Alma Mater Studiorum
Università degli Studi di Bologna

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Gestionale

Tesi di Laurea in Modelli e Metodi per il Supporto alle Decisioni

Algoritmi per il Problema del Caricamento Ottimo di Navi

Candidato:
Anna Franceschetti

Relatore:
Chiar.mo Prof. Alberto Caprara

Correlatore esterno:
Chiar.mo Prof. Marco Luebbecke

Correlatori interni:
Chiar.mo Prof. Paolo Toth
Ing. Valentina Cacchiani

Anno Accademico 2008-2009 - Sessione I

“Intuition ist immer noch eine gute Sache”

Paul Klee

Sommario

Questa tesi é frutto di una attivitá di ricerca svolta presso il Combinatorial Optimization & Graph Algorithms group, Fakultaet II - Mathematik und Naturwissenschaften Institut, Technische Universitaet Berlin.

Il progetto di ricerca presentato in questa tesi é relativo a tematiche proprie delle compagnie di navigazione ed é in particolare finalizzato allo studio del problema dello stivaggio dei container sulle navi cargo.

Secondo quanto emerso dalle recenti analisi, il tempo di sosta complessivo trascorso da una nave portacontainer ai terminal marittimi ricopre all'incirca il 60% della durata del viaggio. Da alcuni studi economici risulta che ció incide significativamente sul totale costo di movimentazione dei container ai porti (Pallottino e Sciomachen, 1999). É stato inoltre stimato che una gran parte di questo e' impiegata in operazioni di carico e scarico merci.

Obiettivo del lavoro é l'utilizzo di strumenti e tecniche della programmazione matematica al fine di elaborare una strategia di determinazione dei piani di stivaggio dei container che consenta la minimizzazione del tempo di permanenza della nave ai porti, attraverso la velocizzazione delle operazioni di handling. Ció si traduce in una riduzione del costo totale sostenuto dalla compagnia di trasporto navale e in un cruciale vantaggio competitivo.

L'ottimizzazione dello stivaggio delle merci su navi cargo é una operazione particolarmente diffcile e delicata che non puó prescindere da vincoli operativi, strutturali e di stabilitá riferiti alla nave, ai container, e alle particolari esigenze dei terminal marittimi.

Considerata la vasta complessitá del problema in questione, abbiamo scelto di iniziare l'analisi esaminando un sua versione semplificata. Tra le varie ipotesi adottate le piú considerevoli sono: i container da trasportare hanno tutti la stessa dimensione (20' o 1 TEU) e lo stesso peso, non vi sono carichi speciali, il numero di gru di banchina assegnate alla nave a ciascun porto é noto ed invariato,

la quantitá di carico da consegnare non eccede mai la capacitá della nave, infine, l'espressione "gruppo di container", che comparirá diverse volte nei capitoli seguenti, é usata per indicare l'insieme di container che partono dallo stesso porto ed hanno la medesima destinazione.

Riprendendo un approccio divisionale giá adottato da altri ricercatori (Ambrosino et all., Kang and Kim) abbiamo scelto di scomporre il problema in due sottoproblemi che interagiscono tra loro attraverso un continuo scambio di informazioni. Il primo é finalizzato a realizzare un piano di assegnamento dei container alle baie della nave, il secondo provvede a definire l'esatto posizionamento di ciascun container all'interno della baia.

Trattandosi di un progetto di ricerca ancora work-in-progress, questa tesi esaminerá solamente il primo dei due sottoproblemi presentando i passi mossi nello sviluppo di un algoritmo euristico per la definizione di un piano di stivaggio delle baie finalizzato a ridurre i tempi di movimentazione del carico dalla banchina alla nave (e vice versa).

Tale procedura si compone di tre fasi sequenziali, chiamate *pre stivaggio*, *stivaggio delle baie* e *local search*.

Il *pre stivaggio* segue un approccio per alcuni versi similare al *Problema dello Scheduling delle Gru di Banchina*. Questa fase prevede di suddividere la nave in macro aree dette parti, in modo tale che a ciascuna di esse sia assegnata una determinata gru di banchina. Quindi si procede ridistribuendo i container (tra le parti) cosi' da bilanciare il piú possibile il carico di lavoro delle gru. Il solo vincolo operativo considerato in questa fase é cosituito dalla capacitá della nave portacontainer, espressa in TEU *Twenty-Foot Equivalent Unit*, tutti gli atri vincoli per il momento vengono rilassati.

Lo *stivaggio delle baie* cosidera i container che sono stati assegnati a ciascuna parte ed elabora il loro piano di carico nelle relative baie. Anche in questo caso il solo vincolo operativo é il vincolo di capacitá, inoltre la procedura euristica sviluppata é basata sul noto modello del *Knapsack Problem*.

Dopo aver completato questo secondo step viene verificata la stabiltá della nave in uscita dai porti e, nel caso questa risulti non garantita, verrá eseguito un algoritmo di ricerca locale, *local search*, che si proporrá di ripristinare l'ammissibilitá della soluzione. L'intorno investigato (*neighborhood*) verrá generato effettuando due tipi di mosse: spostamento di un gruppo di container da una baia ad una altra (o scambio di due gruppi di container stivati in baie diverse) e spostamento di un gruppo di container da una parte ad una altra (o scambio di due gruppi di container stivati in parti diverse).

Analizzando nello specifico lo sviluppo del seguente lavoro, il primo capitolo illustra le caratteristiche del problema dello stivaggio dei container sulle navi, fornendo inoltre una breve rassegna della letteratura scientifica sul tema e una breve anticipazione della procedura risolutiva proposta. Nei successivi tre capitoli vengono analizzate in dettaglio le fasi principali che compongono l'approccio risolutivo. Nel quinto capitolo sono presenti una prima serie di risultati ottenuti testando l'algoritmo su diverse istanze e la descrizione della procedura risolutiva di un semplice caso. L'ultimo capitolo fornisce una serie di considerazioni riguardo lo stato dell'arte del lavoro e traccia i prossimi sviluppi della ricerca.

Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I want to thank my supervisor, Prof. Alberto Caprara, who gave me the boost to carry on this research, co-supervisors Prof. Paolo Toth and Ing. Valentina Cacchiani and the German advisor, Prof. Marco Luebbecke, for allowing me to join his team, for his expertise, kindness, and most of all, for his patience. I have furthermore to thank all the members of the Combinatorial Optimization & Graph Algorithms group, who welcomed me at Technische Universitaet Berlin and gave me the wonderful opportunity to share their research project. I am grateful to Prof. Anna Sciomachen, Universitá degli Studi di Genova, for the professional advices.

I would like to express my sincere gratitude to Laura and Fawzi for their patience and availability, to Castrense for all the useful suggestions regarding Latex use, to Walter for having been at my side and to all my friends and colleagues who have advised and encouraged me during this period.

At last, I wish to thank my relatives for the support given and especially for being always there for me in this achievement.

Contents

List of Figures

List of Tables

Introduction

Containerization is defined by the Containerization Institute as ”the utilization, grouping or consolidating of multiple units into a larger container for more efficient movement” [1].

Containers are basically metal boxes used for carrying goods and came onto the market for international conveyance of sea freight almost five decades ago. They have been designed for easy and fast handling of freight. Besides facilitating discharge and loading operations, the standardization of metal boxes introduced many advantages to the customers. In particular they protect the load against weather and pilferage, and enable improved and simplified scheduling and control resulting in a profitable physical flow of cargo.

The first regular sea container service began about 1961 with an international container service between the US East Coast and points in the Caribbean, Central and South America.

The breakthrough after a slow start was achieved with large investments in specially designed ships, adapted seaport terminals with suitable equipment and the availability (purchase or leasing) of containers. A large number of container transshipments then led to economic efficiency and a rapidly growing market share.

Today, over 60 % of the world’s deep-sea general cargo is transported in containers, whereas some routes, especially between economically strong and stable countries, are containerized up to 100 %.

The growing number of container shipments causes higher demand on the seaport container terminals, container logistics, and management, as well as on

[1]Containerization and Intermodal Institute, www.containerization.org.

technical equipment [Vas08].

In the last years, the increasing complexity of the logistic systems and the bigger impact of concurrent methods have accentuated significantly the importance of research in container terminal operations.

Considering the complexity of the optimization method and the reduction in time of the planning horizon, today objective methods are necessary to support decisions. Different logistic concepts, decision rules and optimization algorithms have to be compared by simulation before they are implemented into real systems.

There are many challenging operational problems associated with container shipping, and these problems are only growing in importance and difficulty as ships grow in capacity. One of the most difficult ones is the Container Stowage Problem (CSP).

The Ship Stowage Planning Problem

The stowage of a container ship is one of the problems that has to be solved on a daily basis by any company which manages a container terminal (Thomas, 1989).

Stowage planning is the core of ship planning, it involves different operators with different points of view, pursuing various objectives. Among them it is necessary to optimize the available space and prevent damage to the goods, the container ship, its crew and equipment. Moreover, it is preferable to minimize the berthing time of the container ship at the terminal (Atkins, 1991).

A good stowage plan solution has to take into consideration different factors such as ship's stability, container movement handling costs, cargo types, destination and departure port, etc.

Actually stowage planning covers an important rule in the container transportation business because it's directly involved in the evaluation of the shipping line's operating costs.

The biggest component of the transportation time and cost is the time employed loading and unloading at ports, where, on average a container ship spends approximately 60 % of her time [Pet90].

Therefore the Container Stowage problem concerns the determination of a viable configuration of containers that minimize the time spent for the loading and unloading operations.

The standard input data of this problem refers to the route, to the characteristics of the ship and of the containers.

The container stowage problem is a combinatorial problem the size of which depends upon the ship capacity (TEU units) and the container's supply/demand at each port. It is combinatorially explosive with the number of possible stowage configurations for a medium-sized container-ship being vast [2].

Even for the smallest vessel sizes, container stowage planning is a large-scale problem due to the large number of variables (e.g. vessel intact-stability, hazardous cargo segregation, the need to consider stowage across a number of ports) which require consideration.

Generally, many theoretically plausible solutions have been suggested in order to solve this problem which has been described as being NP-Hard [Bot92] [Avr98]. This is to say it is not possible to guarantee that an optimal solution can be found for commercial sized ships in a reasonable processing time.

This thesis deals with the problem of stowing a container ship. Our aim is to find a stowage plan that minimizes the berthing time of a container ship. In general, the stowage plan must take into account various strength requirements which concerns the ship's stability and the placement of containers. In this thesis we only deal with the placement of containers among the bays of the ship in order to minimize the time of load and unload at each port, without specifying the exact position of each container and taking into account only some stability conditions.

Ouline of the thesis

This thesis is structured as follows.

[2]Dillingham and Perakis (1986) state that the number of possible configurations for a 2000 TEU ship is approximately 3.3 times ten to the 5735th power.

Chapter 1 gives a general introduction to the *Ship Stowage Planning Problem* by taking a brief look at the main features, then introducing the proposed solution approach. In Chapter 2 the first phase of our resolutive method, the *Pre-stowage phase*, will be discussed. First the 0/1 linear model will be presented, followed by a description of the corresponding heuristic procedure for solving it. Following the same structural approach, Chapter 3 describes the second decisional phase, the *Bay Stowage Phase*, which completes the containers allocation among the ship bays. Chapter 4 shows how we check the stability of the ship and it presents our proposed exchange algorithm to make the infeasible solutions feasible. The following chapter, Chapter 5, applies our algorithm in solving some sample instances. The closing chapter, Chapter 6 presents the conclusions of this thesis, and recommendations for future work. This is followed by a bibliography.

Chapter 1

The Container Stowage Problem

As already mentioned the stowage of containers on the cargo ships is an increasing problem and in continuous evolution. The objective of this Chapter is to provide general information that can facilitate the comprehension of this problem. Moreover some notions about the cargo ship and the handling equipment used in today's container terminal will be provided. Furthermore a summary description of the different approaches to the container stowage problem developed in the last years will be given. Finally the assumptions made in this study will be presented and the developed method briefly explained.

1.1 The container cargo ship

As shown in Figure 1.1, the cargo space of a container ship is made up of cells where each cell is 20', long 8' wide and 4'3" high [Wil99]. Cells are grouped vertically into stacks which are in turn grouped into bays by number and variable dimension. Each bay is divided into above-deck and below-deck by hatch covers and sub-areas called holds. Each cell is a physically location where a container is to be loaded.

Each location is identified by three indices, each one consisting of two numbers that give its position with respect to the three dimensions. They are respec-

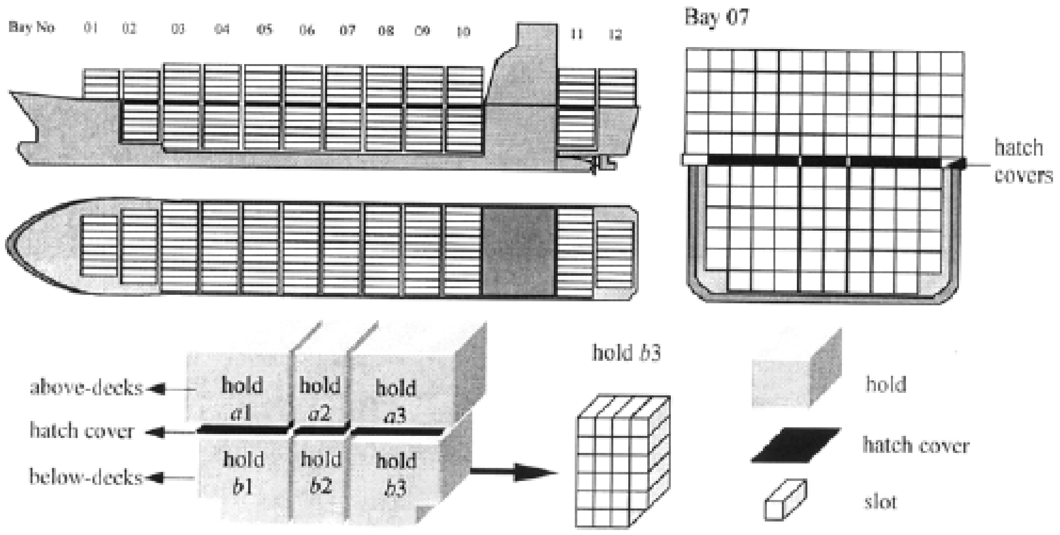


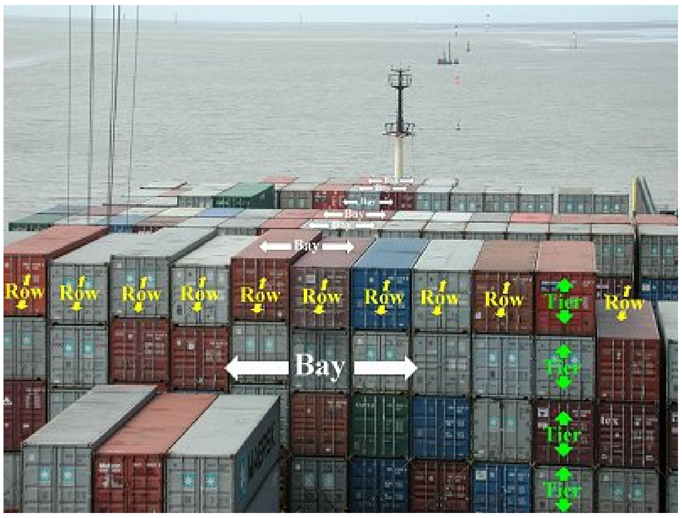
Figure 1.1: Horizontal and cross sections of a container ship [Kan02]

tively: a) *bay* that, as mentioned above, gives its position relative to the cross section of the ship; b) *row* that gives its position relative to the vertical section of the corresponding bay; c) *tier* that gives its position relative to the horizontal section of the corresponding bay.

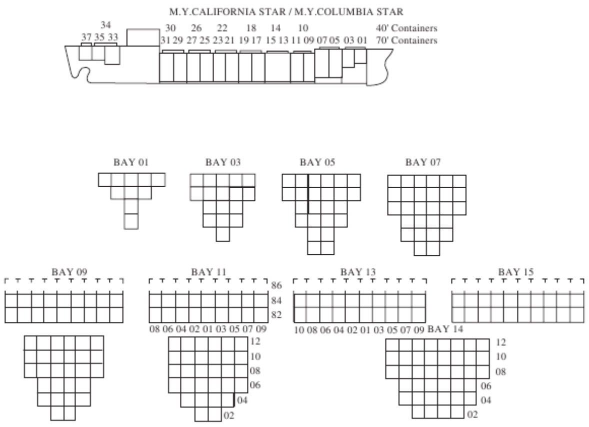
Nevertheless there are different identification systems adopted by the cargo company. Among them the most common consists of numbering the 20' bays with odd numbers, i.e. bay 01, 03, 05, etc., while two contiguous odd bays conventionally yield one even bay for the storage of the 40' containers, i.e. bay 04=bay 03+bay05, see Figure 1.2 [Amb04].

An important restriction is that the containers loaded in a below-deck hold can only be unloaded after all the containers loaded in the same above-deck hold are unloaded as well as the corresponding hatch cover. Moreover, the below-deck bays have some special restrictions upon the container dimension that can be accommodated, particularly correlating to the weight of the load.

[1](a) http://www.containerhandbuch.de, (b) [Amb06]



(a) Containers



(b) Ship

Figure 1.2: Bay plan configuration [1]

1.2 The main features of the problem

Since the Ship Stowage Planning is a very complex problem, in this Section some general notions are introduced about its most important features, in order to give a sufficiently exhaustive description of the topic covered in this thesis.

Size of containers As mentioned above the so-called standard container refers to a twenty foot (20') or forty foot (40') long container, with an of 8' x 8' section.

Generally containers are measured in terms of TEU (Twenty-foot Equivalent Units), where a TEU is a container 8 feet wide, 8 feet high and 20 feet long, and a 40 foot container is equivalent to two TEU. Containers of 40' require two contiguous locations of 20' each or, in other words they have to be located in an even bay.

Normally the 40' containers are loaded onto two contiguous odd bays, in other words they occupy an even bay. Moreover for security reasons the 40' containers cannot be loaded over empty locations and the 20' containers cannot be put over eight 40' containers or empty locations.

Type of containers As well as considering the physical dimensions of the cargo, the planner must consider how the cargo contents can restrict placements.

Apart from standard containers, some non standard ones are normally stowed in a container ship which do not conform to an ISO classification. A particular type of these containers is called hazardous and has specific stowage requirements which include segregation from other cargo.

There are some rules governing requirements to segregate hazardous cargo at specific distances from each other and from certain cargo [Wil99].

There can be also refrigerated containers that normally must be loaded near electrical outlets in order to maintain the required temperature during transportation.

Weight of containers The standard weight of an empty container ranges from 2 to 3.5 tons, while the maximum weight of a full container ranges from 20 to 32 and 30 to 48 tons for 20’ and 40’ containers respectively.

On the basis of their size, containers are located in the hold or in the upper deck in order to respect the weight capacity constraint.

Normally three classes of weight are considered: light (from 5 to 15 tons), medium (from 15 to 25 tons) and heavy (more than 25 tons). Moreover, the weight of a stack of three containers of 20’ and 40’ cannot be greater than an a priori established value.

Finally the weight of a container located in a tier cannot be greater than the weight of the container located in a lower tier having the same row and bay.

Destinations of containers In order to optimize the unload and load operations, a good rule normally adopted in the stowage planning is to load first the containers destined for the final stop of the ship and last those containers which have to be unloaded first. This is because, as already mentioned above, in the cargo ship the containers are loaded and unloaded only from the top.

Stability of the ship The stability of the ship is related to the weight distribution on it. Sometimes the stowage plan results in the instability of the ship, in which case it would be necessary to rearrange the load.

Referring to Golberg 1980, the cargo weight should be spread evenly to avoid heeling (an inclination from the vertical towards port or starboard), and ensure close to zero trim (which reflects the angle of the vessels fore to aft).

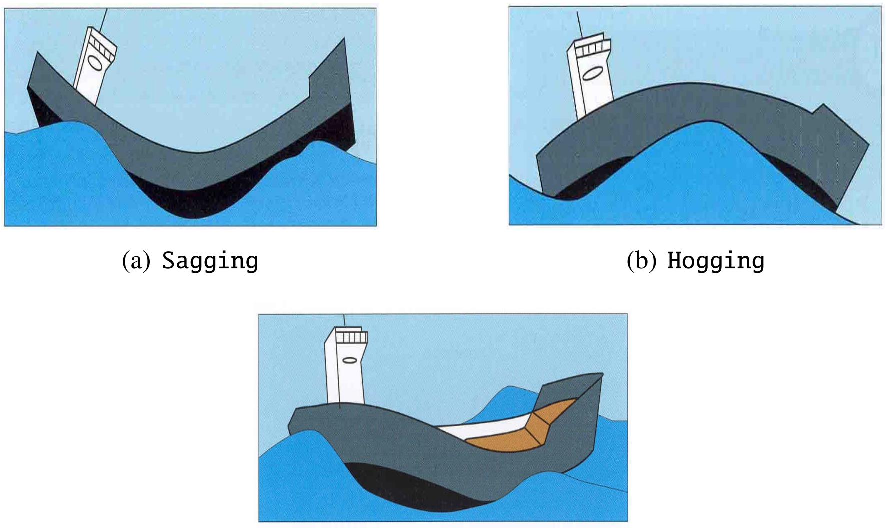
Irregular weight distributions also produce forces which can distort the physical structure of the ship, such as sagging Figure 1.3(a), hogging Figure 1.3(c) and torsion Figure 1.3(c).

The first occurs when the bow and the stern of the vessel are each supported by a wave crest, causing the hull to bend downward in the middle.

The second occurs when the crest of a wave occurs around the middle of a vessel causing stress at the hull longitudinally.

Both sagging and hogging cause “bending moments” (acting from bow to stern) in a vessel.

Finally torsion occurs because wave movement on either end of the ship are in opposite directions. This causes a “twisting moment” in the hull.



(a) Sagging (b) Hogging

(c) Torsion

Figure 1.3: Ship Stresses

Ballast (seawater) must be used to stabilize a vessel, but is considered additional cargo and should be kept to a minimum.

In order to control the vessel’s stability conditions three parameters can be considered:*meta centric height (GM)*, *heel* and *trim*.

The *meta centric height (GM)* of a ship is given by the distance between the center of gravity (G) and the meta center (M) of the ship, as illustrated in Figure 1.4.

In order to avoid the ship capsizing, the value of GM must be greater than the minimum allowable meta centric height of the ship. The only way to do this consists in placing the heavier containers at lower positions. Nevertheless observe that increasing the GM may conflict with the objective of minimizing the time of handling operations, particularly when the heavier containers are destined for nearer ports.

The *heel* is the inclination of a ship resulting from turning the ship in the direction of starboard or port, see Figure 1.4(a), and it must be as close as possible to zero with respect to the centerline.

The *trim* of a ship is the difference in draft forward and aft of the ship 1.4(b). As already suggested for the heel, the trim should also be at zero, or at least very close to it, in order to guarantee a good performance of the ship.

The stability constraints stated above (GM , heel and trim) can be linearized by representing them as the moments resulting from the composition of all weights of the cargo with respect to the vertical, transverse and longitudinal coordinates [Kan02].

Finally note that the ship must be loaded in such a way that its stability is guaranteed in a variety of weather conditions and that the stability constraints must be satisfied after each load/unload operation at intermediate destinations.

Note that in this study not all the described aspects of the problem are taken into consideration, but rather, as will be explained more thoroughly in Section 1.6, we have decided to concentrate our attention on only a few aspects for simplicity reasons.

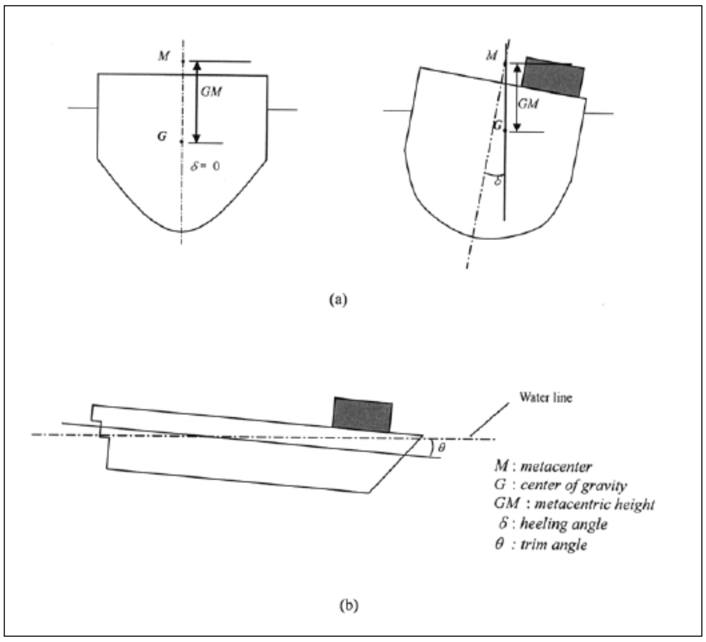


Figure 1.4: Stability of a ship. [2]

[2](a) GM and heel, (b) trim. [Kan02]

1.3 Literature review

During the last years many different approaches for solving this problem have been presented in recent literature.

Botter and Brinati [Bot92] propose an integer programming model considering different constraints related to stowage operations, characteristics of the ship and containers, loading sequence and crane movements and solve it by using an implicit enumeration method.

Since it is difficult to find the solution to the problem in a reasonable period of time, some researchers have developed different heuristic algorithms with the aim of maximizing the number of containers loaded onto a ship or minimizing the number of required shifting operations, where ”shifting” refers to the temporary operations of unloading and reloading upper containers in order to access a lower container designated for the current port.

Shields [Sch84] proposes an algorithm in which a number of possible loading plans are randomly generated and the best is selected.

Avriel and Penn [Avr93] developed a solution called the ”Whole Columns Heuristic“, a heuristic procedure where integer programs are solved after preprocessing the data.

Avriel et al. [Avr98] focus on minimizing the number of unproductive shiftings throught a heuristic procedure called the Suspensory Heuristic Procedure, in which the problem is treated as a two-dimensional stacking problem.

In order to minimize the time for shifting and crane movements, J-G Kang and Y-D Kim [Kan02] developed a heuristic method solution where the problem is divided into sub-problems which interact continuously with each other. The first one is solved by a greedy heuristic based on the transportation simplex method which assigns the container groups to the holds of the ship. The second one is solved by a three search method which determines a loading pattern of containers assigned to each hold.

Sciomachen and Tanfani (2003) [Sci03] present a heuristic method for the MBPP, the problem of finding optimal plans for stowing containers in a container ship, based on its connection to the three dimensional bin packing problem.

Martin et al. (1988) [Mar88] developed a heuristic algorithm that considers the quay cranes, in order to minimize their global longitudinal movement time and the total number of shifts in the successive ports.

The inconvenience of most heuristic methods is that they do not consider the stability of the ship, the movements of the cranes and the stowage plans at the subsequent ports. The Shields' algorithm [Sch84], for example, considers a lot of practical constraints, nevertheless it does not guarantee the optimality of the solution found and it takes a long computation time.

Finally we can observe that the integer programming approach guarantees the optimality of the solution when it can be solved, but it can't be solved in a reasonable time even for small-size problems.

1.4 The focus of the thesis

As is easy to infer, the variety of actors involved in the stowage problem results in several conflicting objectives to be aspired to. Furthermore taking many objectives into consideration at the same time is extremely difficult as well as determining the trade-off between them.

Consequently this study focuses on the *Ship Stowage Planning Problem* elaborating on the actual implications of the containers load plan for the cargo ship companies.

Our attempt is to develop an alternative methodology to find an optimal stowage plan for the containers on cargo ships which guarantees minimizing its berthing time.

Remembering the solution developed by J-D Kang and Y-D Kim (2002) [Kan02], the main problem is articulated into two sub-problems: the first one for assigning the container groups to the holds of the ship and the second one for determining a loading pattern of containers assigned to each hold. These two sub-problems interact with each other and each one is solved by using the information drawn from the other.

In particular the thesis deals with the the first sub-problem, it presents the first steps moved by our research work and describes the main ideas that we are developing.

1.5 The impact of Ship Stowage Planning on the transport costs

As suggested above the object of our method is to minimize the length of time the ship spends in the ports by speeding up the containers' load operations.

Observe that this time minimization is reflected in the reduction of the total transport costs incurred by the shipping company.

This Section analyzes the effect of the time berthing reduction on the total transport costs proposed by Pallottino and Sciomachen, 1999 [Pal99].

The following will try to underline the economic advantages of optimizing of the stowage plan.

According to the definitions given by Pallottino and Sciomachen, the cost function related to the containers' transport can be written as:

$$C_{tot} = \sum_{j=1}^{J} (CN_j + CP_j) + \sum_{c=1}^{C} CM_c \tag{1.1}$$

where J is the number of facilities employed in the containers' transport and C is the number of containers moved in the same time period.
This equation can be easily broken up into three parts:

CN_j is the cost of the ship j, it depends on the time spent by the ship at ports or in the vicinity;

CP_j is the penalty cost due to the ship's delay;

CM_c is the handling cost of the container c due to the use of the terminal resources.

More precisely the cost of the ship is given by:

$$CN_j = cn_j(tw_j + ta_j) + ca_l * ta_j \tag{1.2}$$

where cn_j is the ship cost per unit time and it depends on tw_j the wait time for a free berth on quayside, ta_j is the mooring time and ca_l the berth cost per unit time.

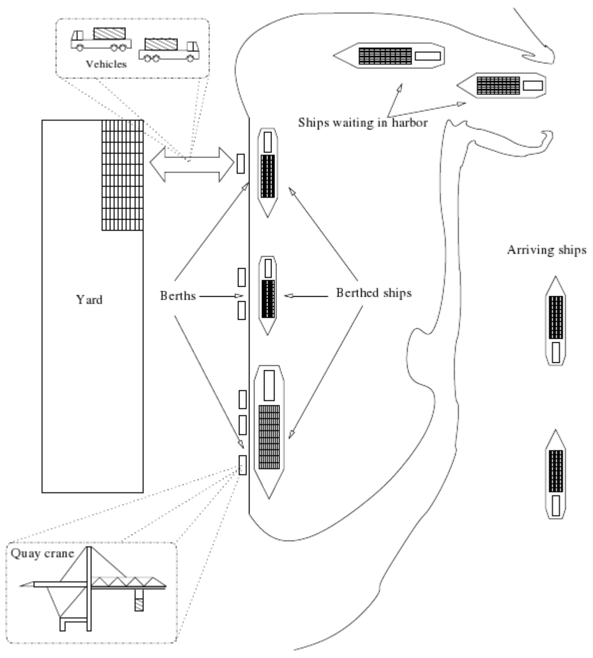


Figure 1.5: Maritime terminal layout

Observe that ta_j the berthing time is given by:

$$ta_j = L_j + am_j + pm_j \tag{1.3}$$

where L_j is the time spent for the load and unload operations, am_j is the time before the start of the operations and pm_j is the wait time after the end of the operations.
For what concerns the penalty cost CO_j it can be evaluated following alternative approaches depending on the the terminal and the ships features, but here I will not dwell on it.

Lastly, CM_i the handling cost of the container c, is given by:

$$CM_c = \sum_{i=1}^{I} cl_i t_{ic} + \sum_{r=1}^{R} cr_r t_{rc} + \sum_{w=1}^{W} cs_w t_{wc} \qquad (1.4)$$

where I is the total number of cranes employed by the container c , cl_i is the crane cost per unit time and t_{ic} is the time spent by the crane i handling the container c. R is the number of transport vehicles, cr_r is the cost per unit time of the terminal resource r and tr_{rc} is the time in which the container c uses the resource r. W is the number of storage areas used by the container c, cs_w is the price per unit time of the storage area w and t_{wc} is the the time spent by the container c into the area w.

Observing the last cost expression, formula (1.4), it is easy to understand that the reduction of the crane's movements through an optimal stowage of containers on the ship involves the decrease of the container handling cost CM_c.

Finally, from the previous considerations, it is easy to infer that we are trying to reduce this cost particularly by minimizing the total handling time L_J, which weighs on the berthing time ta_j, formula (1.3), which in turn increases the total cost of the ship, formula (1.1).

1.6 Relevant assumptions

Considering the high complexity which characterizes the maritime transport systems resulting from the numerous actors involved, the plurality of employed handling equipment and many other factors in part previously described, it is important to observe that this studio focuses on only a few of these aspects.

Particularly, in order to obtain a simplified version of the problem, the following assumptions have been adopted.

1. Containers to be delivered on the container ship have the same dimension and the same weight (20 ft standard containers). For this hypothesis the total weight of the containers loaded in each bay is linearly proportional to their number as occupied space.

2. The route of the ship's tour is given.

3. The number of containers to be delivered from one port to another is known for all the pair of ports included in the tour.

4. At the starting port the ship is empty.

5. At each port the number of containers to be loaded doesn't exceed the capacity of the ship.

6. The unit cost (time) related to the crane's movements is the same at all ports.

7. All the containers are ordinary containers, in other words there are no special containers.

8. The number of cranes that load and unload the container ship is the same at each port and it is known beforehand.

Besides these points, for a better understanding of the proposed solution, it's important to specify some additional notations:

1. It's a defined group of containers each set of containers with the same departure and destination port.

2. Since the number of cranes is known, the set of available locations of the ship (bays) is divided into different subsets, (for simplicity in the following I'll call them "parts") such that all the containers that are destined for each part will be loaded and unloaded by the same crane. This assumption is important in order to split the groups of containers among the cranes.

3. To assure the stability of a container ship several constraints must be satisfied. For the above-mentioned assumptions, in this simplified problem version it is not possible to evaluate all the stability constraints that are normally involved in the Ship Stowage Planning problem, but rather only a fraction of them.

1.7 Solution approach

The proposed solution is a decomposition solution approach consisting of three decisive phases which operate in a greedy way.

The first one, called *Pre Stowage Phase*, constructs an initial generic solution by splitting the ship into different parts (each ones formed by a subset of bays) and assigning to each one a group of containers in order to balance the load of work of each crane at each port.

The second one, called *Bays Stowage Phase*, improves the solution previously found by defining a more detailed load plan of the containers among the bays of the preassigned part.

The third one, called *Local Search Procedure* is used when the obtained solution is not feasible. In this case it's looked for modify it by using an exchange algorithm which is based on a local search techniques.

Observe that herein the main principles of the proposed resolutive procedure are simply summarized in order to facilitate the comprehensions of the following chapters, where these three resolutive phases will be examined in more detail.

Chapter 2

The Pre Stowage Phase

This Chapter focuses on the first decisional phase of the proposed resolutive procedure called ”*Pre Stowage Phase*”.
First a brief general description of the analyzed problem is given and furthermore the relative 0/1 linear model is introduced.
Finally the developed heuristic approach is presented in detail.

2.1 The problem

In this first step we provide to split the set $\mathbb{I}$ of container groups into different partitions. This way allows us to solve separately the loading problem in each part of the ship without considering the stability conditions, which will be successively considered.

At first we partition the vessel into different parts such that at each one corresponds a quay crane which must load and unload all the assigned containers.

Then we allocate the containers among the parts trying to balance at each port the loading works of the cranes as much as possible.

Observe that in the recent literature we can easily find other analogous decomposition approaches which have been developed in order to reduce the complexity of the overall problem. Some of them have been proposed by Wilson and Roach and Ambrosino, Sciomachen, Tanfani.

Analogously it is also easy to find other different solution approaches which have the purpose to minimize the working time of cranes, like the Cranes Scheduling Problem developed by Yi Zhu and Andrew Lim.

Observe that our resolutive method tries to combine together these two approaches in order to define a load plan which allows to minimize the loading and unloading times of containers, reducing contemporaneously the problem's complexity as much as possible.

2.2 Differences and analogies with the Quay Crane Scheduling Problem

According to the assumptions done, in the proposed solution we do not consider the destination constraints which strongly affect the computational time (Wilson and Roach) [Wil99].

In particular, we do not take care that the destination's constraints force containers which have to be unloaded at first to be stowed into the highest tiers. Observe that these constraints from the operative point of view are very important.

For simplicity in the following we hypothesize that the unloading time of each container is the same and it is not affected by the position in which it is stowed. Moreover, we suppose that there are no "empty moves" which increase the total berthing time.

These simplifications imply that the loading and unloading time of every container is the same and equal to a unit time "ut". All that involves that the total working time of one crane at one port is directly proportional to the total number of containers loaded and unloaded by that crane at that port.

The basis idea of our solution method and in particular of this first phase, derives from the main concept of the scheduling problem. As it is easy to observe the *Pre Stowage Phase* and the Quay Crane Scheduling Problem (QCSP) have the same aim. Nevertheless both the introduction of some simplifications and the will to highlight some particular aspects which are not carefully considered in the cranes scheduling problem involve some significant differences among these two approaches.

At first our solution does not take into consideration the set of spatial constraints recently studied in the Cranes Scheduling Problem, in particular the most important "non-crossing constraint" which claims that crane arms cannot be crossed over each other simultaneously [Zhu04].

A further difference is the type of results obtained thought these two solution methods.
The cranes scheduling provides to determine the optimal sequence of jobs that each crane has to execute.[1] In other words for each part of the ship are determined not only the lists of the containers to load, but rather the sequences of loading and unloading containers from or onto a container ship.
Differently our method restricts itself to elaborate a container stowage plan without taking care of the loading and unloading precedences.

To conclude, it can be argued that the suggested solution has some visible commonalities with the quay Crane Scheduling Problem, nevertheless on the whole these two resolutive approaches remain different.

2.3 The 0/1 linear model

In this Section we introduce the 0/1 Linear Programming model relative to the first phase.

At first we present the notations used in the formulation.

Indices

i index for container groups, $i = 1, \ldots, |\mathbb{I}|$

j index for parts of the ship, $j = 1, \ldots, |\mathbb{J}|$

k index for ports, $k = 1, \ldots, |\mathbb{K}|$

[1]Is defined job each loading or unloading operation executed by a crane.

Parameters (given)

c_j capacity of the part j

n_i number of containers of the group i

P_k subset of container groups ($P_k \subseteq \mathbb{I}$) which depart from the port k

D_k subset of container groups ($D_k \subseteq \mathbb{I}$) destined for the port k

Parameters (to be estimated)

t_{kj} total time of work of the crane assigned to the part j at port k

T_k berthing time at port k

TB total berthing time

Observe that:

$$\sum_{k \in \mathbb{K}} |P_k| = |\mathbb{I}| \tag{2.1}$$

$$\sum_{k \in \mathbb{K}} |D_k| = |\mathbb{I}| \tag{2.2}$$

Decision Variable

$$x_{ij} = \begin{cases} 1 & \text{If the group } i \text{ is loaded onto the part } j, \\ 0 & \text{Otherwise.} \end{cases}$$

Estimation of t_{kj}

$$t_{kj} = \sum_{i \in P_k} x_{ij} n_i + \sum_{i \in D_k} x_{ij} n_i \qquad \forall j, k \tag{2.3}$$

Estimation of T_k

$$T_k = max_j \quad t_{kj} \qquad \forall k \tag{2.4}$$

Estimation of TB

$$TB = \sum_{k \in \mathbb{K}} T_k \tag{2.5}$$

Objective function

$$Minimize \quad TB \tag{2.6}$$

Subject to

$$\sum_{i \in \mathbb{I}} x_{ij} n_i \leqslant c_j \qquad \forall j \tag{2.7}$$

$$\sum_{j \in \mathbb{J}} x_{ij} = 1 \qquad \forall i \tag{2.8}$$

$$x_{ij} \in \{0, 1\} \tag{2.9}$$

The objective function to be minimized (2.3) denotes the berthing time spent by the cargo ship during the entire tour. This value is given by the sum of the times spent by the cranes for handling containers at each ports. Observe that at each port the handling time corresponds to the latest completion time of work.

As it is shown in the t_{kj} formulation, the working time of the crane assigned to the part j at the port k is given by the sum of the number of containers that have to be loaded and unloaded onto/from the part k at the port j.

The (2.7) is the knapsack constraint, the main constraint of this model. It states that the total number of containers loaded into the part j does not exceed the capacity (expressed in number of 20' containers, or equivalent TEU) of this part.

The (2.8) is the assignment constraint, it ensures that all containers have been loaded. More precisely it states that each container must be loaded only one time and located in only one part of the container ship.

Observations As it is easy to observe the stability constraints which, as already mentioned, influence significantly the *Ship Stowage Planning Problem*, have been relaxed.

In this first phase we only deal with the allocation of containers among the parts of the ship in order to share out the loading and unloading works among the quay cranes without considering the vessel's stability and several constraints.

Observe that in this study we do not underestimate the important rule that this constraint plays in the determination of the load pattern of containers, but rather we limit ourself to consider it following.

2.4 The heuristic algorithm

The quay crane scheduling is the basis idea of this *pre stowage* heuristic procedure.

Considering a container group [2] which leaves the port p and must be delivered at port d, the algorithm looks for the crane which has the minimum working time at both these ports p and d.

When this crane is found, if the corresponding part of the ship during the rout from p to d has enough free capacity, the considered container group will be loaded on it. Otherwise, the algorithm will look for another different location. In particular it will repeat the same procedure considering the remain parts until to find a part which has enough free capacity.

Observe that this solution approach tries to allocate the entire container group onto the same part of the vessel avoiding whereas is possible its division.

In practice this placement allows to reduce the variety of container groups loaded onto each part and in such a way to ease the subsequent load planning of containers among bays and holds. At the same time, the choice to consider the entire container group instead of the single container enables to reduce significantly the computational complexity of the algorithm.

Observe that the most part of the container groups is allocate following this procedure and for that reason it can be defined ”standard“.

Nevertheless it's important to emphasize that this “standard“ procedure guarantees a good performance only when the dimension of the container group is no

[2]It's defined container group each set of containers with the same departure and destination port.

bigger than a prefixed value. For example in the case in which the number of containers of a group is bigger than the capacity of each part of the ship, has no sense to look for the crane with the lowest working time and try to insert the entire group on it because its capacity is certainly not enought.

When the number of containers is very high the solution which guarantees to balance in the best way the load distribution among the ship's parts (and then among the cranes) reducing at the same time the computational complexity, consists of allocating into each part an equal quantity of containers.
Observe that this allocation method is adopted only when the dimension of a group is bigger than a predetermine value.
This ”threshold value” can be defined by using different criteria since it is influenced by both the characteristics of the cargo ship and the number of cranes. In particular, as it is easy to infer, it should be bigger than the average capacity of each part of the container ship.

The two resolutive procedures performed for the standard size groups and for the special ones (with a number of containers bigger than the threshold value) are introduced in the follow Section and described more in detail respectively in Section 2.5.2 and 2.5.1.

2.5 The solution approach

As previously anticipated in this first phase we provide to allocate the container groups among the parts of the ship.

Observe that the aim we would want to reach in this *Pre Stowage Phase* is to find a containers load plan which allows not only to minimize the berthing time of the container ship, but also to ease the subsequent decisional phases of allocation of containers among bays and holds. For this reason we try to load onto the same part of the ship the greater number as possible of containers belonging to the same group in order to minimize the containers variability into each part.
Observe that the reduction of the variability will facilitate the following allocations of the containers among the bays and holds.

As it's easy to suppose, considering the variety of the involved factors our attempt to minimize the berthing time of the vessel trying at the same time to reduce the variability of containers [3] into each part is quite complicate.

The best solution which guarantees to obtain satisfactory results consists of allocating at first, when the parts of the ship are still almost empty, the bigger container groups. Gradually more the parts are filled and the free capacity decreases, more diminish the dimensions of the container groups to load.

To perform this system the allocation procedure starts sorting the container groups according their decreasing dimensions. In this way the bigger ones are considered at first. Then it considers the first group, it classifies it and proceeds into allocate it following the corresponding approach.

Here is reported the sequence of the performed moves.

Inputs (given):

- $\mathbb{I}$, $\mathbb{J}$, $\mathbb{K}$
- R_i set of remained containers of the group i
- n_i, c_j, t_{kj}
- p_i departure port of the group i
- d_i destination port of the group i

Inputs (to be estimated):

- Threshold value v

Start: The ship is empty

[3]With variability we indicate the presence of different groups of containers.

Main steps:

Step 1 *Initialization.* Consider the set $\mathbb{I}$ of container groups and sort them according their decreasing dimensions, then consider the first group $i = 0$.

Step 2 *Sorting.* Considering the selected group i compare its dimension n_i with the threshold value v.

- If $n_i \leq v$ proceed following the standard procedure.
- Otherwise follow the alternative one.

Alternative Procedure:

Step 3 *Division.* Divide the considered container group i into NP subgroups of n_{i_y} containers with $y = 0, \ldots, NP - 1$, and initialize a new empty set $R_i = \emptyset$. There can be two situations:

Case 3.1 n_i is divisible for NP then $n_{i_y} = n_i/NP$

Case 3.2 n_i is NOT divisible for NP then $n_{i_y} = \lfloor n_i/NP \rfloor$ and put the remain containers into R_i.

Step 4 *Initialization.* Start considering the subgroup $y = 0$ and the corresponding part j such that $j = 0$.

Step 5 *Evaluation.* Evaluate the free capacity c_j of the considered part j during the sea journey from the port p_i to the port d_i.
There can be two situations:

Case 5.1 if $c_j \geq n_{i_y}$, *the free capacity is enough.* Insert the entire subgroup y into the part j and update the free capacity of this part.
If there are other subgroups to insert select the next one $y = y + 1$ and the part $j = y$ and go to Step 5.
Else go to Step 6.

Case 5.2 if $c_j < n_{i_y}$, *the free capacity is not enough.*
Two cases:

Case 5.1.1 if $c_j \neq 0$. Insert c_j containers of the subgroup y into the part j. Update the free capacity of this part and add the remained containers $n_{i_y} - c_j$ to the set R_i.
If there are other subgroups to insert select the next one $y = y + 1$ and the part $j = y$ and go to Step 5.
Else go to Step 6.

Case 5.1.2 if $c_j = 0$. Add all containers of the subset y to the set R_i.
If there are other subgroups to insert select the next one $y = y + 1$ and the part $j = y$ and go to Step 5.
Else go to Step 6.

Step 6 *Remained containers.* Consider the set R_i There can be two situations:

Case 6.1 if $R_i = \emptyset$. If there are other container groups to insert select the next one $i = i + 1$ go to Step 2.
Else STOP.

Case 6.2 if $R_i \neq \emptyset$ allocate these containers following the standard procedure (go to Step 7).

Standard Procedure:

Step 7 *Sorting.* For each part j consider its corresponding crane and evaluate the sum of the working times tt_j at both the ports p_i and d_i, so that $tt_j = t_{pj} + t_{dj}$. Sort all the parts according to their increasing value of tt_j and consider the first one $j = 0$.

Step 8 *Evaluation.* Considering the selected part j evaluate its free capacity c_j between the ports p_i and d_i. There can be two situations:

Case 8.1 if $c_j \geq n_i$, *the free capacity is enough.* Insert the entire group i into the part j and update the free capacity of this part.
If there are other container groups to insert select the next one $i = i + 1$ and go to Step 2.
Else STOP.

Case 8.2 if $c_j < n_i$, *the free capacity is not enough.* Two cases:

Case 8.2.1 if $c_j \neq 0$. Insert c_j containers of the group i into the part j. Update the free capacity of this part and the number of containers of the group i to be inserted such that $n_i = n_i - c_j$. Select the next part $j = j + 1$ and go to step 8.

Case 8.2.2 if $c_j = 0$. Select the next part $j = j + 1$ and go to Step 8.

2.5.1 The Alternative Procedure

This procedure is employed when a very numerous container group must be allocated.

As mentioned above, the allocation of container groups which have a dimension greater than the average capacity of the parts of the ship, requires to split the group into different subsets and to allocate them separately.
This procedure has been introduced in order to simplify the computational complexity of the algorithm and at the same time to not compromise significantly the ship's stability.

The basis idea consists of subdividing the container group into a number of subsets equal to the number of the parts, then to allocate each of these subsets into a part.

This solution approach allows to simplify the "standard" allocation procedure because it plans a priori the division of the container group. In this way it avoids to try to fill the entire container group into each part of the ship. In fact, considering the big group's dimension this operation results firmly superfluous.

Observe that the proportional allocation of the load among the parts enables to trade off the working times of cranes. Furthermore at the same time it lets to balance the weight distribution on the ship.

As it is easy to infer a homogeneous distribution of the containers among the parts is not always possible because of the parts capacity constraint (some parts may be more full than others) and different reasons. For example when the partition of the group into various subgroups gives a rest value.

In these particular cases the containers which have not been yet inserted in any parts (remained containers) will be allocated following the standard procedure. For this reason in the passages described above has been initialized an empty set R_i destined to contain the remained containers.

Then the algorithm after it divided the container group i into NC subsets and tried to insert them into the corresponding parts, considers the set R_i.

If there are some not-inserted containers, the algorithm considers and allocates them following standard procedure.

Note that, from the hypothesis that the number of containers to be delivered cannot exceed the capacity of the ship, derives that the number of container of the set R_i is certainly smaller than the threshold value. For this reason is not necessary to compare again the containers' number with this value but rather is possible to proceed directly with their allocation.

2.5.2 The Standard Procedure

As its name implies, this is the ordinary procedure used to allocate the most part of the container groups.

This procedure starts considering a determine group of containers i. Then according to its departure and destination ports respectively p_i and d_i, it computes the working time of the cranes at these two ports.

Once calculated the working time tt_j[4] the cranes are sorted according the increasing value of tt_j.

Remembering that the ship has been portioned in such a way that at each part corresponds one crane to refer to a crane means to refer to the corresponding part of the ship. For this reason in order to simplify the used variables, the algorithm refers only to the parts of the ship.

So after that the parts of the ship have been sorted, the first one is considered and its free capacity c_j between the ports p_i and d_i is evaluated. Observe that, as

[4]tt_j is the result of the sum of the working time of the crane j at both ports p_i and d_i, which are respectively the departure and destination port of the considered container group i.

it's easy to deduce, the free capacity of this part in the other trades is not relevant for the storage of the considered container group.

If the free capacity c_j is enough, the entire container group is loaded onto the considered part, otherwise there can be two different situations.
If the considered part is not full, a quantity of containers corresponding to its avaiable capacity will be loaded on it and the remained containers will be stowed into different parts.
Otherwise, if the considered part is full, the entire group will be allocated into one or more different parts.
In both these cases, after loading, whereas is possible, the containers onto the selected part j, the next part $j = j + 1$ is considered and the same passages are repeated until to place all the containers of the group i.

This procedure is repeated for all the "ordinary"[5] container groups.

Observe that all the containers must be loaded since there is the hypothesis that the number of containers to transport must be smaller than the available ship's capacity. If it does not happen the algorithm will return an error message.

2.6 The output data

This first phase of the algorithm provides to arrange the list of containers which have to be loaded onto each part of the ship.

This subdivision of container groups among the parts has the aim to balance and coordinate as much as possible the work of the quay cranes which have to load and unload the ship at each port.

As we already know, each of these parts consists of a subset of bays which in turn are subdivided into different holds.

As previously described because of the computational complexity of the problem in this study we restrict ourself to allocate the container groups among the bays. The assignation of containers to holds of the ship and the determination of the load pattern in each hold will be developed in a second moment.

[5]It is referred to the container group with the dimensions smaller than the threshold value.

The next step of our algorithms is the *Bay Stowage Phase*. It considers each part of the ship and the corresponding containers list, then provides to arrange the containers among the relative bays. It is decribed in detain in the follow Section.

Chapter 3

The Bay Stowage Phase

This Chapter focuses on the second decisional phase of the proposed resolutive procedure, called *Bay Stowage Phase*.

First a brief description of the problem is given and subsequently the relative 0/1 linear model is introduced.

Finally the developed heuristic approach is presented in detail.

3.1 The problem

In this phase we process the results obtained by the *Pre Stowage Phase* with the aim to define a load plan of containers among the vessel's bays.

More precisely considering the load lists previously elaborated we try allocate the container groups among the relative bays in the best way possible.

Observe that this stowage problem is solved independently into each part of the ship. This choice makes the process simple and does not involve any worsening in terms of extension of the berthing time.
Moreover still for simplicity reasons, according to the considerations previously done (see Section 2.4) we continue to consider the groups of containers instead of the single containers. Consequently observe that in this Chapter the term *container group* refers to a group of containers which have the same departure and

destination port and must be loaded onto the same part of the ship.

3.2 The objective of the Bay Stowage Phase

As it is easy to suppose a worst spread of containers may cause the concentration of all the weight on the same vessel's side and this may produce forces which can distort the ship structure.

It is important to observe that the vessel's stability has not been checked yet, this constraint has been relaxed. That's because to evaluate the ship's stability we need to know the exact load's position on the ship. In other words until that the loading patterns of containers among the bays is not defined the evaluation of the vessel's stability conditions is not possible.

Therefore, in order to avoid to determine a load plan which compromises significantly the ship's stability, our purpose is to balance as much as possible the load distribution among the bays.

Furthermore, for the same reasons previously exposed in Section 2.5, we also try to concentrate into the same bay the biggest number as possible of containers with the same destination port. In this way we facilitate the subsequently containers disposition inside each bay.

All this involves that we will continue to distribute the container groups among the relative bays taking care to split them as less as possible.

Finally observe that for the assumptions mentioned in Section 2.2, the berthing time of the ship is affected only by the number of containers handled by each crane. In other words it means that the objective function of the entire problem is influenced only by the assignment of the containers to the parts of the ship, or simply by the decisions taken in the *Pre Stowage Phase*.

All this implies that the decisions taken on this phase do not affect directly the ship berthing time but they influence only the feasibility of the solution in terms of both capacity and stability constraints.

3.3 The 0/1 linear model

This Section gives a mathematical formulation of the part of the problem considered in the *Bay Stowage Phase*.

Note that the model described below refers to the containers allocation problem in only one of the parts in which the ship has been partitioned.

Indices

i index for container groups, $i = 1, \dots, |\mathbb{I}|$

b index for bays of the ship, $b = 1, \dots, |\mathbb{B}|$

k index for ports, $k = 1, \dots, |\mathbb{K}|$

Parameters (given)

c_b capacity of the bay b

n_i number of containers of the group i

Parameters (to be estimated)

v_b number of containers loaded onto the bay b

$Mmax$ number of containers loaded onto the more full bay

$Mmin$ number of containers loaded onto the less full bay

$D\triangle$ difference of containers between the more full and the less full bays

Decision Variable

$$x_{ib} = \begin{cases} 1 & \text{If the group } i \text{ is loaded onto the bay } b, \\ 0 & \text{Otherwise.} \end{cases}$$

Estimation of v_b

$$v_b = \sum_i x_{ib} n_i \qquad \forall b$$

Estimation of Mmax

$$Mmax = max_b \quad v_b$$

Estimation of Mmin

$$Mmin = min_b \quad v_b$$

Estimation of D△

$$D\triangle = Mmax - Mmin$$

Objective function

$$Minimize \quad D\triangle \tag{3.1}$$

Subject to

$$\sum_{i \in \mathbb{I}} x_{ib} n_i \leqslant c_b \qquad \forall b \tag{3.2}$$

$$\sum_{b \in \mathbb{B}} x_{ib} = 1 \qquad \forall i \tag{3.3}$$

$$x_{ib} \in \{0, 1\} \tag{3.4}$$

As it is easy to note in the 0/1 linear model the ship's stability does not constitute any strictly constraint. However, as suggested in Section 3.2, this constraint is taken into consideration though the definition of the objective function (3.3). It consists of minimizing the gap existing between the more filled and the more empty bays. More clearly it means that all the bays of the considered part must contain approximately the same quantity of containers.

Constraints (3.2) and (3.3) are the basic combinatorial ones. The first one is the knapsack constraint, it underlines that each bay has a fixed capacity and the number of containers to load on it must be inferior than this value. The second one is the assignment constraint and states that each container group must be loaded only in one bay.

3.4 The heuristic algorithm

Since a cargo ship visits different ports and in each of them loads and unloads different quantities of containers, the stowage plan at each of these ports is affected by the decisions taken at the ports previously visited.

To cope with this interdependency of the stowage plans this solution approach do not define the load plan at each port separately but rather it makes the plans simultaneously considering all together the container groups to be transported and organizing their allocation.

As is easy to infer, with a view to balance the load on the ship, a good method consists of replacing at each port the containers destined for this port with the containers which depart from it. In this way the weight distribution does not suffer substantial variations and the ship's stability is not particularly compromised.

Obviously this solution approach can be adopted only when the number of containers to load is equal or bigger than the number of containers to unload. Differently, if the containers to load are less than those to unload there may be some problems, in particular when this difference is quite big and the containers to unload are concentrated on the same side of the ship.

Thus, in order to cope more carefully with these situations, we classify as *critical ports* the ports in which the containers to load are less than those to unload. In this way we can distinguish these ports and dedicate more attentions in the allocation of containers which are destinate for them.

In particular our resolutive method classifies the container groups according to their destination port before starting to allocate them and creates an insertion list. Firstly the container groups destined for a *critical ports* (called *critical container groups*) are considered and arranged according their increasing dimension, subsequantly all the other groups are taken into account and arranged still according their increasing dimension.

This order allow us to distinguish three different categories of container groups: the *critical container groups*, the *normal container group*, and the *last container group*.

To these three typologies correspond three allocation methods respectively:

the *critical allocation* (Section 3.5.1), the *normal allocation* (Section 3.5.2) and the *last group allocation* (Section 3.5.3).

Each of these allocation methods will be explained in detail in following.

The next Section introduces in detail the developed heuristic procedure relative to the *Bay Stowage Phase* enumerating the corresponding passages.

3.5 The solution approach

Inputs (given):

- $\mathbb{B}$, $\mathbb{K}$, $\mathbb{I}$
- $\mathbb{S}_j$ subset of container groups ($S_j \subseteq \mathbb{I}$) to load onto the part j [1]
- c_b, n_i
- b_j number of bays of the part j
- p_i departure port of the group i
- d_i destination port of the group i

Start: All the bays are empty.

For each part j:

Main steps

Step 0 *Preprocessing*. Let Des_{pj} the number of containers destined for the port p to load into the part j and Dep_{pj} the number of containers which depart from the port p to load into the part j. For each port p evaluate the difference between Des_{pj} and Dep_{pj} and create the set C_j of the *critical ports* according the follow criterion:

[1] This is the output of the previous phase.

- if $Des_{pj} < Dep_{pj}$ p is a *critical ports*;

For each port k:

Step 1 *Sorting*. Given the part j, consider the container groups belonging to S_j which leave k and sort them according their destination in the strength of the following criteria:

- First of all must be considered the container groups destined for *critical ports* and they must be ordered according their increasing dimensions.
- Then are considered all the remaining groups and they are ordered according their increasing dimensions.

In this way is produced an insertion list L.

Step 2 *Allocation*. Considering the container groups on the strength of the insertion list L, insert them according the following three criteria:

- container groups destined for *special ports*: *critical allocation*.
- all remaining container groups except for the last one: *normal allocation*.
- the last container group of the insertion list L: *last group allocation*.

Observations Note that the Steps 0, 1 and 2 must be executed separately for each part of the ship j. Furthermore the Steps 1 and 2 are performed for each visited port k. This means that considering a determined part j, for each port $k \in \mathbb{K}$ we consider all the containers which depart from k, classify them, sort them and allocate them following one of the three allocation approaches above introduced. These three allocation methods are introdiced in detail in the following Sections.

3.5.1 The Critical Allocation

This allocation method is reserved to the container groups which have to be delivered to a *critical port*.

The main idea of this allocation procedure consists of distributing the containers destinate to the same *critical ports* among the bays in the balanced possible way. In this way when these containers will be unloaded the ship's stability will not be appreciably compromised and the containers replacement will be not needed to restore the load's balancing.

More clearly, considering a container group i destined for the port k that have to be loaded onto the part j, the aim of this procedure is to split the containers among the corresponding bays in such a way to balance the weight distribution as much as possible.

Observe that this approach is used only when a container group i has a dimension $n_i > b_j$, otherwise it is allocated according the *normal allocation*.

Analogously to *alternative procedure* described in the Section 2.5.1, this procedure divides the container group i into b_j subgroups and tries to allocate them into the corresponding bays. Furthermore note that if there are some not-inserted containers because is not possible to divide the containers into b_j equal subgroups or some of the considered bays have no enough free capacity, as in the *alternative procedure* they are loaded onto the more empty bay.

Adopting this allocation system, when the ship arrives to a *special port* the discharge of well-balanced containers do not compromise the ship's equilibrium or at most slightly. Moreover, observe that if there are some containers which leave that port, they are loaded in such a way to compromise as less as possible the equilibrium of the cargo ship or to re-stabilize it.

3.5.2 The Normal Allocation

In this allocation method we provide to allocate the container groups inserting them into the more empty bay and trying to not split them.

In particular we look among the bays for the more empty one, then we evaluate its free capacity. If its free capacity is enough the entire container group will be loaded on this bay and the relative free capacity will be updated. Otherwise we allocate in this bay only a number of containers equal to the available capacity and we will repeat the entire procedure until to allocate all the remained containers.

Observe that the sets of containers to load onto each part of the ship determined at the previous phase, satisfy certainly the capacity constraints of the relative parts. In other words the quantity of containers that has been associated to each part does not exceed the capacity of this part. All this implies that during the allocation of the containers among the relative bays if the free capacity results no sufficient the algorithm will return an error message.

Let's considered the container group i to be loaded onto the part j, the relative *normal allocation*procedure is here below synthesized:

- evaluate the free capacity c_b of each bay b between the current port p (observe that $p = p_i$) and d_i and consider the more empty bay b'.
  There can be three cases:

  Case 1 $c_{b'} > n_i$ If the capacity $c_{b'}$ is enough, assign the entire container group to this bay b' and update its free capacity.

  Case 2 $c_{b'} \leq n_i$ If the free capacity $c_{b'}$ is not null but at same time it is no sufficient, load $c_{b'}$ containers on the bay b' and update its free capacity. Repeat the entire procedure until to allocate all remain containers.

  Case 3 $c_{b'} = 0$ If the free capacity of the more empty bay b' results null and there are some containers that have not yet been loaded the algorithm will return an error message.

3.5.3 The Last Group Allocation

Since the stowage of container groups according to the *normal allocation* previously described may result not uniform [2] the last group of the list L, which for the insertion order is the biggest one, is used whereas is required to restore the load balance.

In particular this procedure can be briefly summarized in two main cases:

- If all the bays have been equally filled the considered container group will be split and fairly distributed among them.

[2]The normal allocation focuses primarily on concentrating the biggest quantity of containers of the same group into the same bay and only in a second time it considers the stability conditions.

- If there are some bays more loaded than others, the containers will be arranged in order to restore the load balance as much as possible. In practice the available containers will be initially distributed among the more empty bays leveling the load until to using up all the containers.

The whole procedure is here below synthesized in a pseudo-code:

Last Group Insertion - Balancing Insertion

```
    int mini;   /* most empty bay */
    int maxi;   /* most full bay */
    int second; /* second-most full bay */
    int c_mini; /* number of containers loaded onto the most empty bay */
    int c_maxi; /* nunber of containers loaded onto the most full bay */
    int c_sec;  /* number of containers loaded onto the second-most full bay */
    while(there are some containers to insert){
        compute mini, maxi, second;
        compute c_mini, c_maxi, c_sec;
        if(c_mini ≠ c_maxi){ /* there are some bays more empty than others! */
            if(c_sec==c_mini){ /* there are no intermediate replenishment ↩
                levels! */
                recall procedure A;
                break;
            }
            else {
                recall procedure B; /* there are various intermediate ↩
                    replenishment levels! */
            }
        }
        else{ /* all the bays have the same replenishment level!*/
            recall procedure "Critical Allocation"; /* see section \ref{sect:↩
                CriticAll}*/
            break;
        }
    }
    delete mini;
}
```

Last Group Insertion - Procedure A

```
    input cti  /*number of containers to insert*/
    int num;    /* number of bays with the minimum replenishment level */
    int vax = c_max-c_min;  /* difference between the maximum and the minimum ↩
        replenishment level (expressed in number of 20' containers, or TEU) */
    int tot=vax*(num);  /* total number of containers required to balance the ↩
        load amog the bays */
    if(tot ≤ cti)
        while(there are containers to insert){
            load vax containers onto the bays with the minimum replenishment ↩
                level;
            cti = cti-tot;
            recall procedure "Critical Allocation"; /*  see section \ref{sect:↩
                CriticAll}, now all the bays have the same replenishment level!↩
                 */
        }
    }
    else{
        redistribute equally the containers to insert among the bays with the ↩
            minimum replenishment level;
    }
};
```

Observations This procedure is followed when among the considered bays there are only two replenishment levels. This means that to balance the replenishment levels is sufficient to consider the more empty bays and load on them a quantity of containers equal to the replenishment's difference between the more full and the more empty bays.

What may happen is that the number of containers to allocate is bigger than the quantity required to "level the load" or it is smaller.

In the first case we provide to load the containers onto the more empty bays until all the bays have the same replenishment level. Then we redistribute the remained containers among all the bay following the *critical allocation* which provides to redistribute them equally among the bays.

In the second case we considered the more empty bays and we rearrange the containers among them in such a way to reduce the gap between the bigger and the smaller replenishment level.

Last Group Insertion - procedure B

```
    /* input: cti number of containers to insert*/
    int num;    /* number of bays with the minimum replenishment level */
    int num_second;     /* number of bays with the second-highest level */
    int vax = c_sec-c_min;  /* difference between the second-highest and the ↩
        minimum replenishment level (expressed in number of  20' containers, or↩
         TEU) */
    int tot=vax*(num);  /* total number of containers to insert for leveling ↩
        the bays replenishment */
    if(tot ≤ cti)
        while(there are containers to insert){
            load vax containers onto the bays with the minimum replenishment ↩
                level;
            cti = cti-tot;
            recall procedure "Last Group Insertion - Balancing Insert"; /* Now ↩
                repeat all the procedure! */
        }
    }
    else{
        redistribute equally the containers to insert among the bay with the ↩
            minimum replenishment level;
    }
};
```

Observations This procedure is followed when among the considered bays there are more than two replenishment's levels. This means that to balance them we have to load on each bay a quantity ot containers equal to the difference between the filling's levels of the considered bay and the filling's level of the more full bay.

In order to ease this procedure we proceed gradually considering the bays with the minimum replenishment's level and the bays with "second minimum" replenishment's level and trying to balance their load. Then we continue analogously to the "Procedure A" considering the bay with "second minimum" replenishment's level in the same way as the more full bays. The only difference is that after balancing the fillings of the considered bays, instead to execute the *critical allocation* this procedure is repeated until the second minimum bay coincides with the more empty one.

3.6 The output data

At the end of this decisional phase we obtain for each bay the corresponding list of containers to load.

As it was previously described each bay is made up of sub areas called holds, which in turn are made of various slots.

According to our decision to follow the decomposition approach introduced in Section 2.1, our attempt restrichts itself to find an optimal load plan of containers among the bays. This implies that the consequent distribution of containers among the holds and the determination of the loading patterns will be executed in a second moment.

Finally is opportune to observe that now we obtain the lists of containers which have to be loaded onto each bay but, for the reasons previously mentioned, we have now yet checked the feasibility of the obtained solution in terms of ship's stability.

The next step of the proposed solution method provides to test the unfeasibility due to the horizontal equilibrium [3] of the whole ship. Then when the obtained load plan causes the ship's unbalancing a local search procedure will be performed in order to make it feasible. In detail we propose an exchange algorithm which moves some containers trying to restore the load balance. This procedure is fully described in the next Section.

[3]The horizontal equilibrium means that the weight on the stern must be equal (within a given tolerance) to the weight on the bow.

Chapter 4

The Local Search Procedure

Local search is an emerging paradigm for the combinatorial search which has been recently shown to be very effective for a large number of combinatorial problems [Sch99].

Search algorithms are characterized by the idea of looking for a solution exploring the possible combinations of values of the variables or models of solution.

The *Local Search Procedure* generally starts from the current solution and while exists a solution belonging to an appropriate neighborhood, which is feasible and has a better goodness value, this solution replaces the current one.

Note that the definition of neighborhood is fundamental for search algorithms and it dependents on the specific problem under consideration.

After completing the containers' stowage planning among the bays, the ship's stability is checked.

In particular, when the solution found does not guarantee the stability of the vessel a local search will be performed in order to remove the infeasibility taking care to not compromise its goodness value.

In this Chapter we introduce at first the considered stability constraints, then we present the developed exchange method in detail.

4.1 The stability constraints

Now we check the feasibility of the stowage plan obtained in the previous phases. More precisely we verify the ship's stability constraints considering its horizontal equilibrium.
Observe that with this term we mean that the weight on the stern must be equal within a given tolerance L to the weight on the bow of the vessel.
Referring to Kam and Kim (2002), the stability constraint can be linearized by expressing it as the moment resulting from the composition of all the weights of the cargo with respect of the longitudinal coordinates. Such constraint results thus:

$$-L \leq \sum_i \sum_j \sum_b W_{jb} x_{ijb} n_{ijb} \leq L \tag{4.1}$$

Where:

$$\sum_j \sum_b x_{ijb} = 1 \qquad \forall i \tag{4.2}$$

Note that x_{ijb} is equal to 1 if some containers of the group i are loaded onto the bay b on the part j and n_{ijb} is the number of these containers. The parameter W_{jb} is the longitudinal coordinate of the center of the bay b, part j.
The sequence of passages performed by the proposed algorithm to check the solution's feasibility is here reported in a c-like procedure.

Feasibility check

```
Begin
  consider the obtained solution $x*$
  compute its horizontal balance value $\omega x*$
```

```
    begin
    if $(-L < \omega x* < L)${
      return "the solution found $x*$ is feasible";
      end;
    }
    else{
      recall procedure "local search";
      end;
    }
End
```

4.2 The exchange solution

If the solution $s*$ obtained at the end of the second phase does note satisfy the ship's stability we execute an exchange algorithm in order to make it feasible.

Precisely, we consider some containers which have been assigned to a determine bay b and a determine part b and we move them into an other different location with the aim to re balance the load on the ship.

As already mentioned, the goodness of the obtained solution s^*, identified by δ_{s^*}, depends directly on the total berthing time of the vessel. Then let $\delta_{s'}$ the goodness of the feasible solution s' obtained from the initial solution s^* after an exchange of containers' positions, the cost $\theta_{s'}$ results:

$$\theta_{s'} = \delta s' - \delta s^* \tag{4.3}$$

This value gives a sort of "feedback" of the locally optimal solution obtained and it can be viewed in some senses as a penalty function for obtaining feasibility in terms of ship stability [Dub02].

As in any local search the aim is to choose among a determined neighborhood the feasible solution s'' which has the minimum cost's value $\theta_{s''}$. Nevertheless, observe that using this procedure a feasible solution is not always reachable.

As already suggested, the definition of the neighborhood and the exchange strategy are relevant factors because they influence the "feasibility" of the solutions found.

In our algorithm have been adopted two exchange methods (Bay exchange and Part exchange) which will be introduced below respectively in definitions (4.2) and (4.2).

Definition *Bay exchange.*

Consider two container groups s and f with different dimensions[1] belonging to the same part but loaded onto different bays, respectively bg and bh ($bg \neq bh$). If and only if the bays' capacity constraints are not violated, exchange the locations between the two container groups, such that s is moved from bg and located in bh and vice versa.

Observe that this exchange method includes also the one-side exchanges. This happens when $ns > 0$ containers of the group s have been allocate into the bay bs and $nf = 0$ containers of the group f have been located into the bay bh (or vice versa). Given that $ns \neq nf$ if the capacity of the bay bh is enough the container group s is moved from bg to bh.

Definition *Part exchange*

Consider two container groups l and p, loaded onto different parts[2] respectively pk and pv.

If and only if the parts' capacity constraints are not violated, analogously to the *Bay exchange* procedure, the locations of the considered two container groups are exchanged.

Observe that this exchange method performs also the one-side exchanges, for the same reasons exposed in the *Bay exchange*. Furthermore it is important to note

[1]Observe that for container group we refer in this case to a set of containers with the same departure and the same destination loaded onto the same bay. Note that can also happen that the two considered groups have the same departure and destination ports. This coincidence is not relevant, the only conditions which have to be observed are that the containers must be loaded onto different bays and composed by a different number of containers.

[2]Observe that in the part exchange all the containers with the same departure and destination ports loaded onto the same part of the ship are considered, without taking care of the bay in which they have been allocate.

that after exchanging the containers' locations , the distribution of the load among the relative bays follows the *Bay Stowage Phase* procedure exposed in Chapter 3.

Remembering that our aim is to minimize the berthing time of the container ship, as it easy to infer the *Bay exchange* does not compromise the goodness value $\delta_{s'}$ of the solution found s' and this implies that $\theta_{s'}$, formula 4.3, is equal to 0. Given that, in order to reduce the computational complexity of this exchange procedure, we perform always the *Bay exchange* at first, in this way when a feasible solution is reached the algorithm stops. While the *Bay exchange* does not return any feasible solution the *Part exchange* is performed. Observe that in this case all the possible exchanges between the parts are tested in oder to find the feasible solution s' which has the minimum value $\theta_{s'}$.

Moreover, when an execution of the *Part exchange* procedure gives an infeasible solution s^I, before considering the next *Part exchange*, a *Bay exchange* starting from the current solution s^I is performed.

Here above is reported a pseudo-code procedure which shows schematically the sequence of moves executed by the developed algorithm.

Local Search

```
Begin
  /* initialization: consider the infeasible solution $s*$ */
    compute $\delta s*$

    begin
      recall procedure "Bay exchange";
      if($s'$ feasible solution found){
        return "found feasible solution $s'$;  /* observe that $\delta s' = \delta s*$ */
        end;
      }

      else{
        solution $r*$ /* let r* the best among the solutions found */
        int $\delta r* = 0$;
        while (all the possible exchanges have not been tested){
          recall procedure "Part exchage";
          if(no feasible solution found){
          recall procedure "Bay exchange";
          }
          if ($s''$ feasible solution found){
            compute $\delta s''$;
            if ($\delta s'' < \delta r * or \delta r * = 0$){
              $r* == s''$;
              $\delta r* == \delta s''$;
            }
          }
        }
      if (feasible solution found){
        return "found feasible solution $r*$";
        end;
      }
    }
End
```

4.3 Some critical issues

Considering that the *Local Search Procedure* is NP-hard, in other words the time required to solve it is an exponential function of its inputs' size, the complexity of this algorithm is very high.

In particular, hypothesizing that c is the number of parts in which the vessel has been partitioned, b the number of bays for each part and n the number of container groups, the computational complexity of the *Bay exchange* is proportional to the square of b and n, precisely it is $(O(b^2 * n^2))$.

Moreover, also the *Part exchange* results proportional to the square of p and n, $(O(p^2 * n^2))$. Nevertheless, it's important to observe that when the solution obtained by the *Part exchange* results no feasible, a *Bay exchange* procedure is performed. This choice enables to investigate a big neighborhood but, at the same time, it increases appreciably the computational complexity which results proportional to $O(p^2 * n^2) * O(b^2 * n^2)$.

For the hypothesis previously done in Section 1.6 the number of parts p in which the ship has been partitioned, coincides with the number of quay cranes which load and unload the ship at ports. Thus, considering the ordinary dimensions of a medium container ship and a general maritime terminal's lay out, it's easy to deduce that the cranes' number normally is a small value. This implies that the complexity of this *Local Search Procedure* is mostly affected by both the ship's dimensions, which in turn condition the value p, and the number of visited ports, which in turn affects the value n, for the definition of container group given in Section 1.6.

The high complexity of this procedure constitutes a substantial problem which compromises the efficiency of the developed method making it not suitable for large-scale problems.

As already mentioned the performance of a local search procedure depends on the neighborhood's definition and, in particular, on the method employed to explore it.
The efficiency of the local search changes on the basis of the trade-off between small and large neighborhood. A larger neighborhood would seem to hold promise of providing better local optima but it will take longer to search. Vice versa, a smaller one takes a short time to search but it does not guarantee the reaching of good feasible solutions.

Before discussing the computational results I would point out that the proposed heuristic approach is just a first development of our solution strategy, there are still many aspects which must be refined and the reduction of the computational time of the *Local Search Procedure* is one of the most important.

Our idea is that one of the further improvements should be the perfecting of the ship's stability check procedure in order to get more informations about its balancing conditions. In particular, we should not restrict ourself to knowing only if the ship is balanced or not, but rather, we should be perfectly informed on the ship's stability conditions. For example, when the ship is not balanced we should know the side in which it is more laded and the entity of this imbalance.
In this way, by exploiting the informations of the vessel's equilibrium is easier to develop a strategy which allows to define a smaller and more focused neighbor-

hood structure.
The way to do that will be taken up following in Section 5.3.

4.4 The output data

At the end of the proposed heuristic procedure we obtain the load plans of containers among the ship's bays. In detail, we achieve the lists of containers that have to be loaded onto each bay.

As already mentioned may happen that the algorithm is not able to determine a feasible solution which respects the stability constraint. There can be different factors which influence this situation, one of the future directions of our study should be the resolution of these situations. This purpose is better exposed in the follow in Section 6.

Finally, we have conducted a series of experiments to test the efficiency of our algorithm. In the next Section the obtained results are shown and some considerations are deduced.

Chapter 5

Performance evaluation

In this Chapter the performances of our algorithm on some computational experiments are evaluated.

In the first part the main features of the sample instances are briefly introduced. Then the obtained results are showed in detail and some considerations are derived. Finally a simple case study is presented.

5.1 The sample problems

As suggested earlier, the algorithm presented here restricts itself to finding a containers stowage plan for the containers among the ship bays, without specifying their exact position in terms of hold, row and tier. In other words, that means that the developed procedure can deal only with a sample problem version in which the ship is simply partitioned into parts and bays, without taking account of the above and below deck bays.

To investigate the performances of the suggested procedure some computational tests are done on 20 different problems. In these problems the number of parts in which the ship has been portioned and the capacity of each bay is a fixed value; in particular they are respectively 4 and 200 (TEU).

The variable parameters that are randomly combined to generate these problems are the number of bays contained in each part, which directly affects the ship's dimension, and the number of ports to be visited (3, 4, 5). More precisely, we consider three dimension sizes for the ship's capacity: small 2400 TEU, medium 4000 TEU and large 4800 TEU. Each vessel is partitioned into 4 parts which, according to the ship's dimension (small,medium, large) are respectively constituted by 3, 4 and 5 bays.

In these problems all the containers have the same dimension (20' or one TEU) and the same weight, moreover all the containers are ordinary, there isn't any ”special container”.

Since it is assumed that there aren't any weight levels, the ship's capacity can be expressed in terms of container dimension and more precisely it is defined as the number of containers which can be loaded onto a determined part or bay.

The number of containers which have to be transported from one port to another one is randomly generated and it is important to remember that it does not exceed the capacity of the ship. On the other hand, since this number does not change proportionally to the ship's dimension, the resulting problems will be characterized by different levels of replenishment.

An important consideration concerns the adopted check stability procedure. For the simplification introduced in this model (Section 1.6), it becomes quite complicated to apply the laws of mathematics and physics normally used to determine the buoyancy and stability of a ship. Without precisely knowing the exact position of each container on the ship, normally expressed in terms of bay, hold, row and tier, it becomes impossible to evaluate the forces and the moments which affect the vessel's stability.

As already mentioned before in this model we will take into account only the horizontal equilibrium of the ship for the meaning defined at Section 4.1. Nevertheless there isn't any mathematical formula used in real life problems either, to evaluate the horizontal ship's balancing , using the available position information.

For all these reasons we implement a stability check procedure which has the only purpose for testing the performance of the developed algorithm in cases for which the solution found is not feasible in the vessel's stability. In particular observe that this formula may result as ineffective in real life applications.

In Table 5.1 some tested instances are reported, all these computational tests have been performed on a PC Pentium M processor 1.86 GHz.
The column ”Destinations” gives the number of the visited ports without considering the first departure port ”0” and the following columns report the total quantity of containers destined for each of them.
The following two columns are related to the container’s informations. In particular the column “% Full” gives the average replenishment level of the ship, which is affected by the total number of containers that have to be delivered, and the column ”variance“ gives the standard deviation of the containers’ group. More clearly, as already suggested, for each visited port there are some container groups which leave this port and are destined for a following one, the standard deviation is the measurement of the variability of the dimensions of these container groups. A low standard deviation indicates that all the container groups have almost the same dimension, while high standard deviation indicates that their dimensions vary over a large range of values.
The BT and BT* columns are respectively the berthing time of the first solution found, whether it is feasible or not, and the berthing time of the solution found by the *Local Search Procedure* if this procedure is executed. Observe that to evaluate the total berthing time of the ship we use the formula (2.5), where the operating time of each crane is linearly proportional to the number of containers handled by that crane; in particular it will equal its double.

Note that in the instances marked by * and ** are the *Local Search Procedure* has been performed. More in detail the symbol * indicates that a *Bay exchange* has been performed and the symbol ** indicates that both the *Bay exchange* and the *Part exchange* have been performed.
The column θ shows the effects of the *Part exchange* procedure in terms of berthing time.
Remember that in those instances in which only the *Bay exchange* is performed the goodness of the solution is not compromised (see Chapter 4).
The columns CPU and CPU* report respectively the computational time for solv-

ing the problem without performing any *Local Search Procedure* and the computational time required by the exchange procedure performed when the first solution results as infeasible for the ship's stability.

Finally observe that sometimes after a certain number of exchanges the algorithm is unable to find a feasible solution, in this cases instead of the berthing time the initial ”nfsf”, which means no feasible solution found, is reported..

Observations Looking at the results reported in Table 5.1 one can see that the the algorithm computes the first solution, performing the *Pre-stowage phase* and the *Bay allocation* in a reasonable time. Furthermore it is possible to observe the CPU time variation according to the different number of visited ports, the different ship dimensions and the different quantity of containers to be transported. In the following Section 5.2, some graphics are presented that show the correlation between the CPU time, the variance of the container groups and fullness of the ship.

What is easy to note is the long computation time required for solving the *Local Search Procedure* (Instances 1,5,8,9) and the presence of one unsolved instance (Instance 8).

As already mentioned the neighborhood structure affects significantly the quality of the solution generated by a neighborhood search algorithm. With this method the long computational time required by the*Local Search Procedure* and the occasional failure in finding a feasible solution have reference to the typology of the investigate neighborhood. The Section 5.3 explains fully the reasons for these ”bad performances”.

5.2 Some correlation

As already mentioned the computational complexity (CPU) of the first two algorithm phases is linearly proportional to the dimension of the ship and to the number of visited ports. What is not so easy to deduce is how the other factors considered affect the CPU time. In this Section the correlation between the

Instance	Ship		Destinations	Containers							%Full	Variance	CPU	CPU*	BT	BT*	θ
	Ship Capacity (TEU)	Bays		Tot. (TEU)	d_1 (TEU)	d_2 (TEU)	d_3 (TEU)	d_4 (TEU)	d_5 (TEU)	d_6 (TEU)							
1**	2400	12	3	3247	107	1934	1111	0	0	0	63	429,76	0.11	3.72	1629	1629	0
2	2400	12	3	4212	107	1934	1271	0	0	0	78	393,44	0.14		2112		
3	2400	12	4	3799	180	1060	1019	1540	0	0	72	224,04	0.09		1906		
4	2400	12	5	3590	80	860	1019	1341	290	0	61	212,6	0.06		1851		
5**	4000	20	4	5520	380	1110	1880	2150	0	0	70	297,8	0.16	175.44	2765	2765	0
6	4000	20	3	7003	2100	1339	3564	0	0	0	91	576,36	0.37		3506		
7	4000	20	3	6670	1380	2010	3280	0	0	0	89	162,22	0.31		3338		
8**	4000	20	5	5259	180	260	1319	2540	960	0	73	246,65	0.1	392.76	nfsf	nfsf	
9**	4000	20	5	5779	1900	160	819	1940	960	0	55	475,5	0.23	1197.61	2895	2896	1
10	4000	20	4	5508	1900	1150	1008	1450	0	0	44	580,26	0.23		2762		
11	4000	20	6	5310	100	1150	198	350	1377	2135	56	309,85	0.16		2664		
12	4800	25	5	8380	1300	1100	2900	1080	0	0	73	595,82	0.41		4190		
13	4800	24	6	4387	356	120	117	1273	1113	1399	34	186,74	0.08		2197		
14	4800	24	5	11721	1260	2372	2362	1793	2702	0	89	571,64	0.53		5865		
15	4800	24	5	4621	140	240	1049	2436	338	0	50	395,43	0.14		2307		
16	4800	24	4	10400	3000	3000	4400	0	0	0	95	997,78	0.86		5200		
17	2400	12	4	3200	600	900	1700	0	0	0	90	335	0.1		1600		
18	2400	12	5	4000	800	800	700	1200	0	0	95	214,48	0.07		2000		
19	4800	24	3	6982	2007	1934	3041	0	0	0	78	399,47	0.33		3498		
20	2400	12	6	2308	207	214	1251	170	240	157	41	221,96	0.06	574.6	1165	1166	1

Table 5.1: Performance of the algorithm in 20 different sample instances

Ins.	%Full	Variance	CPU
1	44	299,95	0,04
2	57	354,15	0,07
3	17	82,48	0,01
4	46	549,17	0,12
5	68	338,54	0,06
6	38	232,37	0,04
7	25	206,83	0,03
8	43	565,91	0,12
9	7	427,47	0,16
10	58	922,66	0,46
11	57	438,49	0,13
12	42	505	0,12
13	64	767,87	0,27

Ins.	%Full	Variance	CPU
1	57	591,37	0,24
2	75	1000	0,26
3	76	539,86	0,25
4	79	512,45	0,26
5	8	51,344	0,01
6	69	655,27	0,23
7	71	1166,71	0,55
8	7	99,64	0,02
9	47	1085,09	0,44
10	40	687,7	0,2
11	50	857,26	0,2
12	54	253	0,1
13	73	621,99	0,26

Table 5.2: Performance of the algorithm considering two ship's dimensions: small (left) and big (right).

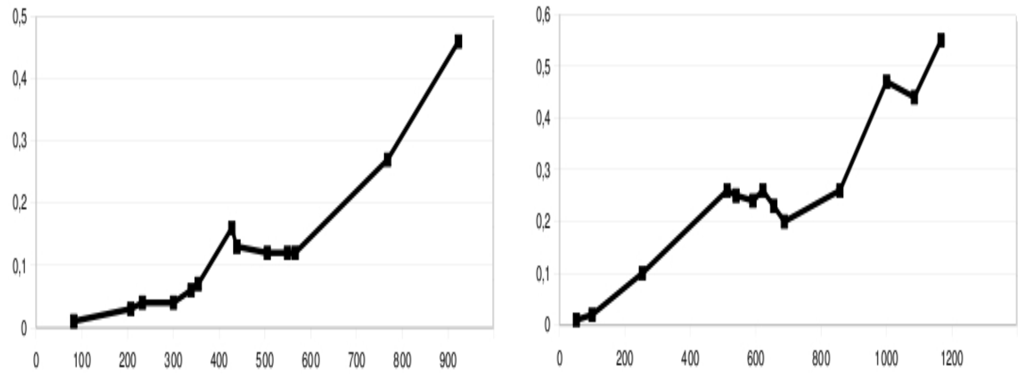
computational time's CPU, the variance and the occupation level (%Full) are examined.

To investigate these correlations, 26 (see Table 5.2) instances have been generated, each with a container ship which visits three ports, transporting different quantities of containers. The table on the left side reports the results obtained considering a small size ship (2400 TEU) while the table on the right shows is referred to a big size ship (4000 TEU). In each of these two cases, the ship features and the port's number remain equal, the only variable elements are the dimension of the container groups to be delivered and consequently the variance and the replenishment level.

The variation in computational times according to the different variance values of the container groups' dimensions is graphically reported in Fig. 5.2(a), 5.2(b). While the correlation between the computational time and the degree of ship fullness is shown in Fig. 5.3(a), 5.3(b).

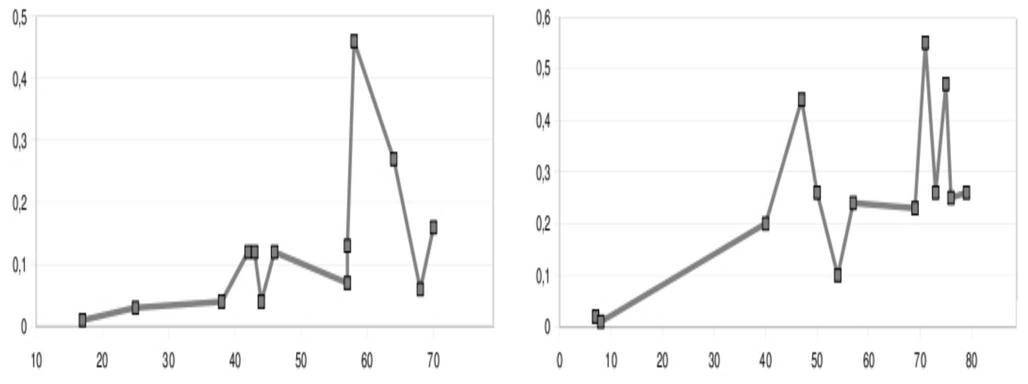
As is easy to notice, the computational time CPU tends to grow with the increase in variability of the container groups' dimensions. In other words the bigger the difference between the quantities of containers to be delayed from one port to

Figure 5.1: Trend of the computational time according to the variance of the container groups' dimensions



(a) `small` (b) `big`

Figure 5.2: Trend of the computational time according to the ship's occupation level (%)



(a) `small` (b) `big`

another, the more time is needed by the algorithm to compute a feasible load plan. In contrast, we can not individualize a significant trend between the occupation level of the ship and the CPU time.

5.3 Effects of the neighborhood size

Let’s look at Table 5.1 where it’s easy to observe that some results reflect a ”bad performance” of the *Local Search Procedure*. Accordingly, the long time required to perform it and the presence of unsolved instances constitute some ”critical issues”. They reduce the efficiency of the proposed method and limit its applicability.

At this point one should remark that the feasibility of each solution is expressed in terms of vessel’s balancing. The stability conditions adopted in these texts are deliberately very tight in order to emphasize this constraint. That accentuates the infeasibility of the solutions found. The real stability conditions should be tested.
Furthermore it’s important to point out that these unsolved instances are primarily due to a bad definition of the neighborhood structure.
In our model in fact the investigated neighborhood is defined simply by moving one or two entire container groups from their current location to different one. Observe that the choice to consider the entire container group instead of a smaller number of containers enables the reduction of the neighborhood dimension. But on the other hand the neighborhood quality is strongly affected by the dimension of each container group. This reduces the local search efficiency.

A more effective neighborhood search technique should try to move different quantities of containers instead of entire groups. This approach consents to improve significantly the quality of the solutions found but on the other hand it compromises the computational complexity.

What we can do to improve the efficiency of our method is to define a new neighborhood, narrower and more focused on restoring the ship’s balance. In these terms it’s easy to infer that the neighborhood definition is strongly affected by the stability check procedure. Our purpose is to improve the check stability procedure in order to collect more detailed informations about the vessel's stability conditions. In this way, using this information it is possible to define a limited range of moves focused on restoring vessel equilibrium. By "moves' we mean that we can find out where in the ship it's necessary to weigh down the load and where it's necessary to lighten it.

This system enables a reduction in computational complexity because it does not examine all the possible exchanges between the bays and the parts but rather it considers only the containers that are stowed in some specific locations.

5.4 A practical example

This Section shows in detail the resolutive procedure performed by the algorithm in solving the first of the previously presented instances (Table 5.1).

In this sample problem we consider a small size container ship. It is a 2400 TEU container ship with 12 bays of 200 TEU capacity partitioned into 4 parts. This vessel starts from the port 0 and visits 3 following ports; observe that for notational simplicity ports are indexed according to the order they are visited on a the ship route.

The follow Table 5.4 reports the input data of the considered instance.
In the first column the container groups are listed, identified by the departure and destination ports.
The second column reports the quantity of container which constitutes each group.

The first stowage plan obtained performing the first two phases of the described algorithm (*Pre Stowage* and *Bay Stowage*) is shown in the following Tables. Each Table refers to one visited port and it reports the load distribution among the parts when the ship leaves that port.

Note that there isn't any Table which refers to port 3 because it is the last port and here the ship ends its route.

The fist column of each Table specifies if the container groups are loaded in the considered port or if they have been loaded in one of previous port. These

Group		Quantity
Dep.	Dest	(TEU)
0	1	107
0	2	934
0	3	134
1	2	1000
1	3	95
2	3	977

Table 5.3: Input data of the sample case analyzed

two situations are identified respectively by CL ”Currently Loaded” and PL ”Previously Loaded”. The last row gives the total weight allocated in each part of the vessel.

	Group		Part			
	Dep.	Dest	0	1	2	3
CL	0	1	29	26	26	26
	0	2	233	235	233	233
	0	3	33	33	35	33
Tot.			295	292	294	292

Table 5.4: Stowage plan at Port 0 obtained performing the *Pre Stowage Phase* and the *Bay Stowage Phase*

Considering that each crane spends a unit time ”ut” for each load and unload operation, the total time required at the port 0 for the load operation results in 295 ut. Moreover, observing the last row of the Table 5.4 it’s easy to see that the load distribution among the parts assures the vessel’s stability.

Let’s consider the Port 1, Table 5.4. The total time required for load operations at this port is 302 ut. This time is performed by both cranes assigned to the ship parts 0 and 1.

Now considering the last row, observe that there are two colored cells which highlight a critical weight distribution.

Comparing the load distribution it is easy to recognize that the weight allocated in the bow and in the stern of the vessel is less than the weight allocated in the middle parts. That compromises the ship’s balance causing the hull to have

	Group		Part			
	Dep.	Dest	0	1	2	3
PL	0	2	233	235	233	233
	0	3	33	33	35	33
CL	1	2	250	250	250	250
	1	3	23	26	23	23
Tot.			539	544	541	539

Table 5.5: Stowage plan at Port 1 obtained performing the *Pre Stowage Phase* and the *Bay Stowage Phase*

the tendency to bend downward in the middle. This kind of ship stress is called ”sagging”.

Observe that his unbalanced load distribution affects the feasibility of this solution[1].

	Group		Part			
	Dep.	Dest	0	1	2	3
PL	0	3	33	33	35	33
	1	3	23	26	23	23
CL	2	3	245	244	244	244
Tot.			301	303	302	300

Table 5.6: Stowage plan at Port 2 obtained performing the *Pre Stowage Phase* and the *Bay Stowage Phase*

Now considering the situation at Port 2, Table 5.4, observe that the total time required for the load operations is 729 ut. Moreover observing the load distribution in the last row it's easy to see that after unloading the container groups 0-2 and 1-2 and loading groups 2-3, ship's stability is restored.

Given that the unloading time at the last port 3 is 303 ut, the total berthing time obtained using the presently described load plan is 1629 ut.

Furthermore observe that due to the violation of the stability constraint underlined in Table 5.4, this solution is not feasible.

At this stage the algorithm performs the *Local Search Procedure* in order to

[1]Note that as previously mentioned the stability constraint used in this test is deliberately tight and it may not reflect the real stability conditions.

find the feasible solution which worsens the berthing time as little as possible.

At first a *Bay exchange* is performed but it does not result in any feasible solution. Then the *Part exchange* procedure is executed which returns some feasible solutions. Among them the feasible solution that gives the minus berthing time is obtained exchanging the containers of the group 0-2 loaded into part 1 with the containers belonging to the same group loaded into part 3.

This new solution is described in the following Tables.

	Group		Part			
	Dep.	Dest	0	1	2	3
CL	0	1	29	26	26	26
	0	2	233	233	233	235
	0	3	33	33	35	33
Tot.			295	292	294	294

Table 5.7: Stowage plan at Port 0 obtained performing the *Local search*

Let's observe the new solution in Table 5.7. At Port 0 the stability of the ship continues to be uncompromised by the containers' displacement. Moreover the time required for the load operations still remains equal.

	Group		Part			
	Dep.	Dest	0	1	2	3
PL	0	2	233	233	233	235
	0	3	33	33	35	33
CL	1	2	250	250	250	250
	1	3	23	26	23	23
Tot.			539	542	541	541

Table 5.8: Stowage plan at Port 1 obtained performing the *Local search*

Now considering the situation at port 1, Table 5.4, it is easy to note that the new solution solves ”sagging” stress and restores the vessel's balance. As in the previous port, here too the time of unload and load is not affected by the new stowage plan.

Since the container groups exchanged are both destined for the port 2, it is easy to infer that the allocation plan at this port, see Table 5.4, does not suffer any

	Group		Part			
	Dep.	Dest	0	1	2	3
PL	0	3	33	33	35	33
	1	3	23	26	23	23
CL	2	3	245	244	244	244
Tot.			301	303	302	300

Table 5.9: Stowage plan at Port 2 obtained performing the *Local search*

modification. Furthermore given that the berthing times at port 0 and 1 have not been changed either, the total berthing of the ship remains the same.
In appendix A is given a graphical explanation of the resolutive procedure here described.

Chapter 6

Conclusions

This final chapter, provides some thoughts on what has been and what remains to be accomplished in the areas we have studied.

Concluding remarks

As have been demonstrated, the stowage plan of containers on a ship appreciably affects the berthing time and then influences the transport costs supported by the ship company.

This thesis deals with the Ship Stowage Planning Problem with the aim to define a load plan of containers on the ship which allows to reduce the cargo handling time at visited ports as more as possible.

In particular, our purpose is to minimize the working times of quay cranes balancing the load distribution on the ship.

Because of the high complexity of the Ship Stowage Planning Problem we have adopted a decomposition approach which consists of dividing the problems into two subproblems. The first assigns the containers to the bay of the ship, while the second determines their exact position on the vessel in terms of hold, row and tier. Observe that these two problems will be solved interactively through a continued data exchange. In this study we deal only with the first of these sub-

problems. Precisely, we investigate the role that the load stowage plan plays in both the transport's costs and the vessel's stability conditions. Therefore, we develop a solution method which assigns the containers to the bays in such a way that the load's handling time is minimized and the ship's stability is not compromised.
In particular, we decompose the solution process into three main phases: the *Pre Stowage Phase*, the *Bay Stowage Phase* and the *Local Search Procedure*. For each of them we introduce the relative 0/1 model and the heuristic procedure.

The main idea of the fist phase derives from the Quay Crane Scheduling Problem (QCSP). In this part, we focus only on the minimizing of the load's handling time spent at ports relaxing the ship's stability constraint. We partition the ship into a defined number of parts such that each one corresponds to one crane. We then distribute the containers among the parts equalizing the working load of the cranes at each visited port as much as possible. At the end of this phase, we obtain for each part, the list of containers that have to be loaded onto it.

In the second phase, the stability constraints are still relaxed and it takes some aspects from the Knapsack Problem.
Considering each part and the corresponding container list, we allocate the containers among the relative bays taking care not to compromise the vessel's stability. The solution generated consists of containers lists to load onto each bay of the vessel.

In the last phase, we check the feasibility of the obtained solution in terms of ship's stability, in particular for the simplification acquired, we consider only the horizontal component of the vessel's equilibrium. When the defined load plan of the containers among the bays does not satisfy this constraint, we execute a local search procedure which performs some exchanges between containers' locations with the aim to restore the ship's stability.

Finally, we conduct a series of experiments to test the correctness and efficency of our algorithm. The obtained results show that there is the potential for the suggested algorithm to become a suitable approach for solving various kinds of problems. In particular we can see that the proposed approach achieves good quality results in terms of both the solution's quality and the computational time in the performance of the first two phases. Nevertheless, one can note that there are some ”unsolved issue” to clarify, one of which is the high time complexity required to perform the *Local Search Procedure*.

More research needs to be done on how to improve our solution approach and make it practical and effective for real cases. The following Section presents the direction that we should follow in future research.

Future research

Although the proposed algorithm has the potential to become a good solution for the container stowing problem, it remains not practical for large cases and some questions still remain open.

As previously mentioned, the most important improvement which should be introduced to better the applicability of our method, is the reduction of the computational time required by the *Local Search Procedure*. Besides this point, we already discussed the importance of the neighborhood size in the quality of the local search solution. Furthermore we pointed out in what way the assumed stability constraints affect the definition of the investigated neighborhood .

As already explained in the presented algorithm, we consider only the horizontal component of the ship's equilibrium, but the ship's instability may be caused by different factors. For these reasons many other parameters should be considered . For example, if we know the stress the ship is subject to, we can know which parts should be lightened and which should be weighted down. In such cases we could specify precisely which *Bay exchange* or *Part exchange* must be performed to restore the ship's stability. In this way, the neighborhood dimension will be reduced because the algorithm does not try all the possible exchanges. In addition it will be more effective because the containers' displacements will be more controlled and focused.

To make the research practical, we do not considered all the aspects and characteristics of cargo handling operations In this study but more features still remain to be evaluated. For example, we suppose that all the containers have the same standard size, but in the real conditions there may be different sizes. In these cases we should consider additional constraints. One of them is to consider that a 40 foot container can be loaded on two 20 feet containers but not vice versa. Furthermore in our solution, we suppose that all the containers are equal and they have the same dimension ordinary ones. Observe that in a real case this is not possible, because generally there may different types of cargo which require particular handling. For example, there can be special cargo (reefer containers, dangerous containers) which must be positioned only in specific areas.

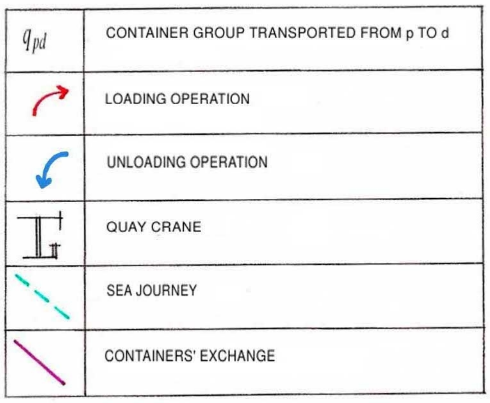
We should also take into account the constraints and the characteristics of the maritime terminals and the handling equipment. In defining the stowage plan we should consider the so-called ”non-crossing” constraint which claims that crane arms cannot be crossed over each other simultaneously. To assure that, we must determine the sequences of lading and unloading operations that have to be performed by each quay crane.

Furthermore one also has to consider the constraints and characteristics of the terminal structure and handling equipment.

In fact, when the container ship arrives at the port, it is assigned to a berth equipped with cranes to load and unload containers. Unloaded containers are moved to a dedicated area in the yard, while containers arriving by road or railway at the terminal are handled by the internal equipment and distributed to the respective stocks in the yard. For these reasons, it’s important to define a ship’s load plan which also takes into account the particular requirements of the quay cranes and of the handling equipments. Thus a further direction of our future research activity should involve the the role played by the container stowage plan on the whole yard system.

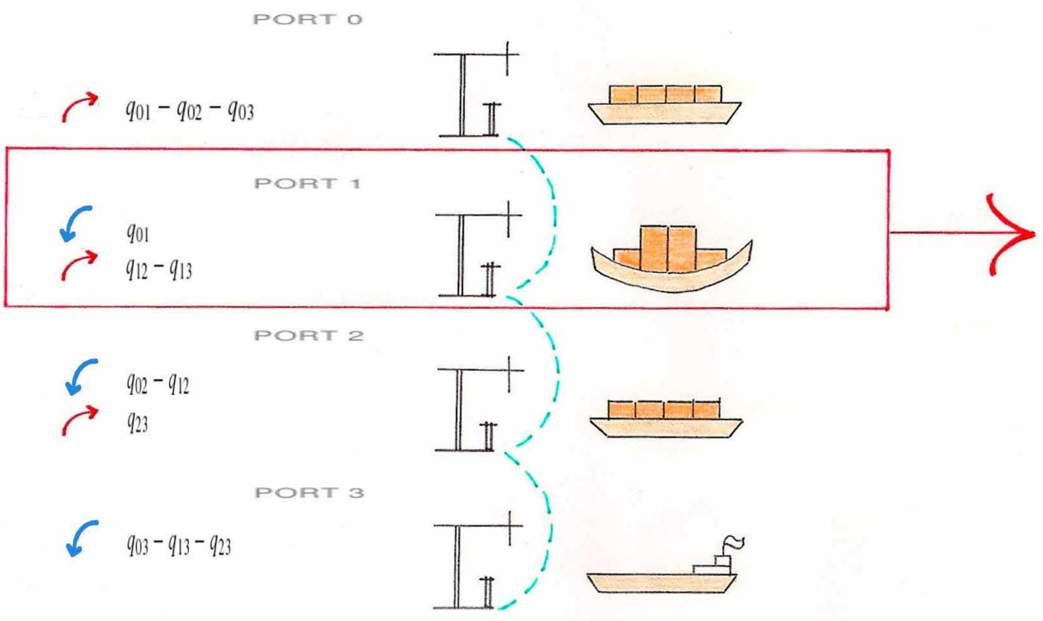
Appendix A

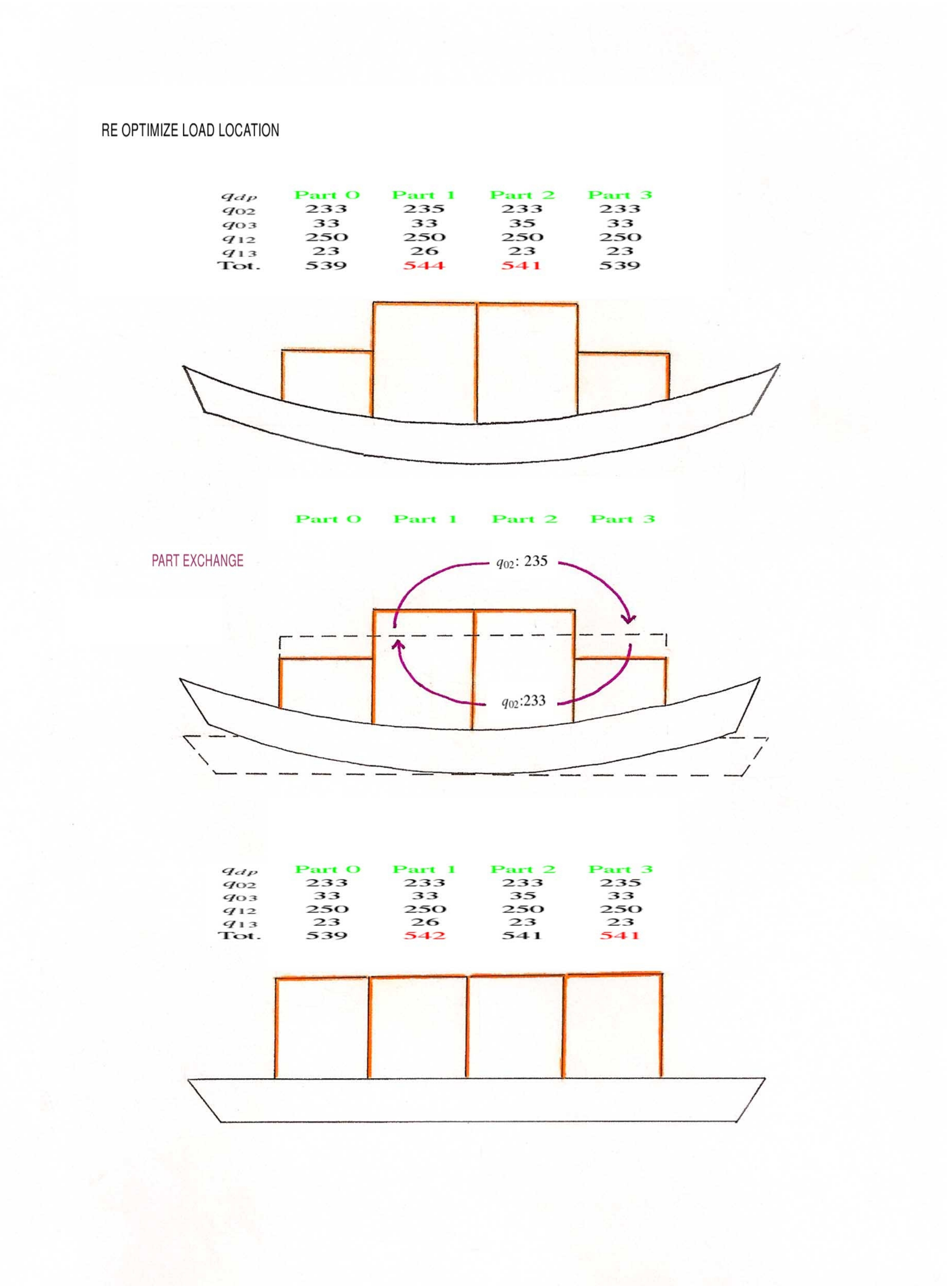
A simple case study



q_{pd}	CONTAINER GROUP TRANSPORTED FROM p TO d
↷	LOADING OPERATION
↶	UNLOADING OPERATION
⊥	QUAY CRANE
╲	SEA JOURNEY
╲	CONTAINERS' EXCHANGE

UNFEASIBLE INITIAL SOLUTION





q_{dp}	Part 0	Part 1	Part 2	Part 3
q_{02}	233	235	233	233
q_{03}	33	33	35	33
q_{12}	250	250	250	250
q_{13}	23	26	23	23
Tot.	539	544	541	539

q_{dp}	Part 0	Part 1	Part 2	Part 3
q_{02}	233	233	233	235
q_{03}	33	33	35	33
q_{12}	250	250	250	250
q_{13}	23	26	23	23
Tot.	539	542	541	541

Figure A.1: A graphical representation of the *Local Search Procedure* described in Section 5.4.

Bibliography

[Amb06] D. Ambrosino, A. Sciomachen, E. Tanfani, *A decomposition heuristics for the container ship stowage problem* ,2006, *Journal of Heuristics* Volume 12 , Issue 3 (May 2006) 211-233.

[Amb04] D. Ambrosino, A. Sciomachen, E. Tanfani, *Stowing a containership: the master bay plan problem* ,2004, *Transportation Research Part A* 38 (2004) 81-99.

[Arm04] Armstrong, John, *Small Ships Training and Operational Manual*,2004, *Journal of Heuristics*3rd ed. Queensland Government.

[Avr98] Avriel, M., M. Penn, N. Shpirer, and S. Witteboon., *Stowage Planning for Container Ships to Reduce the Number of Shifts*,1998, *Annals of Operations Research* 76: 55-71.

[Avr93] Avriel M. and Penn M., *Exact and approximate solutions of the container ship stowage problem*,1993, *Comput Indust Engng* 25: 271-274.

[Bot92] R.C. Botter and M.A. Brinati, *Stowage container planning: a model for getting an optimal solution*,1992, *IFIP Trans.* B-5: 217-229.

[Dub02] Opher Dubrovsky, Gregory Levitin, Michal Penn, *A Genetic Algorithm with a Compact Solution Encoding for the Container Ship Stowage Problem*,2002, *Journal of Heuristics* 8(6): 585-599.

[Hif05] M.Hifi, M. Michrafy, A. Sbihi, *A Reachtive Local Search-Based Algorithm for the Multiple Xhoice Multi-Dimensional Knapsack Problem*,2005, *Computational Optimization and Applications* .

[Kan02] J-G Kang and Y-D Kim, *Stowage planning in maritime container transportation*,2002, *Journal of the Operational Research Society* 53:415-426.

[Kia99] Kia M., Shayan E., Ghotb F.*The Importance of Information Technology in Port Terminal Operations*,1999, *International Journal of Physical Distribution & Logistics Management* 30, 3/4, 331-344.

[Mar88] Martin, JR., SU. Randhawa, and ED.McDowell *Computerized Containership Load Planning: A Methodology and Evaluation*,1988, *Computers and Industrial Engineering* 14: 429-440.

[Pal99] Pallottino S., Sciomachen A., *Scienze delle decisioni per i trasporti*,1999, *Franco Angeli.*

[Pet90] Peterkofsky, Roy I. and Daganzo, Carlos F., *A branch and bound solution method for the crane scheduling problem*,1990, *Transportation Research Part B: Methodological* 24(3):159-172.

[Sch99] Andrea Schaerf, Maurizio Lenzerini, Marco Cadoli, *Local++: A C++ Framework for Local Search Algorithms*,1999, *Technical Report* 11-99, May 1999.Dipartimento di Informatica e Sistemistica, Universita di Roma La Sapienza, ITALY.

[Sch84] Schields JJ, *Container ship stowage: a computer-aided preplanning system*,1984, *Marine Technol* 21: 217-229.

[Sci03] Sciomachen, A., and E. Tanfani, *A Solution Method Based on its Connection to the Three Dimensional bin Packing Problem*,2003, *IMA Journal of Management Mathematics* 14(3), 251-269.

[Shi08] Shigeyuki Takahara *A Multi-start Local Search Approach to the Multiple Container Loading Problem*,2008, *Advances in Greedy Algorithms* pp: 586.

[Sta07] Robert Stahlbock, Stefan Voss *Operations research at container terminals:a literature update*,2007, *OR Spectrum* (2008) 30:1-52.

[Ste04] D. Steenken, S. Voss, R. Stahlbock., *Container terminal operation and operations research-a classification and litterature review*,2004, *OR Spectrum* 26: 3-49.

[Ste00] Steenken D., Winter Th., Zimmermann U.T., *Stowage and transport optimization in ship planning*,2000, *Online optimization of large-scale systems* 731-745.

[Vas08] A. Vasilis Vasiliauskas, J. Barysiené, *An Economy Evaluation model of Logistic System based on Container Transportation*, 2008, *Transport* 22(4) 311-315.

[Wil99] I.D. Wilson and P.A. Roach, *Principles of Combinatorial Optimization Applied to Container-Ship Stowage Planning*,1999, *Journal of Heuristics* Volume 5 , Issue 4 (December 1999): 403 - 418.

[Zhu04] Yi Zhu, Andrew Lim *Crane Scheduling with Spatial Constraints: Mathematical Model and Solving Approaches*,2004, *Naval Research Logistics* 51: 386-406.