \sum_{s \in B^k_r} y^k_s \leq 1 \qquad \forall r \in R, \forall t \in T. \tag{4.5}$$

Indeed, for a given $k \in K(t)$ the variables $y^k_{r-u^k_r+1}, \ldots, y^k_r$ are such that if one of them is equal to one, then the row r is used by group k as first row (the case $y^k_r = 1$), or as last row (the case $y^k_{r-u^k_r+1} = 1$), or as intermediate row in the other cases. Thus, constraints (4.5) state that if row r is used by a group at

time step t, but not necessarily as a first row, then its use is forbidden for all other groups staying in the yard at that time step.

The new model $\mathscr{F}_2$ still has $|K| \times |R|$ binary variables, but the number of its constraints is now $O(|K| + |T| \times |R|)$. We found that model $\mathscr{F}_1$ can only solve small instances, whereas we are able to solve model $\mathscr{F}_2$ for larger instances. We will present this comparison in Section 4.5.

4.2.3 Computational complexity

In the following we prove that ARDAP is strongly $\mathscr{NP}$-hard.

Theorem 4.2.1. ARDAP is strongly $\mathscr{NP}$-hard. *Proof* — We prove this result by showing that the generalized assignment problem (GAP), which is is strongly $\mathscr{NP}$-hard, is a particular case of the ARDAP. In the GAP the aim is to determine a minimum cost assignment of a set of weighted items to a set of knapsacks (Martello and Toth, 1992). Let $N = \{1, ..., n\}$ be the set of items, and $M = \{1, ..., m\}$ the set of knapsacks. We indicate by c_{ij} the assignment cost of item i to knapsack j, by w_{ij} the weight of item i when assigned to knapsack j, and by W_j the capacity of knapsack j. An equivalent ARDAP instance can be defined as follows:

- an item i corresponds to a group k and vice versa, i.e $K = N$, and in the following we equivalently refer to items or groups;
- the ARDAP time horizon consists of only one time step, $|T| = 1$, and all groups defined above arrive and leave the terminal at this time step, i.e. $K(1) = K$;
- the number of rows is equal to the sum of the knapsacks capacities, $|R| = \sum_{j \in M} W_j$;

- we partition the set R into m subsets $S_j, j \in M$: $S_j = \{r_j, ..., s_j\}$, where $r_j = \sum_{l=1}^{j-1} W_l + 1$ and $s_j = r_j + W_j - 1$, i.e. $|S_j| = W_j$; in the following we denote these sets as artificial knapsacks;

- the group forward row request q_r^k is constant for the row belonging to a given subset S_j, and it is equal to the corresponding weight of the item: $q_r^k = w_{kj}, \forall r \in S_j, j \in M$; similarly, the group backward row request u_s^k is equal to the weight of the item in each subset S_j;

- the group to row assignment cost $c_{o^k r}^k + c_{r d^k}^k$ is constant for the row belonging to a given subset S_j, and it is equal to the corresponding cost of the item, i.e $c_{kj}, \forall r \in S_j, j \in M$;

- the set $R(k)$ is constructed so as to avoid assignments of group k to rows that would exceed the capacity of the artificial knapsack: row $r \notin R(k)$ if there are two artificial knapsacks j and l such that $F_r^k \cap S_j \neq \emptyset$ and $F_r^k \cap S_l \neq \emptyset$.

The procedure outlined above constructs an ARDAP instance equivalent to a GAP. An optimal solution for this ARDAP instance could be polynomially transformed into an optimal solution for the GAP. Therefore, if there exists a pseudo-polynomial algorithm $\mathscr{A}$ for the ARDAP, then $\mathscr{A}$ would solve the GAP as well. Since the GAP is known to be strongly $\mathscr{NP}$-hard, the result follows. □

4.2.4 Extensions of the model

In real-life, yard assignment decisions are made on a daily basis by the yard planner who knows with a high degree of reliability the list of calling vessels and the groups of cars that will arrive and depart within a planning horizon of one week. Data regarding the following weeks are considered to be insufficiently reliable for yard planning. Every day the planner assigns

the groups expected to arrive within the planning horizon, but assignment for an incoming group can change between two subsequent plans. The final assignment is determined upon the arrival of the group. In this sense, the yard management operates according to a *rolling horizon* framework. This dynamic setting, and the favorite policy of assigning a group to adjacent rows can cause infeasibilities because of yard fragmentation. This occurs whenever the number of free rows is at least equal to the number of requested rows, but is insufficient to park the cars according to the favorite "adjacent rows" policy. Whenever this situation occurs the yard planner must determine a configuration amenable to the favorite policy. He can decide to break an incoming group into smaller ones, or to relocate some groups of cars. This last option is the least preferred and it is avoided as much as possible.

We have devised a modification of the objective function in order to consider this issue. The idea consists in favoring yard plans that have a large set of free adjacent rows at the end of the planning horizon. Thus, the fragmentation risk is minimized when the new plan is drawn on the following day. Let L_t be the largest total length of free adjacent rows at time step t in a given yard plan. Then, the modified objective function is

$$\text{minimize} \sum_{k \in K} \sum_{r \in R(k)} (c^k_{o^k r} + c^k_{r d^k}) y^k_r - \gamma_1 L_{|T|}, \tag{4.6}$$

where $\gamma_1 > 0$. In Section 4.5 we will highlight the tradeoff between minimizing handling times and minimizing fragmentation.

The yard planner faces another set of issues related to manpower planning. Whenever the handling activities are low, he can choose assignments of incoming groups to less favorable positions, i.e. more distant ones. This strategy could result in an advantage because positions that are closer to the quay, and thus more favorable, are left free for busier periods. It is then of paramount importance to profile the level of handling activity in the termi-

nal as a result of the yard allocation process. The concept of resource profile of planning activities upon shared terminal resources was introduced by Won and Kim (2009), and was also used by Giallombardo et al. (2010) for quay cranes in berth allocation plans. At time t the total handling induced by yard allocation is equal to $\sum_{k\in K_a(t)}\sum_{r\in R(k)} c^k_{o^k r} y^k_r + \sum_{k\in K_b(t)}\sum_{r\in R(k)} c^k_{r d^k} y^k_r$. We have added the following additional term to the objective function in order to obtain "close to desired" handling profiles:

$$\gamma_2 \sum_{t\in T} \left[\sum_{k\in K_a(t)} \sum_{r\in R(k)} c^k_{o^k r} y^k_r + \sum_{k\in K_b(t)} \sum_{r\in R(k)} c^k_{r d^k} y^k_r - H_t \right]^+ . \tag{4.7}$$

Here we indicate by H_t the largest desired handling value at time t. Thus (4.7) is the sum of the positive deviation from the desired handling profile. The positive weighting factor γ_2 is used to control the relative importance of this term of the objective.

The model can be solved iteratively by using arbitrarily large H_t values at the first iteration, which in fact disables the term (4.7). Then, if the planner prefers to smooth the resulting handling peaks of this first solution, the model is solved by imposing the desired H_t values. The process is iterated until a feasible and satisfactory solution has been found.

The model extensions just introduced suggest an iterative use of the model under different assumptions and input data such as group priorities, forecasts, desired fragmentation, desired handling profile, etc. The modified objective function (4.6), plus the term (4.7), could be incorporated within the integer linear programming formulation by adding proper variables and constraints. However, solving this problem exactly is impractical even for the basic model because of its computational complexity. These are additional motivations for the metaheuristic algorithm presented in Section 4.4.

4.3 Relations with other optimization problems

The scientific literature related to the management of container terminal yards is rich and expanding. For reviews see Vis and De Koster (2003), Steenken et al. (2004), and Stahlbock and Voß (2008). The wide range of contributions is justified by the many possible technological configurations, the different decision levels (strategic, tactical, operational, real-time), and the types of container flows (import, export, transshipment) that exist. Automotive terminals can be seen as another type of technological setting. The distinguishing features of yard management in automotive terminals with respect to container terminals have been discussed in Mattfeld and Kopfer (2003), and Mattfeld (2006). These features derive from the no-relocation policy, and the low density yard in this type of terminals. Therefore, container terminal based approaches cannot be straightforwardly applied to this context.

The work of Mattfeld and Orth (2006) is the closest to our study. These authors present a task scheduling and allocation problem in a large automotive terminal under different assumptions than ours. They differentiate between inbound storage tasks and outbound retrieval tasks. Both types of tasks must be executed within given time windows. This flexibility is exploited to handle the objective of leveling manpower utilization. This feature does not arise in our application because the terminal in our case deals mainly with vessel to vessel flows, and the arrival and departure times of groups are input data. Furthermore, in Mattfeld and Orth (2006) the space allocation is modeled at a more aggregate level than in our model (with knapsack type capacity constraints). This is justified by the larger and more complex layout of the terminal of Bremerhaven which serves as a basis for their study. The smaller size of the Gioia Tauro terminal enables us to optimize the assignment of cars to parking rows under the favorite policy of adjacent rows for groups.

The ARDAP can be also viewed as a variant of the two-dimensional rect-

angle packing problem. An assignment of a group to a set of yard rows is represented by a rectangle with the duration of stay as height and the number of occupied rows as width. The two dimensions of a bin are rows and time, and there are as many bins as the number of ending rows. Thus, solving the ARDAP is equivalent to packing a set of rectangles in several bins so that placement cost is minimized. Figure 4.2 illustrates the solution of an ARDAP instance with 20 groups to be allocated on a yard with 374 rows (horizontal axis) in a time horizon of 31 time steps (vertical axis), and with only one ending row (resulting in one bin). In this instance group 8 arrives at time 0, departs at time 9, and occupies rows 0 to 57. However, the ARDAP ex-

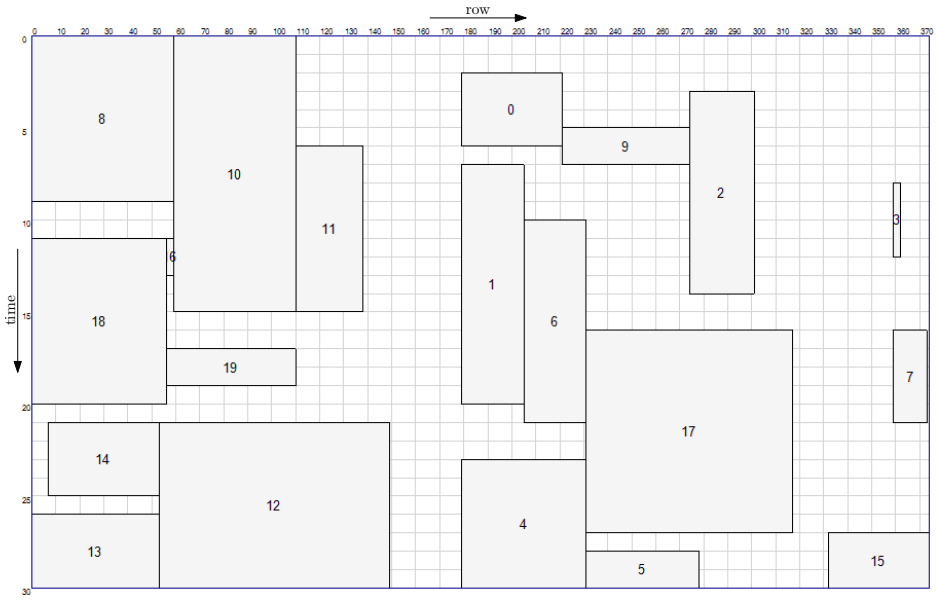

Figure 4.2: Optimal solution of an ARDAP instance in the row-time plane

hibits many differences with respect to classical rectangle packing problems. The most popular of these problems, see e.g. Lodi et al. (2002), are the bin packing problem (BPP), and the strip packing problem (SPP), where the objective function to be minimized is the number of bins (for the BPP), or the

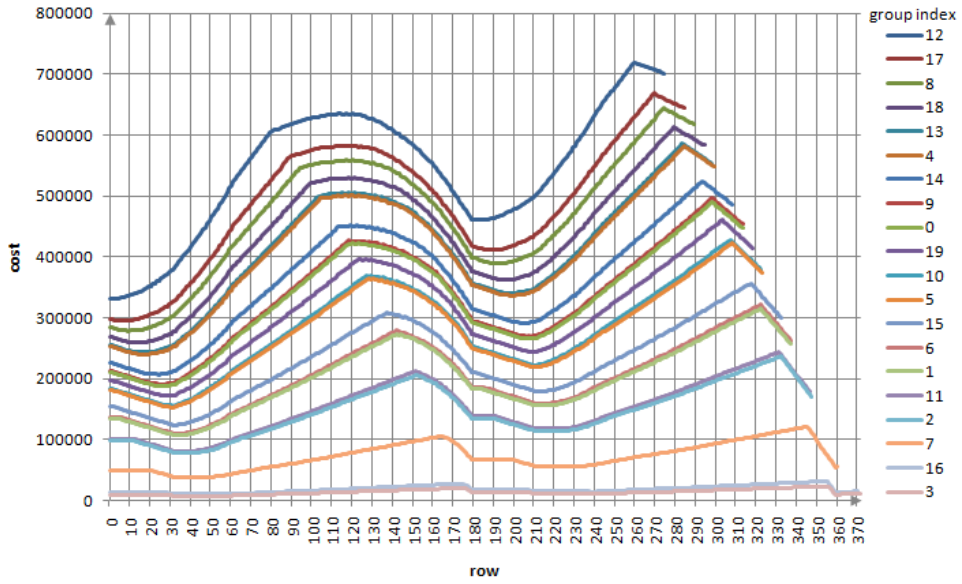

Figure 4.3: Cost function for each group of the instance depicted in Figure 4.2

height of the strip containing all rectangles (for the SPP). The ARDAP objective function is different because for each rectangle placement there is a position specific cost whose sum must be minimized. We also point out that the placement cost along the row axis is non-convex in our application and is specific for each group (i.e. for each rectangle). In the following we provide an illustrative example which needs some additional information about the application context.

The instance depicted in Figure 4.2 is derived from historical data of the operational database of the Gioia Tauro terminal. The assignment cost function is obtained considering the road network of the yard, and Figure 3 depicts these cost functions for all groups of this instance. Figure 4.4 illustrates the yard layout with the parking rows. Filled parking rows are represented as gray shaded. A similar shade intensity between adjacent rows indicates that

these rows are assigned to a group of cars. Cars are usually unloaded from vessels berthing at the North quay (where the vessel on the right of Figure 4.4 is berthed) and loaded in vessels at the East quay (on the left of Figure 4.4). This results in unloading and loading costs for each assignment of groups to sets of adjacent rows. In Figure 4.4 we denote as area A_i and area B_i the set of rows relative to the same quay segment, and A_i is closer to the quay than B_i is. The set of rows are numbered in increasing order from left to right of Figure 4.4, with the exception of area B_0 which is a special area. This same ordering is applied to the numbering of individual rows and the rows of type A have a lower index than those of type B. In order to relate the cost functions of Figure 4.3 with the layout of Figure 4.4 we mention that the row numbered as 178 is the last row of area A_6 on the right of Figure 4.4. In fact, this row could be considered as an ending row not adjacent to the row 179 which is in the area B_1 on the left of Figure 4.4. Considering ending rows would cause discontinuities in the cost functions. We preferred to omit ending rows in the example for simplicity. This is not arbitrary because the yard planner often does not enforce the ending row concept and some groups are allocated following this numbering order.

We can now discuss the impact of these rules on the solution method. The usual search strategy in rectangle packing heuristics explores the space of contiguous rectangles because this allows area minimization, but this could result in non-optimal solutions for the ARDAP. The solution of Figure 4.2 is optimal even though some rectangles are not contiguous. The optimal solution tends to assign some rectangles close to positions that would minimize their cost function, whereas other rectangles are "sacrificed" with different placements that tend to be as close as possible to other locally minimum positions. This issue is not particular to this problem. For example, it also occurs in berth allocation problems where yard costs are considered in the objective function. For the berth allocation problem see, e.g. Park and Kim (2003),

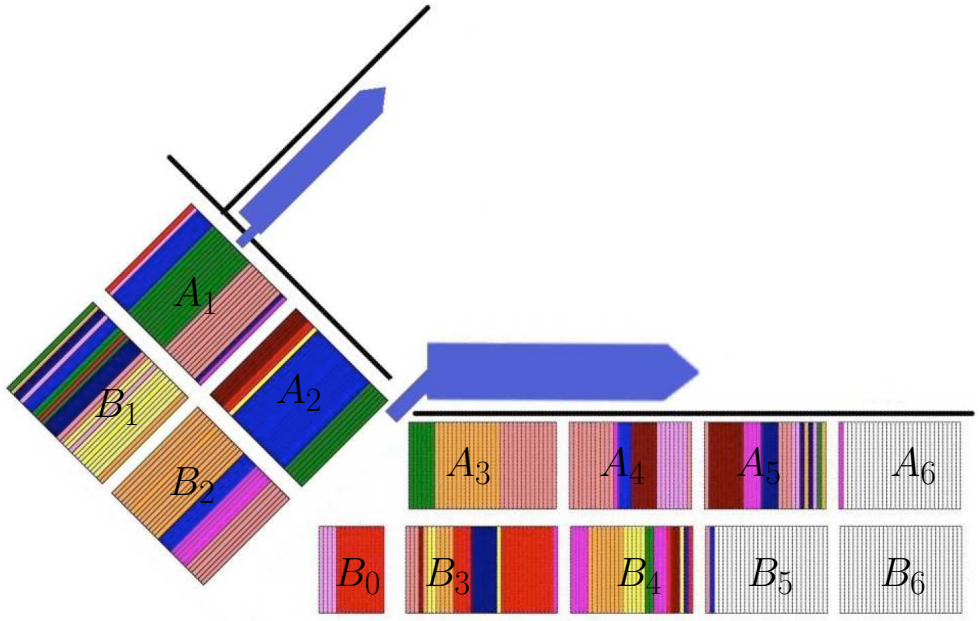

Figure 4.4: Example of yard allocation at the BLG Italia terminal

Cordeau et al. (2005), Meisel and Bierwirth (2009), and for a recent survey Bierwirth and Meisel (2010). This is the reason that motivated us to develop a metaheuristic algorithm that looks for the explicit minimization of the cost function instead of the used area.

The ARDAP is more similar to the rectangle packing problem with general spatial costs introduced by Imahori et al. (2003), and Imahori et al. (2005). However, the ARDAP exhibits distinctive features: the width of a rectangle is dependent on the assigned position (the q_r^k values in our notation), and the placement cost also depends on the assigned position. The general packing problem of Imahori et al. (2003), and Imahori et al. (2005) could handle the ARDAP artificially by expanding the number of modes of a rectangle. In the framework of these authors, a rectangle can have different modes, i.e. dimensions, and mode specific costs. The ARDAP could be modeled by in-

troducing a mode (r,k) for each $r \in R(k)$, and assigning a sufficiently high value to the cost of assigning a mode (r,k) to a row l with $l \neq r$. However, it is clear that these ARDAP specific features (position dependent rectangle dimensions and position specific costs) render the adaption of an existing algorithm for the rectangle packing impractical. Another ARDAP feature which makes the problem more constrained is that a rectangle can only move horizontally. The vertical placement cannot change since the arrival and departure times are fixed. In view of this, we are interested in exploiting the features of our problem. In particular, we can use the information of given duration of stay of groups. We have thus defined search schemes using this information.

The literature concerning storage assignment for automatic warehouse systems is also relevant to our problem. For example, an implicit assumption in yard management is the use of a *shared storage policy* (as opposed to a dedicated storage policy) whose merits have been discussed in the seminal paper of Goetschalkx and Ratliff (1990).

In the previous paragraphs we have examined the relationships between the ARDAP and the GAP. Moccia et al. (2009) have introduced an extension of the GAP, called the dynamic GAP (DGAP). Like the ARDAP, the DGAP considers a discretized time horizon and associates a starting time and a finishing time with each task. The ARDAP is more constrained than the DGAP which allows relocation during the duration of stay of the tasks (groups), whereas the ARDAP does not. However, the main difference between the two problems consists in the degree of detail on the spatial allocation: whereas the DGAP has knapsack-like capacity constraints, the ARDAP defines adjacent row assignment for each group.

4.4 A metaheuristic algorithm for the ARDAP

In the following we introduce a metaheuristic algorithm based on the adaptive large neighborhood search (ALNS) framework. In the ALNS a number of simple heuristics compete to modify the current solution (Pisinger and Ropke, 2007). A master level layer adaptively selects heuristics to intensify and diversify the search. At each iteration a heuristic is chosen to destroy the current solution, and another is chosen to repair it. The new solution is accepted if it satisfies the criteria of the local search algorithm chosen at the master level. The ALNS framework can be applied to a wide class of optimization problems. The adaptive layer chooses the heuristics according to the *scores* obtained at previous iterations. We denote the past score of the heuristic H_i by π_i, and the probability of selecting the heuristic H_j is

$$\frac{\pi_j}{\sum_{i=1}^{z} \pi_i}, \tag{4.8}$$

where z is the number of heuristics. Details about the scores will be given in Section 4.4.5. In order to solve a given optimization problem using the ALNS framework one needs to design some destroy and repair heuristics, as well as a local search framework at the master level. The destroy heuristics remove groups from the yard, and the repair heuristics try to insert them in new positions. Sections 4.4.1 and 4.4.2 describe these two sets of heuristics. In our ALNS implementation we use a two-phase mechanism. A first phase looks for feasibility only, whereas the second phase tries to obtain good quality solutions. The reason for this is that some of our destroy and repair heuristics are either useful for feasibility or for optimality. We have designed a first phase in which the criterion for choosing the destroy and repair combination of heuristics is fixed. The algorithm starts by assigning each group to a dummy position with a high assignment cost. This is the starting in-

feasible solution. The first phase ends when a feasible solution is obtained, or the maximum number of iterations has been reached, in which case no feasible solution has been identified and an error message is returned. After this first phase, the selection of the destroy and repair heuristics is guided by the adaptive heuristic selection mechanism described in Section 4.4.5. Each improving solution is refined by applying a post-optimization procedure to be presented in Section 4.4.3. The master level local search is described in Section 4.4.4. Algorithm 3 outlines the ALNS algorithm.

Algorithm 3 ALNS algorithm

```
1: Construct a starting solution x
2: x* = x
3: repeat
4:    Choose a destroy heuristic H⁻ and a repair heuristic H⁺ according to
      a given rule (first phase), or according to the probability based on the
      previously obtained score π (second phase)
5:    Generate a new solution x' from x using the heuristics H⁻ and H⁺
6:    if x' can be accepted then
7:       x = x'
8:       Update score
9:       if f(x) < f(x*) then
10:         Apply a post-optimization procedure
11:         x* = x
12:      end if
13:   end if
14: until stopping criterion is met
15: if x* is feasible then
16:    return x*
17: else
18:    return an error message
19: end if
```

4.4.1 Destroy heuristics

A destroy heuristic takes as input a given solution x and determines the ω groups to remove from the yard, where ω is an input parameter. When selecting the group to be removed it is important to consider the *time relatedness* of groups. Two groups are considered time related if they are both in the yard at the same time step, i.e. if $T(k) \cap T(h) \neq \emptyset$. Figure 4.5 depicts an example of time related groups. The main idea is that by re-assigning a set of

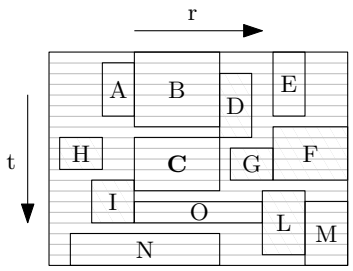

Figure 4.5: Example of time related groups in a space-time representation. D, F, G, H, I and L are the groups related to group C

groups such that each group is time related to at least another group in the set increases the likelihood of improving the objective function value.

In order to select the groups to be removed, the list of groups is sorted according to some criterion. The groups are then chosen by scanning the list and selecting a group with probability p, where p is an input parameter in the interval $(0, 1]$. This is accomplished by randomly drawing a random number ρ in $(0, 1]$ and comparing it with p. Whenever $\rho \leq p$ the current group is chosen from the list; otherwise the next group in the list is considered. The parameter p plays a randomization role. Whenever $p = 1$ the groups are selected exactly as in the list order. A smaller value of p increases the probability of choosing groups at the end of the list. The removed groups are added to a *destroy set* to be used by the repair heuristic that will try to reallocate

them. Removing a group from the yard corresponds to releasing the yard rows previously assigned to that group. Algorithm 4 outlines a generic destroy heuristic. We use four destroy heuristics that differ mainly in the sorting

Algorithm 4 Generic scheme of a destroy heuristic

1: Sort groups according to some criterion
2: **repeat**
3: Select a group k scanning the sorted list according to the randomization parameter p
4: Add k to Destroy set
5: Remove k from the list, from the current solution x and free yard rows previously assigned to k
6: Perform some heuristic specific operations
7: **until** ω groups are removed

criterion of the list of groups. The first is used only at the beginning of the process, to find a feasible initial solution.

Largest-out heuristic

The largest-out heuristic lists the groups in non-decreasing order of number of cars, which is a proxy for the number of rows that will be used by a group. The list is then scanned and a first group is selected with probability p. We denote the first chosen group as the *seed* group. As soon as a seed group is selected, groups that follow in the list are chosen only if they are time related to the seed group. If the end of the list is reached before selecting ω groups, then the procedure iterates by choosing another seed group among the remaining ones in the list.

Time-step destroy heuristic

The purpose of this heuristic is to remove a set of groups that are contemporary in the yard. The algorithm randomly chooses a time $\bar{t}$, and then

creates a list of groups such that each group in the list is in the yard at time $\bar{t}$. Whenever the number of groups in this list is less than ω all of them are added to the destroy set and the procedure iterates by choosing a different time step. Otherwise, exactly ω groups are selected from the list by using the randomization parameter p.

Worst-out heuristic

The worst-out heuristic is based on the *quality* of group-row assignments. For each group k we define the quality index $\triangle_{f_k}$ as the difference between the current assignment cost for the group k and the ideal cost f_k^*. The value f_k^* is the best assignment cost that the group k could have if the yard was entirely available, i.e. $f_k^* = \min_{r \in R(k)}\{c^k_{o^k r} + c^k_{r d^k}\}$. The heuristic first chooses a group k by scanning the list of groups ordered by non-increasing values of $\triangle_{f_k}$. It then chooses groups that are time related to k by scanning the list starting from a random position. The process is iterated until ω groups are selected.

Random removal heuristic

This is the simplest heuristic. It randomly selects ω groups to remove from the yard, the only condition being that a group must be time related to at least another group already in the destroy set. The aim of this heuristic is to diversify the search.

4.4.2 Repair heuristics

We now describe two greedy repair heuristics. As mentioned, the first heuristic is aimed at recovering feasibility. This heuristic is used in the first phase of the algorithm. The second heuristic tries to balance feasibility and

cost minimization and is always applied in the second phase after a feasible solution has been found.

Largest-first heuristic

The largest-first heuristic exploits as greedy principle the *dimension* of a group, defined as the product of the total length of the cars of a group and their duration of stay expressed in number of time steps. This value is a proxy for the area of the rectangle in the row-time plane. Observe that in this plane the spatial dimension is variable, whereas this dimension index has the advantage of being fixed. The groups are assigned to the yard in non-increasing order of this index and according to the randomization parameter p. Every selected group is assigned to the first feasible position in the yard filling direction. The aim of this heuristic is to minimize the used area in the row-time plane, which helps reaching feasibility.

Worst-first heuristic

The worst-first heuristic favors the worst allocated groups in order to give them the opportunity to be assigned first. This algorithm orders the groups by non-increasing values of $\triangle_{f_k}$. Groups are chosen by scanning the list with the randomization parameter p. Once selected, a group is assigned to the yard at the minimum cost feasible position with probability p, otherwise it is assigned to the first feasible position in the yard filling direction. Thus, this heuristic looks at both cost and area minimization.

4.4.3 Post-optimization procedure

Every time an improving solution is found we apply a post optimization process that removes and then reassigns each group to a more favorable position, if any. Groups are selected according to non-increasing values of $\triangle_{f_k}$.

The process is iterated until, after a complete scan of the sorted list of groups, no more improvement has been registered.

4.4.4 Master level local search

At the master level we choose to use simulated annealing as local search framework. We accept a new found candidate solution x', given a current solution x, with probability

$$e^{-(f(x')-f(x))/\tau}, \tag{4.9}$$

where τ is the temperature which starts from τ_{start} and decreases at each iteration i according to the expression $\tau_i = c\tau_{i-1}$, where $0 < c < 1$ is the cooling rate. We only accept new solutions that have not been accepted before.

4.4.5 Adaptive heuristic selection mechanism

As mentioned, we have designed a first phase in which the criterion for choosing the destroy and repair combination of heuristics is fixed and it results in the application of the largest-out (Section 4.4.1) and the largest-first (Section 4.4.2) pair of heuristics. This is because the largest-out and the largest-first heuristics is a destroy and repair combination that excels at feasibility, while the other destroy heuristics and the worst-first repair heuristic (Section 4.4.2) are particularly useful for generating good quality solutions. The adaptive selection mechanism of heuristics is used in the second phase of the algorithm. It is based on the scores π_j assigned to each destroy heuristic. The repair heuristic is always the worst-first heuristic. To select the destroy heuristics, we collect the scores, as suggested by Pisinger and Ropke (2007), over a segment of 100 iterations. The score π_{ij} of a heuristic i in a segment j is obtained from the score in the previous segment incremented, at each it-

eration, with the following values depending on the new obtained solution x':

- σ_1, if x' is a new best solution;
- σ_2, if x' is better than the current solution;
- σ_3, if x' is worse than the current solution, but it is accepted.

Since we accept only solutions not accepted before, a long term memory is needed in order to keep track of all solutions already accepted.

4.5 Computational experiments

We now present computational experiments. We first describe how test instances were generated, we then provide implementations details, and finally we discuss results obtained with the metaheuristic algorithm and with a commercial integer linear programming solver applied to the proposed formulations.

4.5.1 Generation of test instances

We have generated a set of 63 instances for the ARDAP problem using real-life data of the Gioia Tauro terminal. We have considered a time step of one day and a time horizon of 31 days. This results in a planning horizon of one month, i.e. four times larger than the usual horizon of one week. The reasons for this choice are the following:

- A longer planning horizon gives more challenging instances.
- Terminal expansion and volume increase could occur in the future, and would result in more difficult yard assignment problems.

- The current practice of planning with a time step of one day could also change. Container terminals normally plan with a time step equal to the length of a work shift, which results in four time steps per day, usually. If the Gioia Tauro terminal were to adopt this practice, the number of time steps per week would be equal to 28, close to the number considered in the generated instance set.

- Another change that could require solving larger instances in the length of the time horizon would be the inclusion in the planning of forecasts for weeks following the current one.

Furthermore, we observe that one-month instances are useful to assess the effect of the rolling horizon. By solving the full instance of 31 days we obtain a lower bound on what can be achieved by solving smaller problems with one week of time horizon for each day of the month.

At the time of our study, average speeds between yard positions where not available in the terminal operational database. We have used physical distances as proxies for handling times. The yard layout is the one described in the introduction. Each instance was generated by fixing the number of groups, and generating randomly the values for the numbers of cars and the duration of stay for each group according to discrete uniform distributions within historical ranges. Arrival times were considered to be uniformly distributed in the time period of 31 days, and every group must leave the terminal during this period. Tables 4.1 and 4.2 report average, minimum, and maximum values for a set of characteristics of the generated instances. We indicate by yard saturation degree at a time step the ratio between the total length of rows required to allocate cars at that time step, and the total length of the parking rows in the yard. The total length of required rows at a time t is computed by assigning the groups in the set $K(t)$ to consecutive parking rows. The sum of the length of the used row defines the total length

of required rows at the time t. This is clearly an optimistic saturation index because it does not account for interferences in the spatial allocation due to the duration of stay of the groups. These 63 instances are the feasible ones of a larger set. Feasibility was established by running one of the proposed formulations on the instance set.

		Yard saturation degree per time step			Number of cars per time step			Duration of stay per group (days)			Number of cars per group		
$\|K\|$	Instance index	avg	min	max	avg	min	max	avg	min	max	avg	min	max
20	1	0.63	0.10	0.85	2390	371	3238	8	2	14	410	27	958
	2	0.61	0.15	0.85	2327	564	3261	7	3	16	476	83	993
	3	0.62	0.20	0.85	2372	802	3254	6	2	15	503	60	930
	4	0.65	0.21	0.86	2490	836	3288	7	1	16	488	13	964
	5	0.67	0.20	0.86	2557	790	3297	8	2	15	449	30	973
	6	0.64	0.21	0.86	2451	820	3294	6	2	15	501	12	948
	7	0.57	0.23	0.74	2204	881	2839	7	1	16	439	13	916
	8	0.54	0.22	0.79	2053	853	3021	7	1	15	415	23	941
	9	0.59	0.35	0.86	2245	1338	3292	7	1	16	501	46	949
	10	0.58	0.21	0.78	2232	808	2967	7	2	16	440	26	936
	11	0.57	0.21	0.77	2185	838	2938	7	1	16	468	63	966
	12	0.59	0.32	0.80	2261	1224	3046	6	2	15	530	48	999
40	13	0.68	0.03	0.87	2584	122	3257	7	1	16	243	15	470
	14	0.64	0.02	0.85	2423	87	3212	7	1	16	239	20	487
	15	0.64	0.13	0.86	2423	491	3279	7	1	15	275	23	494
	16	0.64	0.13	0.86	2402	491	3276	7	1	16	242	10	497
	17	0.67	0.16	0.89	2530	613	3370	7	1	16	245	28	499
	18	0.63	0.02	0.92	2371	81	3490	7	1	16	238	10	481
	19	0.57	0.20	0.76	2142	767	2876	7	1	16	229	11	482
	20	0.59	0.08	0.73	2242	305	2783	7	1	16	222	10	465
	21	0.56	0.12	0.81	2110	444	3054	6	1	15	239	13	484
	22	0.62	0.14	0.78	2339	538	2961	7	2	15	243	12	483
	23	0.61	0.19	0.80	2325	732	3048	7	2	16	233	10	497
	24	0.56	0.06	0.81	2121	223	3078	7	1	15	229	10	481
50	25	0.56	0.12	0.77	2095	462	2893	7	1	16	160	20	290
	26	0.55	0.07	0.81	2092	255	3022	7	1	15	165	18	295
	27	0.58	0.01	0.80	2178	31	2987	7	1	16	162	11	298
	28	0.57	0.11	0.77	2151	423	2882	7	1	16	155	13	294
	29	0.56	0.06	0.82	2122	243	3065	8	1	16	145	12	293
	30	0.56	0.01	0.77	2101	49	2876	7	1	16	154	11	290

Table 4.1: Characteristics of the generated instances, Part I.

		Yard saturation degree per time step			Number of cars per time step			Duration of stay per group (days)			Number of cars per group		
$\|K\|$	Instance index	avg	min	max	avg	min	max	avg	min	max	avg	min	max
20	31	0.73	0.38	0.94	2785	1450	3615	7	2	15	572	14	997
	32	0.73	0.39	0.94	2771	1514	3604	9	3	16	466	46	956
	33	0.77	0.35	0.93	2947	1319	3572	7	2	16	554	56	965
	34	0.72	0.24	0.97	2765	930	3715	7	2	16	494	42	986
	35	0.72	0.22	0.93	2757	844	3543	7	2	15	527	68	972
	36	0.77	0.29	0.88	2935	1124	3381	8	2	14	472	67	985
	37	0.68	0.27	0.91	2625	1026	3475	6	2	14	529	17	935
	38	0.65	0.21	0.86	2490	836	3288	7	1	16	488	13	964
	39	0.67	0.30	0.90	2553	1169	3423	8	1	16	449	49	992
	40	0.69	0.26	0.91	2638	996	3474	7	1	15	545	26	970
	41	0.69	0.33	0.90	2652	1258	3427	7	2	16	533	13	916
	42	0.66	0.32	0.89	2532	1226	3419	7	3	16	465	22	893
40	43	0.73	0.10	0.97	2789	389	3690	7	1	16	281	41	496
	44	0.72	0.21	0.95	2742	828	3595	7	2	16	255	18	489
	45	0.73	0.24	0.91	2764	939	3435	8	1	16	263	12	495
	46	0.72	0.12	0.88	2740	454	3342	7	1	16	263	22	493
	47	0.72	0.18	0.98	2746	686	3714	7	2	16	269	18	489
	48	0.73	0.10	0.93	2754	383	3519	8	3	16	249	21	476
	49	0.68	0.03	0.87	2584	122	3257	7	1	16	243	15	470
	50	0.67	0.18	0.92	2534	705	3487	8	1	16	237	16	471
	51	0.73	0.24	0.91	2764	939	3435	8	1	16	263	12	495
	52	0.67	0.12	0.89	2525	457	3378	7	2	15	254	26	481
	53	0.72	0.12	0.88	2740	454	3342	7	1	16	263	22	493
	54	0.68	0.17	0.91	2598	628	3462	7	1	15	261	13	484
50	55	0.66	0.09	0.93	2494	343	3497	8	1	16	164	20	299
	56	0.68	0.17	0.93	2538	640	3490	8	1	16	163	11	299
	57	0.68	0.07	0.93	2556	246	3521	8	3	16	166	19	298
	58	0.66	0.12	0.89	2496	455	3349	8	2	16	169	19	300
	59	0.67	0.13	0.87	2496	507	3264	8	2	16	165	19	300
	60	0.67	0.06	0.92	2515	231	3490	8	3	16	160	19	298
30	61	0.78	0.32	0.96	2979	1205	3660	8	2	16	344	56	599
	62	0.78	0.15	0.99	2986	576	3760	8	2	16	327	25	600
	63	0.77	0.22	0.99	2954	829	3769	8	2	16	333	50	598

Table 4.2: Characteristics of the generated instances, Part II.

4.5.2 Implementation details

We have implemented the mathematical formulations by using the integer linear programming solver CPLEX 11.1. When reporting the experiments

with the two formulations $\mathscr{F}_1$, and $\mathscr{F}_2$ implemented in CPLEX we denote their results by the name of the formulation, i.e. $\mathscr{F}_1$ indicates the formulation as well as its CPLEX implementation. We have not tweaked the CPLEX parameters. We have only set the stopping rule for CPLEX to 3% of the integer solution gap. The computational experiments were executed on a computer equipped with two Xeon 3 GHz processors and 4 GB of RAM.

The metaheuristic algorithm was coded in C++. We have executed the tests for a maximum number of η iterations, and we report the results for different values of this parameter: $\eta = 50 \times 10^3$ and, $\eta = 200 \times 10^3$. The ALNS input parameters were determined by some testing on a subset of the instance set. In our tests we have found that good values for the σ parameters are $\sigma_1 = 2, \sigma_2 = 0.1$, and $\sigma_3 = 0.01$. We have used randomly generated integer values for ω in the interval $[\min\{5, |K| \times 0.2]\}, ..., \min\{12, |K| \times 0.8\}]$, and the randomization parameter p was set equal to 0.3. Regarding the master level simulated annealing parameters, we have used a τ_{start} value such that a new solution with objective function value differing by 0.5% from the initial feasible solution will be accepted with a probability of 50%. The cooling rate c was set in such a way that the temperature value at the final iteration is equal to a given value (1000 in our experiments).

We denote by ALNS the algorithm with the same objective function as formulations $\mathscr{F}_1$, and $\mathscr{F}_2$, and by ALNS-RH the modified version with the objective function (4.6). The γ_1 parameter for ALNS-RH was set in such a way that the second term of the objective function (4.6) is one order of magnitude larger than the first term.

The metaheuristic algorithm with the objective function of the formulations $\mathscr{F}_1$ and $\mathscr{F}_2$, and the additional term (4.7) is referred to as ALNS-PS for its peak shaving capabilities. We will report an example of using this algorithm with the parameter γ_2 set equal to one.

4.5.3 Results

We report in Table 4.3 the comparison between the formulations $\mathscr{F}_1$ and $\mathscr{F}_2$. The listed four instances are the only ones out of the 63 instances where the formulation $\mathscr{F}_1$ does not exceed memory limits. For these instances we report the objective function values at the best found solutions, as well as the gaps. Here and in the following tables the (percent) gaps are computed as $100 \times (\text{upper bound} - \text{lower bound})/\text{upper bound}$. The formulation $\mathscr{F}_2$ clearly outperforms $\mathscr{F}_1$ and it has been exclusively used in the following experiments.

Tables 4.4 and 4.5 compare $\mathscr{F}_2$ and ALNS. We list the lower bounds obtained by $\mathscr{F}_2$ at the end of the computation, the computational times, and the gaps. The metaheuristic algorithm obtains high quality solutions within short computational times, whereas CPLEX can be rather slow on some instances. Using a larger number of iterations for ALNS ($\eta = 50 \times 10^3$ versus $\eta = 200 \times 10^3$) improves solution quality only slightly. This is why in the following experiments we have used $\eta = 50 \times 10^3$. Formulation $\mathscr{F}_2$ is rather useful in assessing the quality of the metaheuristic because it provides good lower bounds. However, we already mentioned that the devised solution approach requires an iterative use of the algorithm for different data assumptions, and for the evaluation of different objective functions. Tables 4.6 and 4.7 assess the effect of the rolling horizon. Here formulation $\mathscr{F}_2$ and the metaheuristic algorithms solve smaller instances with a planning horizon of one week. The assignments of groups arriving on the first day of the planning horizon are then fixed, and a new instance is derived as long as the end of the planning horizon does not coincide with the end of the period of 31 days. The lower bound for this type of problem is the same of the one computed by assuming the full month knowledge. Therefore, the solution quality is measured in terms of the gap with respect to this lower bound. For consis-

tency, the reported gap for ALNS-RH considers as upper bound the value of the first term of the objective function only. Because of the rolling horizon framework $\mathscr{F}_2$, ALNS, and ALNS-RH present some infeasibility issues, both $\mathscr{F}_2$ and ALNS fail in eighth instances. The modified objective function of ALNS-RH allows a significant reduction of the number of infeasible solutions which are now only three. However, this advantage comes at the expense of a noticeable worsening of the handling times. We observe that the handling times resulting from the rolling horizon are on average four percentage points worse than those obtained when the instance is solved with the full 31 days horizon. This could be defined as the price to pay for the limited knowledge about the future. When we use ALNS-RH we have an additional eighth percentage points worsening which could be seen as the price of the more prudent assumptions of ALNS-RH. Whether the more costly, but more resilient yard plans obtained by ALNS-RH are valuable is a decision to be left to planners' judgment.

Figure 4.6 illustrates the flexibility given by the objective function term (4.7) in the ALNS-PS algorithm. The dashed line represents the handling profile induced by the yard assignment decisions computed by the ALNS algorithm for instance 13. This handling profile has a very high peak of 450000. By imposing a desired largest handling value of 350000 in the ALNS-PS algorithm we obtain the profile represented in Figure 4.6 by a solid line. This avoids the high handling peak. However, the yard plan computed by ALNS-PS has a total handling cost 7.6 percentage points higher than the one computed by ALNS.

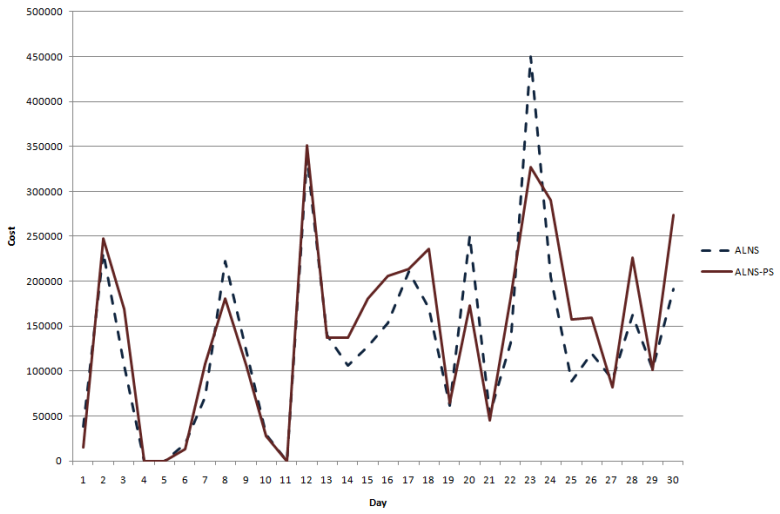

Figure 4.6: Comparison of handling profiles between ALNS and ALNS-PS algorithms

		$\mathscr{F}_1$			$\mathscr{F}_2$		
$\|K\|$	Instance index	Objective value	Time (sec)	Gap (%)	Objective value	Time (sec)	Gap (%)
20	3	5179419	798	2.7	5164749	27	2.3
	7	4054548	457	1.8	3987629	12	0.0
	10	4113687	429	2.0	4082843	27	0.5
	42	4678907	1366	0.1	4791606	62	2.4
	Average		763	1.6		32	1.3

Table 4.3: Comparison between the formulations $\mathscr{F}_1$ and $\mathscr{F}_2$ using CPLEX as integer linear programming solver. The gaps are computed as $100 \times$ (upper bound $-$ lower bound)/upper bound.

		$\mathscr{F}_2$			ALNS $\eta = 50 \times 10^3$		ALNS $\eta = 200 \times 10^3$	
$\|K\|$	Instance index	Lower bound	Time (sec)	Gap (%)	Time (sec)	Gap (%)	Time (sec)	Gap (%)
20	1	3992465	106	2.9	30	0.4	163	0.3
	2	4682893	75	0.4	30	0.8	159	0.8
	3	5043894	27	2.3	32	0.6	172	0.5
	4	4840143	108	2.7	30	0.8	165	0.8
	5	4453641	209	3.0	32	1.6	172	1.3
	6	4928722	112	0.9	32	0.5	172	0.6
	7	3987629	12	0.0	32	0.2	189	0.0
	8	3734106	13	1.6	40	0.1	175	0.1
	9	4710034	17	0.0	36	0.1	175	0.1
	10	4060795	27	0.5	49	0.0	180	0.0
	11	4368846	13	1.6	51	0.6	181	0.3
	12	5064482	13	0.0	48	0.1	194	0.1
40	13	4739218	454	2.4	32	1.5	242	1.4
	14	4525567	755	2.9	33	2.0	234	1.6
	15	5143900	1410	2.5	32	1.3	219	1.1
	16	4662518	306	1.5	33	1.0	244	1.0
	17	4722832	1316	0.9	32	1.6	233	1.1
	18	4705636	1812	1.0	32	2.3	225	1.7
	19	4109508	327	2.1	51	1.8	267	1.0
	20	4057705	860	1.9	54	1.9	268	1.7
	21	4379150	301	0.5	52	1.0	264	0.6
	22	4507207	467	1.0	53	2.5	291	0.6
	23	4241235	643	2.7	55	2.3	299	1.5
	24	4094234	341	0.9	50	1.1	247	0.9
50	25	3560322	844	2.6	48	2.0	255	2.2
	26	3674739	2959	1.6	51	1.3	283	1.2
	27	3752604	2847	0.5	52	1.4	278	0.9
	28	3448997	2865	3.0	52	2.1	260	2.3
	29	3279645	411	2.9	53	2.5	260	1.1
	30	3505008	820	2.3	50	2.5	284	1.6
	Average		682	1.6	42	1.3	225	1.0

Table 4.4: Computational results, Part I. The gaps are computed as $100 \times$ (upper bound $-$ lower bound)/upper bound.

		$\mathscr{F}_2$			ALNS $\eta = 50 \times 10^3$		ALNS $\eta = 200 \times 10^3$	
$\|K\|$	Instance index	Lower bound	Time (sec)	Gap (%)	Time (sec)	Gap (%)	Time (sec)	Gap (%)
20	31	5812495	22	1.6	34	0.4	202	0.4
	32	4777588	167	0.8	29	0.5	142	0.5
	33	5609889	711	2.0	30	0.7	130	0.7
	34	5101031	30	0.0	28	2.2	142	0.2
	35	5457213	183	2.4	31	1.4	160	1.2
	36	4870599	203	1.1	34	1.8	159	1.3
	37	5243680	41	1.6	49	0.4	181	0.4
	38	4840143	107	2.7	45	0.8	166	0.8
	39	4515430	61	2.2	46	2.4	158	0.9
	40	5560969	28	2.8	53	0.7	194	0.7
	41	5424514	491	0.8	48	0.4	165	0.3
	42	4678045	62	2.4	50	0.4	175	0.3
40	43	5477806	1123	2.0	29	4.0	174	3.8
	44	5118623	5290	2.6	29	1.9	200	1.7
	45	5284799	7811	1.5	29	1.5	217	1.5
	46	5035190	14052	2.9	31	3.0	227	2.5
	47	5399131	3686	2.9	29	2.9	176	2.0
	48	4881591	18963	1.2	28	5.0	169	4.0
	49	4739218	455	2.4	46	1.5	242	1.4
	50	4594082	9790	1.7	41	4.1	184	2.9
	51	5284799	7832	1.5	43	1.5	218	1.5
	52	4851810	377	2.4	42	3.3	191	2.1
	53	5035190	14942	2.9	46	3.0	226	2.5
	54	5046526	5645	2.8	43	2.0	217	1.4
50	55	3932682	10380	2.4	41	3.9	193	1.8
	56	3995910	510	0.5	44	1.6	216	1.6
	57	4084447	6736	1.3	43	2.0	182	1.9
	58	4030744	817	2.8	47	2.1	220	1.4
	59	3962657	7768	1.3	47	2.3	228	1.4
	60	3868591	6787	1.3	42	3.6	187	2.3
30	61	5225164	690	2.1	43	4.0	156	2.2
	62	5105719	38357	2.6	45	4.5	127	4.3
	63	5270800	559	0.4	43	3.7	129	3.0
	Average		4990	1.9	40	2.2	183	1.7

Table 4.5: Computational results, Part II. The gaps are computed as $100 \times$ (upper bound $-$ lower bound)/upper bound.

	Rolling horizon		
	$\mathscr{F}_2$	ALNS	ALNS-RH
Instance index	Gap (%)	Gap (%)	Gap (%)
1	7.2	4.3	14.8
2	2.9	1.8	8.8
3	4.8	2.0	11.8
4	n.a.	4.0	5.5
5	6.1	5.9	5.2
6	2.9	3.8	9.9
7	1.7	0.1	13.5
8	3.8	0.4	23.0
9	2.6	1.7	18.3
10	12.4	10.0	16.2
11	4.7	5.0	16.2
12	6.3	4.9	12.6
13	7.0	6.7	17.6
14	7.2	4.6	13.4
15	3.9	6.2	10.9
16	10.6	9.2	13.6
17	n.a.	n.a.	n.a.
18	6.5	6.2	12.6
19	5.1	4.6	11.3
20	6.0	8.2	19.5
21	7.2	3.1	17.1
22	6.8	2.6	20.1
23	7.3	6.8	13.5
24	5.8	7.4	16.8
25	10.2	5.3	14.2
26	9.6	4.1	17.3
27	6.2	5.8	14.6
28	7.8	5.2	13.9
29	11.7	5.8	12.0
30	13.7	6.7	14.7
Average	6.7	4.9	14.1

Table 4.6: Rolling horizon results, Part I. The gaps are computed as $100 \times$ (upper bound $-$ lower bound)/upper bound. We indicate by "n.a." whenever the algorithm fails in obtaining a feasible solution.

	Rolling horizon		
	$\mathscr{F}_2$	ALNS	ALNS-RH
Instance index	Gap (%)	Gap (%)	Gap (%)
31	n.a.	n.a.	7.0
32	5.5	n.a.	9.7
33	n.a.	n.a.	12.3
34	14.7	14.1	8.2
35	7.5	3.5	8.7
36	7.0	3.6	9.7
37	7.7	n.a.	5.4
38	n.a.	4.0	5.5
39	2.8	7.1	8.3
40	4.7	4.0	7.4
41	n.a.	n.a.	n.a.
42	1.8	3.5	4.9
43	4.7	2.1	16.2
44	7.9	5.7	11.6
45	7.4	3.9	16.3
46	6.4	6.0	12.7
47	8.4	5.3	13.7
48	6.9	7.4	12.2
49	7.0	6.7	17.6
50	8.9	10.0	13.5
51	7.4	3.9	16.3
52	9.4	7.2	12.3
53	6.4	6.0	12.7
54	11.4	6.9	16.0
55	7.8	8.7	18.0
56	7.1	5.4	9.1
57	7.6	6.1	15.3
58	10.5	7.7	16.3
59	7.9	6.3	13.5
60	6.0	7.0	15.4
61	5.5	8.8	11.4
62	n.a.	n.a.	n.a.
63	n.a.	n.a.	8.9
Average	7.3	6.2	13.5

Table 4.7: Rolling horizon results, Part II. The gaps are computed as $100 \times$ (upper bound $-$ lower bound)/upper bound. We indicate by "n.a." whenever the algorithm fails in obtaining a feasible solution.

Chapter 5

Conclusion and future work

In Chapter 1 we have first presented operations and typical optimization problems arising in container hub terminals. Then we have described management aspects of automotive hub terminals, highlighting the differences with respect to container terminals.

In Chapter 2 we have addressed the Ship Stowage Planning Problem, as it arises at a transshipment container terminal. We have proposed two Linear Integer Models $\mathscr{F}_1$ and $\mathscr{F}_2$, where the objective function to minimize is the sum of the transportation and the reshuffling times of containers. The ship stowage planning is an offline process, whereas the exact computation of the yard reshuffles can be done only online, since the yard configuration dynamically changes during the loading/discharging operations. Therefore, both $\mathscr{F}_1$ and $\mathscr{F}_2$, although they lie on the same assumptions, underestimate the yard reshuffles. However the lower bound returned by $\mathscr{F}_2$ is more accurate than the lower bound which $\mathscr{F}_1$ yields. We have devised a Tabu Search Algorithm and we have implemented the model $\mathscr{F}_2$. Both have been tested on a set of real instances. The computational results show the superiority of the Tabu Search Algorithm over CPLEX. Moreover, the Tabu Search algorithm is able to find optimal or near-optimal solutions with a small computational burden.

In Chapter 3 we have analyzed the routing problem for straddle carriers at a transshipment container terminal. We have focused on the pooling modality and we have derived scheduling and dispatching models. The computational experiments on real instances from the Gioia Tauro Container Terminal has highlighted both the effectiveness and the complexity of this operational modality. The observed reduction of the empty travels with respect to the gang modality could be further enhanced, by acting at the tactical decision level on the yard management. The re-organization of the yard should support the new operational modality for the quayside transport, for example by resorting to "off line", and thus inexpensive, movements of containers in the yard areas near to the loading points (housekeeping). In order to face the computational complexity of the problem, efficient heuristic algorithms for the scheduling models should be developed; on the other hand, more sophisticated dispatching rules for the real time model, based on a wider lookahead policy, could be devised.

In Chapter 4 we have described, formulated and solved a yard management problem arising in an automotive transshipment terminal. Several constraints and objectives resulting from managerial rules and policies were considered. Computational experiments show that the proposed metaheuristic algorithm obtains high quality solutions when benchmarked with a state-of-the-art integer linear programming solver. Furthermore, the metaheuristic algorithm can handle application specific issues such as a rolling horizon, and a manpower leveling objective. A solution approach based on the iterative use of this fast metaheuristic algorithm is then possible for the application under study.

The discussed problems share many common features: the strong interaction with other complex logistic problems (the ship stowage planning, for example, is strictly related to the quay crane scheduling, the yard management and the routing of the transportation vehicles); the dynamic and aleatory na-

ture of some processes that can hardly be taken into account in a deterministic optimization model (think, for instance, to the effect of yard congestion on the containers transportation time and thus on the productivity of quay cranes).

The scientific community is aware that improved terminal performances could be obtained by an integrated approach to the various operations connected to each other, and some preliminary contributions in this direction have been proposed (Won and Kim, 2009): this is a very difficult but challenging field.

As far as regards the nondeterministic issues, a viable approach seems to be the joint use of optimization and simulation techniques. In Mazza et al. (2010), in order to evaluate the effectiveness of the Tabu search algorithm for the ship stowage planning problem, we use a queuing network simulation model to estimate the number of reshuffles.

The next step could be the solution of the problem with an Optimization via Simulation approach, i.e. by embedding simulation into the tabu search algorithm as a tool for counting the yard shifts. A similar framework is under development to deal with the Straddle Carrier Routing Problem, in which a multi-agents based system is adopted in order to simulate the actual container terminal processes.

5.1 Acknowledgements

My first thanks are to my advisor Prof. Flavia Monaco for her continuous help and warm support throughout the doctoral period.

Prof. Gilbert Laporte and Prof. Jean-Franois Cordeau, during a great (professional and personal) period I spent at the CIRRELT in Montreal, gave me invaluable advice and suggestions.

I am grateful to Marcello Sammarra for his precious help, especially in

writing this thesis.

I will never stop thanking Prof. Manlio Gaudioso who has generously allowed me to become a member of his fantastic research team.

I thank Giovanna Miglionico, Giovanni Giallombardo, Enrico Gorgone, Annabella Astorino, Antonio Fuduli e Pina Bonavita, friends before colleagues.

Special thanks are due to Luigi Moccia, co-advisor but above all, ever-present friend and my guide through this amazing experience.

Bibliography

Alvarez, J. (2006). A heuristic for vessel planning in a reach stacker terminal. *Journal of Maritime Research*, 3(1):3–16.

Ambrosino, D., Anghinolfi, D., Paolucci, M., and Sciomachen, A. (2009). A new three-step heuristic for the Master Bay Plan Problem. *Maritime Economics & Logistics*, 11(1):98–120.

Ambrosino, D. and Sciomachen, A. (2003). Impact of Yard Organisation on the Master bay Planning Problem. *Maritime Economics & Logistics*, (5):285–300.

Ambrosino, D., Sciomachen, A., and Tanfani, E. (2004). Stowing a containership: the master bay plan problem. *Transportation Research Part A: Policy and Practice*, 38(2):81–99.

Ambrosino, D., Sciomachen, A., and Tanfani, E. (2006). A decomposition heuristics for the container ship stowage problem. *Journal of Heuristics*, 12(3):211–233.

Avriel, M. (2000). Container ship stowage problem: complexity and connection to the coloring of circle graphs. *Discrete Applied Mathematics*, 103(1-3):271–279.

Avriel, M., Penn, M., Shpirer, N., and Witteboon, S. (1998). Stowage planning

for container ships to reduce the number of shifts. *Annals of Operations*, 76(4):55–71.

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

Bish, E. (2003). A multiple-crane-constrained scheduling problem in a container terminal. *European Journal of Operational Research*, 144:83–107.

Bish, E., Chen, F., Leong, Y., Nelson, B., Ng, J.-C., and Simchi-Levi, D. (2005). Dispatching vehicles in a mega container terminal. *OR Spectrum*, 27:491–506.

Bose, J., Reiners, T., Steenken, D., and Voss, S. (2000). Vehicle dispatching at seaport container terminals using evolutionary algorithms. In *Proceedings of the 33rd Hawaii International Conference on System Sciences; IEEE*, pages 1–10.

Cordeau, J., Laporte, G., Legato, P., and Moccia, L. (2005). Models and tabu search heuristics for the berth-allocation problem. *Transportation Science*, 39(4):526–538.

Das, S. and Spasovic, L. (2003). Scheduling material handling vehicles in a container terminal. *Production Planning and Control*, 14(7):623–633.

De Monie, G. (1987). Measuring and evaluating port performance and productivity. UNCTAD Monograph on Port Management. UNCTAD/-SHIP/494(6).

Desrosiers, J., Dumas, Y., Solomon, M., and Soumis, F. (1995). Time constrained routing and scheduling. In Ball, M., Magnanti, T., Monma, C., and Nemhauser, G., editors, *Network Routing*, volume 8 of *Handbooks in*

Operations Research and Management Science, pages 35–139. Elsevier Science, Amsterdam.

Dubrovsky, O., Levitin, G., and Penn, M. (2002). A genetic algorithm with a compact solution encoding for the container ship stowage problem. *Journal of Heuristics*, 8(6):585–599.

ESPO (2010). ANNUAL REPORT 2009-2010. Technical report, European Sea Ports Organisation, Treurenberg 6 B-1000 Brussel / Bruxelles.

Evers, J. and Koppers, S. (1996). Automated guided vehicle traffic control at a container terminal. *Transportation Research Part A*, 30(1):21–34.

Gendrau, M. and Potvin, J.-Y. (2005). Tabu search. In Burke, E. and Kendall, G., editors, *Search Methodologies: introductory tutorials in optimization and decision support technics*. Springer-Berlin.

Gendreau, M., Laporte, G., and Guimarães, E. (2001). A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 133:183–189.

Giallombardo, G., Moccia, L., Salani, M., and Vacca, I. (2010). Modeling and solving the Tactical Berth Allocation Problem. *Transportation Research Part B: Methodological*, 44(2):232 – 245.

Glover, F., Laguna, M., and Mart, R. (2007). Principles of tabu search. In Gonzalez, T. F., editor, *SHandbook of Approximation Algorithms and Metaheuristics*. Taylor & Francis Group-London.

Goetschalkx, M. and Ratliff, D. H. (1990). Shared storage policies based on the duration stay of unit loads. *Management Science*, 36(9):1120–1132.

Grunow, M., Gunther, H.-O., and Lehmann, M. (2004). Dispatching multi-load AGVs in highly automated seaport container terminals. *OR Spectrum*, 26:211–235.

Grunow, M., Gunther, H.-O., and Lehmann, M. (2006). Strategies for dispatching AGVs at automated seaport container terminals. *OR Spectrum*, 28:587–610.

Gumus, M., Kaminsky, P., Tiemroth, E., and Ayik, M. (2008). A Multi-stage Decomposition Heuristic for the Container Stowage Problem. In *Proceedings of the 2008 MSOM Conference*.

Hartmann, S. (2004). General framework for scheduling equipment and manpower on container terminals. *OR Spectrum*, 26:51–74.

Imahori, S., Yagiura, M., and Ibaraki, T. (2003). Local search algorithms for the rectangle packing problem with general spatial costs. *Mathematical Programming*, 97(3):543–569.

Imahori, S., Yagiura, M., and Ibaraki, T. (2005). Improved local search algorithms for the rectangle packing problem with general spatial costs. *European Journal of Operational Research*, 167(1):48–67.

Imai, A., Nishimura, E., and Current, J. (2007). A Lagrangian relaxation-based heuristic for the vehicle routing with full container load. *European Journal of Operational Research*, 176:87–105.

Kang, J.-G. and Kim, Y.-D. (2002). Stowage planning in maritime container transportation. *Journal of the Operational Research Society*, 53(4):415–426.

Kim, K. and Bae, J. (2004). A look-ahead dispatching method for automated guided vehicles in automated port container terminals. *Transportation Science*, 38(2):224–234.

Kim, K. and Kim, K. (1999a). A routing algorithm for a single straddle carrier to load export containers onto a containership. *International Journal of Production Economics*, 59:425–433.

Kim, K. and Kim, K. (1999b). Routing straddle carriers for the loading operation of containers using a beam search algorithm. *Computers and Industrial Engineering*, 36:109–136.

Kim, K. H., Kang, J. S., and Ryu, K. R. (2004). A beam search algorithm for the load sequencing of outbound containers in port container terminals. *OR Spectrum*, 26(1):93–116.

Lee, Y. and Chao, S.-L. (2009). A neighborhood search heuristic for premarshalling export containers. *European Journal of Operational Research*, 196(2):468–475.

Lee, Y. and Hsu, N.-Y. (2007). An optimization model for the container premarshalling problem. *Computers & Operations Research*, (34):3295–3313.

Lodi, A., Martello, S., and Monaci, M. (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241252.

Martello, S. and Toth, P. (1992). Generalized assignment problems. In *Algorithms and Computation*, volume 650 of *Lecture Notes in Computer Science*, pages 351–369. Springer, Berlin.

Mattfeld, D. C. (2006). *The Management of Transshipment Terminals, Decision Support for Terminal Operations in Finished Vehicle Supply Chains*, volume 34 of *Operations Research Computer Science Interfaces Series*. Springer, Berlin.

Mattfeld, D. C. and Kopfer, H. (2003). Terminal operations management in vehicle transshipment. *Transportation Research Part A: Policy and Practice*, 37(5):435–452.

Mattfeld, D. C. and Orth, H. (2006). The allocation of storage space for transshipment in vehicle distribution. *OR Spectrum*, 28(4):681–703.

Mazza, R. M., Monaco, M., Sammarra, M., and Sorrentino, G. (2010). A simulation-optimization approach for drawing stowage plans for containerships. In *XXIV European Conference on Operational Research*, Lisbon, July.

Meersmans, P. (2002). *Optimization of container handling systems*. PhD thesis, Erasmus University Rotterdam, The Netherlands.

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(1):196 – 209.

Moccia, L. and Astorino, A. (2007). The group allocation problem in a transhipment container terminal. In *World Conference on Transport Research, University of California at Berkeley, June*.

Moccia, L., Cordeau, J.-F., Monaco, M. F., and Sammarra, M. (2009). A column generation heuristic for a dynamic generalized assignment problem. *Computers & Operations Research*, (36):2670–2681.

Monaco, M. F., Moccia, L., and Sammarra, M. (2009). Operations Research for the management of a transhipment container terminal: The Gioia Tauro case. *Maritime Economics & Logistics*, 11(1):7–35.

Nguyen, V. and Kim, K. (2009). A dispatching method for automated lifting vehicles in automated port container terminals. *Computers & Industrial Engineering*, 56:1002–1020.

Ning, Z. and Weijian, M. (2009). A Model for the Stowage Planning of 40 Feet Containers at Containers Terminals. *International Journal of Information Systems for Logistics and Management*, 4(2):41–49.

Nishimura, E., Imai, A., and Papadimitriou, S. (2005). Yard trailer routing at a maritime container terminal. *Transportation Research Part E*, 41:53–76.

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

Pinedo, M. (1995). *Scheduling - Theory, Algorithms and Systems*. Prentice Hall, Englewood Cliffs, NJ.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435.

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

Sciomachen, A. and Tanfani, E. (2007). A 3D-BPP approach for optimising stowage plans and terminal productivity. *European Journal of Operational Research*, 183(3):1433–1446.

Stahlbock, R. and Voß, S. (2008). Operations research at container terminals: a literature update. *OR Spectrum*, 30(1):1–52.

Steenken, D., Henning, A., Freigang, S., and Voss, S. (1993). Routing of straddle carriers at a container terminal with the special aspect of internal moves. *OR Spectrum*, 15:167–172.

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

Steenken, D., Winter, T., and Zimmermann, U. T. (2001). Stowage and transport optimization in ship planning. In Grötschel, M., Krumke, S. O., and

Rambau, J., editors, *Online optimization of large scale systems*, pages 731–745. Springer-Berlin.

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

Vis, I., de Koster, R., Roodbergen, K., and Peeters, L. (2001). Determination of the number of automated guided vehicles required at a semi-automated container terminal. *Journal of the Operational Research Society*, 52(4):409–417.

Vis, I., de Koster, R., and Savelsbergh, M. (2005). Minimum vehicle fleet size under time-window constraints at a container terminal. *Transportation Science*, 39(2):249–260.

Wilson, I. and Roach, P. (1999). Principles of combinatorial optimization applied to container-ship stowage planning. *Journal of Heuristics*, 5:403–418.

Wilson, I. and Roach, P. (2000). Container stowage planning: a methodology for generating computerised solutions. *Journal of the Operational Research Society*, 51(11):1248–1255.

Wilson, I., Roach, P., and Ware, J. (2001). Container stowage pre-planning: using search to generate solutions, a case study. *Knowledge-Based Systems*, 14(3-4):137–145.

Won, S. H. and Kim, K. H. (2009). An integrated framework for various operation plans in container terminals. *Polish Maritime Research*, 16(3):51–61.