IT University of Copenhagen

Ph.D. Thesis

Fast Generation of Container Vessel Stowage Plans

using mixed integer programming for optimal master planning and constraint based local search for slot planning

Author:
Dario Pacino

Supervisor:
Assoc.Prof. Rune Møller Jensen

June 12, 2012

Preface

During my undergraduate studies, I have always been focused on becoming a good software designer and programmer. During my master studies at the IT-University of Copenhagen I met Rune Møller Jensen who proposed to me to do a project within the shipping domain. Without even realizing it I was thrown into the abyss of research, which changed by carrier plans drastically. I wanted to be a researcher. I was very thankful when the Dansk Maritim Fond granted me a full Ph.D. scholarship at the IT-University of Copenhagen with Rune Møller Jensen as my supervisor.

The road to the successful conclusion of this Ph.D. project was not smooth. Very little was known about the details of container stowage. There is no book telling the whole story. In some way I hope this Ph.D. thesis can cover this knowledge gap for future researchers. The years of industrial collaboration and the hard work of everyone involved in this project has contributed to its success.

During my Ph.D. studies I also have had the pleasure of visiting Professor Pascal Van Hentenryck at Brown University for a period of 5 months. During the visit I worked on decomposition techniques for scheduling problems. The results of this project are published in peer-reviewed venues (Pacino and Hentenryck, 2011; Pacino and Van Hentenryck, 2011). I also have had the pleasure to work with my colleagues from the Decision Optimization Lab, Alberto Delgado and Kevin Tierney, and with Yuri Malitsky on a machine assignment problem for a competition organized by EURO (European Conference on Operations Research), for which we have qualified. These collaborations have undoubtedly contributed to broadening my research prospectives and finally to the completion of my Ph.D. study.

Abstract

Containerization has changed the way the world perceives shipping. It is now possible to establish complex international supply chains that have minimal shipping costs. Over the past two decades, the demand for cost efficient containerized transportation has seen a continuous increase. In order to answer to this demand, shipping companies have deployed bigger container vessels, that nowadays can transport up to 18,000 containers and are wider than the extended Panama canal. Like busses, container vessels sail from port to port through a fixed route loading and discharging thousands of containers. Before the vessels arrives at port, it is the job of a stowage coordinator to devise a stowage plan. A stowage plan is a document describing where each container should be loaded in the vessel once terminal operations commence. When creating stowage plans, stowage coordinators must make sure that the vessel is stable and seaworthy, and at the same time arrange the cargo such that the time at port is minimized. Moreover, stowage coordinators only have a limited amount of time to produce the plan. This thesis addresses the question of whether it is possible to automatically generate stowage plans to be used by stowage coordinators, and it advocates that the quality of the stowage plans and the time in which they can be generated is of the outmost importance for practical usage. We introduce a detailed description of a representative problem of the computational complexity of stowage planning that has enough detail to allow professionals from the industry to evaluate its solutions. A 2-phase hierarchical decomposition of the problem is presented. In the first phase, the problem of distributing containers to sections of the vessel is solved, and it is here that the seaworthiness of the solution is evaluated. In the second phase, the assignment of containers is refined to specific positions within the ship and lower level constraints are handled. The approach has been implemented with a combination of operations research and artificial intelligence methods, and has produced promising results on real test instances provided by a major liner shipping company. Improvements to the modeling of vessel stability and an analysis of its accuracy together with an analysis of the computational complexity of the container stowage problem are also included in the thesis, resulting in an overall in-depth analysis of the problem.

Contents

Chapter 1

Introduction

The world economy has always been heavily dependent on the ability of transporting goods. In the years up to the 50s companies were operating mainly in local markets where the competitive advantage depended on how close to the customers products were manufactured. After the year 1955, with the introduction of containerized shipping, all this changed. Containers made it possible to transport products across the oceans with minimal cost and complexity. The world's economy changed to a global economy, where Asian manufacturing companies could sell their products to the rest of the world at minimal costs, and where supply chains grew larger and more international (Levinson, 2006).

Liner shipping companies faced this increasing demand by building larger ships that nowadays can carry up to 18,000 containers. We also see the creation of specialized harbors that only handle container shipping, easing intermodal transportation and supporting this new global economy.

Employing larger ships, however, is not enough for shipping companies to respond to the demand for cheap transportation. The load of containers on a vessel must be optimized so costs can be reduced. The distribution of containers defines how the vessel sits in the water, and thus influences its bunker consumption. Moreover, load and discharge operations at container terminal are costly, thus reducing the number of moves and the total time at port is essential to achieving cost reductions.

It is the job of *stowage coordinators* to ensure that the containers are correctly stowed in the vessel in a cost efficient way. Stowage coordinators manually produce a so called *stowage plan* [1]. Stowage plans define the exact position of each container within a vessel and are used by container terminals during load and discharge operations. Stowage planning is a central component in the liner shipping industry and can be used in different contexts. For example, it can be used by string managers for the design of cargo flows in a shipping network, where stowage plans define the possible container intakes of the vessels given a specific mixture of cargo. In container terminals, sequencing operations (the decision of which container to be taken from the yard at

[1] Stowage plans, containership stowage plans and container stowage plans will be used interchangeably in this thesis.

which point in time) are strongly affected by each vessel stowage plan. Integration of stowage planning into terminal sequencing would improve terminal operations even more.

Cost-efficient stowage plans, however, are hard to produce in practice. First, they are made under time pressure just hours before the vessel calls the port. Second, large deep-sea vessels often require thousands of container moves in a port. Third, complex interactions between low-level stacking rules and high-level stress limits and stability requirements make it difficult to minimize time at port. Fourth, robust plans require stowage coordinators to evaluate the consequences of decisions in down-stream ports. Information about containers in further ports is, however, scarce and based on forecasting so several scenarios might need to be generated before a final plan can be released.

From a research point of view, it is also hard to access information about stowage planning. Detailed information about vessel designs is necessary in order to produce correct stability and stress calculations. This data is considered confidential and thus very hard to access without a close collaboration with the industry. Expert knowledge about stowage planning itself is also very important, but no written book about stowage planning exists. Knowledge must be extracted from various guides, companies' internal documentations and the experience of senior staff.

These barriers did not stop academics from growing an interest in this problem, and the past decades have seen a growing number of works published in the literature. Previous work on the stowage planning problem has been innovative on solution approaches, but has been challenged by the inaccessibility of the problem. This has lead to limited representative power of the developed models and a lack of benchmark problems.

1.1 Thesis Question

This thesis is based on a long term collaboration with a large liner shipping company, and thus has the opportunity to breach the knowledge and data barrier and deliver highly representative research findings. This thesis aims at answering a question that, according to the results shown in the literature has yet to receive a positive answer:

> *"Can containership stowage plans with high enough quality for practical usage be computed on standard equipment within the time required by the work processes of stowage coordinators?"*

In order to fully understand this question, it is important to clarify some of the key aspects the question is built upon. The first of those aspects is "high enough quality." Like many other industrial problems, stowage planning is riddled with details, special cases and solution preferences. This thesis does not aim at generating stowage planning algorithms that can substitute human stowage coordinators. We believe that both the research field and the current technology is not ready for such a challenge. This thesis aims at producing stowage plans that can be used by human stowage coordinators as guidance in the decision making process of generating a final stowage plan. More specifically, automated stowage planning should model all key aspects of stowage planning such that an automatically generated stowage plan can be used as a reference solution that can easily be modified into a final plan.

The second important aspect of the thesis question is "computed on standard equipment." Shipping companies have teams of stowage coordinators and it would be unreasonable to believe that each coordinator is equipped with a super computer. We aim at identifying the possibility of answering the thesis question using hardware that is affordable, but still has high quality properties. The work presented in this thesis is based on hardware corresponding to 8-12 AMD Opteron processors on a single machine.

The last, and most restrictive, aspect or requirement of the thesis question is "within the time required by the work processes of stowage coordinators." Stowage coordinators have a limited number of hours to produce the final stowage plan, however, one must consider that the stochastic nature of the forecasts used during stowage planning forces coordinators to consider different scenarios. Should an automated stowage planning tool take too long to produce plans, it would not be possible for the stowage coordinator to consider all the necessary scenarios. Moreover, it is often the case that in problems as complex as stowage planning, software only solves an abstraction of the problem and thus stowage coordinators must also spend time on final adjustments. In accordance with our industrial collaborators it was estimated that 10 minutes is the maximum amount of time that a piece of software should take to deliver an answer, if it is to be used during the everyday planning process.

1.1.1 Approach

This thesis attempts at answering the thesis question in a two phase research process. Initially we have analysed the currently available software packages and documentation to collect the necessary knowledge. As questions arose, the expert knowledge of stowage coordinators was used to fill-in the gaps. With all the collected information it was possible, with the agreement of our industrial collaborators, to define a set of stowage planning aspects that any software package should have in order to be considered representative.

With a representative problem at hand, the second phase of the research project focuses on the design of efficient algorithms for stowage planning. The 10 minutes required by the industry is a relatively short time to solve a complex and large scale problem such as stowage planning. The literature have shown that monolithic approaches tend to fail even on small instances (e.g. Ambrosino and Sciomachen (2003); Giemsch and Jellinghaus (2004); Li et al. (2008)), while multi-phase approaches have shown promising results (e.g. Wilson and Roach (1999); Kang and Kim (2002); Ambrosino et al. (2010)). This thesis, inspired by those last works, presents a 2-phase approach to stowage planning that is able to reach phase-wise optimal solutions within the required time. An initial phase distributes containers into stowage areas along the vessel while satisfying stability and stress forces and optimizing time at port. A second phase then assigns specific positions to containers within each of the stowage areas while satisfying low-level constraints such as stacking rules and robustness requirements. The positive results obtained also motivated an extension of the current model to include more accurate stability and stress calculations.

1.2 Thesis Contributions

The thesis has four major contributions:

1. **Accurate models**
   The models presented in this thesis are the most accurate models published in the literature. The master planning models include key aspects about stress forces and vessel stability that have not yet been presented in the literature. Solution objectives that guide the search toward robust stowage plans are also a novel part of this thesis work.

2. **Optimal master planning**
   This thesis presents the first optimal master plans.

3. **Fast approach**
   The models and algorithms presented in this thesis are the fastest among those taking into account similar stowage planning aspects.

4. **Many instances**
   This thesis presents the first results that are based on a large set of industial benchmark instances. The instances are generated from stowage problems that have sailed and thus are highly representative.

It should be noted that only three Ph.D. thesis have been published on the container stowage problem: Aslidis (1989), Wilson (1997), and Kaisar (1999). This last has, however, no related publications.

1.2.1 Contributions in Constraint Based Scheduling

As part of the collaboration with Professor Pascal Van Hentenryck, a number of contributions have been made in the field of constraint based scheduling:

1. **First Decoupling Analysis of Flexible Jobshop Scheduling**
   To the best of our knowledge this thesis presents the first application of decoupling techniques to scheduling problems

2. **Improved Best Solutions on Standard Benchmarks**
   The algorithms implemented in this thesis have brought many new best solutions to standard flexible jobshop scheduling benchmarks.

1.3 Publications

- Pacino, D. and R. M. Jensen (2012).**Constraint-based local search for container stowage slot planning**. In Proceedings of the International MultiConference of Engineers and Computer Scientists, pp. 1467-1472.
- Pacino, D., A. Delgado, R. Jensen, and T. Bebbington (2011). **Fast generation of near-optimal plans for eco-efficient stowage of large container vessels**. Computational Logistics, Volume 6971 of Lecture Notes in Computer Science, pp. 286-301.
- Pacino, D. and R. M. Jensen (2010). **A 3-phase randomized constraint based local search algorithm for stowing under deck locations of container vessel bays**. Technical Report TR-2010-123, IT-University of Copenhagen.
- Pacino, D., A. Delgado, R. Jensen, and T. Bebbington (2012b). **Large-scale container vessel stowage planning using mixed integer and constraint programming**. Under review at Transportation Science.
- Pacino, D., A. Delgado, R. Jensen, and T. Bebbington (2012a). **An accurate model for seaworthy stowage planning with ballast tank**. Under review for the 3nd International Conference on Computational Logistics, (ICCL12).
- Pacino, D., K. Tierney, and R. M. Jensen (2012). **NP-hard components of container vessel stowage planning**. To be submitted to OR Letters.
- Pacino, D. and R. M. Jensen (2009). **A local search extended placement heuristic for stowing under deck bays of container vessels**. Extended Abstract in The 4th Int. Workshop on Freight Transportation and Logistics (ODYSSEUS-09).
- Pacino, D., A. Delgado, and R. Jensen (2012). **Modeling ballast water in container stowage planning**. Accepted for presentation at the 25th European Conference on Operational Research (EURO 2012), 8-11 July 2012, Vilnius (LT).

1.4 Document Outline

Chapter 2 Container Stowage Planning. This chapter introduces the reader to the domain of liner shipping providing the necessary knowledge and terminology for a full understanding of the container stowage problem. This chapter is based on both the knowledge accumulated during our collaboration with the industry and the book on naval architectures by Tupper (2009).

Chapter 3 Related Work. This chapter gives, to the best of the our knowledge, a complete review of all the academic work on the container stowage problem.

Chapter 4 Representative Problem and Algorithmic Framework This chapter presents the representative problem that is solved as part of the work in this thesis. Remarks about the abstractions and assumptions are also presented. In this chapter we also introduce the 2-phase solution framework, called Quad, used to solve the container stowage problem.

Chapter 5 Master Planning. This chapter describes the first phase of the Quad approach to container stowage planning. In particular it introduces a Mixed Integer Programming (MIP) model that can solve the master planning phase and evidence of its efficiency is presented. Later in the chapter, a refinement of the model that focuses on achieving high precision for the stability calculations, is also presented.

This chapter is based on the papers of Pacino et al. (2011), Pacino et al. (2012b), and Pacino et al. (2012a). Note that the defendant has made significant contributions to these papers and the implementation methods.

Chapter 6 Slot Planning. This chapter presents a detailed description of the CBLS algorithm used to solve the slot planning problem in the second phase of the Quad framework. The neighborhood, heuristics, and incremental calculations used in the algorithm are described in details together with the experimental results that show evidence of the solution method's efficiency.

This chapter is based on the papers by Pacino and Jensen (2009), Pacino and Jensen (2010) and Pacino and Jensen (2012). The defendant is the responsible for the implementation methods and is the main contributor to the papers.

Chapter 7 Conclusions. This chapter ends the thesis with the presentation of concluding remarks and giving an outlook of future work and possible research directions.

Appendix A Complexity Analysis This appendix introduces previous work on the computational complexity of the container stowage problem, and presents new results. In particular we prove that the Capacitated k-Shifts Problem for a fixed ship size is in P, and prove that the Hatch Overstow Problem is NP-Complete.

This chapter is the extended and revised version of the paper by Pacino et al. (2012), where the defendant made significant contributions.

Appendix B Decomposition in Scheduling. This appendix describes the work done in collaboration with Professor Pascal Van Hentenryck. It presents the application of decomposition methods to the flexible jobshop problem. Particular focus is given to the Adaptive Randomized Decomposition framework that, for the first time, is being applied to scheduling problems.

This chapter is based on the paper by Pacino and Hentenryck (2011). The defendant is the responsable for the implementation methods and is the main contributor to the papers.

Chapter 2

Container Stowage Planning

Seaborne shipping is nowadays the most used shipping mode. In general there are three main seaborne shipping modes: *industrial*, *tramp* and *liner*. In industrial shipping it is the owner of the cargo that owns the fleet and focuses on the minimization of cargo transports costs. Tramp ships operated like taxis, transporting available cargo to destinations. It is often the case that such shipping companies have contractual commitments with cargo companies. In liner shipping, vessels are assigned a route (or *string*) and follow a specific schedule. Liner shipping is the preferred mode for containerized transportation and it is the assumed transportation mode in the remainder of this thesis.

Containerization is the transportation of cargo into metallic boxes called *containers*. Containers are transported by *container vessels*, which are ships specially designed for the transportation of containers. Such vessels can carry thousands of containers with a small crew. In general we will talk about 20' and 40' long containers and refer to the space occupied by a 20' container as a Twenty-foot Equivalent Unit (TEU). This mean that a 40' container is equivalent to two TEUs.

2.1 Container Terminals

Container terminals are specialized harbors for the handling of containers. When a container vessel arrives at the terminal, it is assigned a *berth* where load and discharge operations will be performed. Terminals are divided into three main areas, the *quayside*, where the berths and the vessels are, the *yard* which is a temporary storage area for containers, and the *landside* where truck and train operations are performed. Import containers are unloaded from the vessels and transported to either the yard, in case of transshipments, or directly to the landside for inland transportation. The opposite cargo flow applies for export containers. Container movements between the quayside, the yard and the landside can be performed by trucks with trailers, multitrailers, Autonomous Guided Vehicles (AGVs) or straddle carriers. Containers in the yard are stored in long rows of container stacks accessible by *gantry-cranes*. Some stacks are

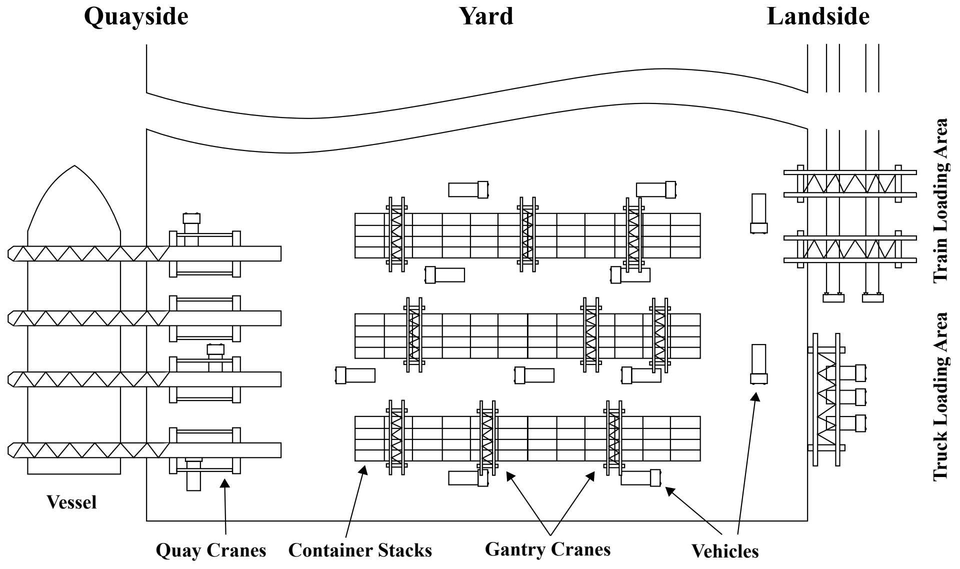


Figure 2.1: Container terminal layout

reserved for containers with special requirements, such as containers requiring electrical power or containers carrying dangerous goods. Figure 2.1 shows an example of a terminal layout.

Berths, in the quayside, are equipped with *quay cranes* that can lift containers to and from a container vessel. Most of today's quay cranes can *twin-lift*, meaning that they can lift two 20' containers or one 40' container in one single lift. Quay cranes have an average operational performance of 30-37 lifts per hour. Notice, however, that a twin-lift takes a little longer time (about 25 lifts per hour) than a single lift of a 40' container (about 32 lifts per hour). The lifting operation for hatch-covers (metallic structures dividing the ship into on and below-deck parts) takes a longer time than that of a container since the *spreader* (the hooking equipment of a crane) is harder to align on a hatch than it is on a container. Most quay cranes run on rails that are parallel to the vessel, and thus their movements are restricted by the adjacent cranes. Some cranes have wheels and can move independently from each other. Quay cranes must also respect security distances, and, depending on the operation to be performed, they must be separated by 40, 60 or even 80 feet. A *crane set* is the operation of moving a crane from its current position to another. It can take up to 2 minutes to move a crane to an adjacent storage area along the vessel. Should a crane be moved across the pilot house, it would be necessary to lift the crane arm. Such an operation can take up to 25 minutes.

Given a vessel with a distribution of moves on its storage areas, a *crane split* is a partitioning of the work areas to each of the cranes assigned to the vessel such that the workload of each partition is close to equal. A good crane split will minimize the makespan of the load and discharge operations and thus minimize the time at port. An

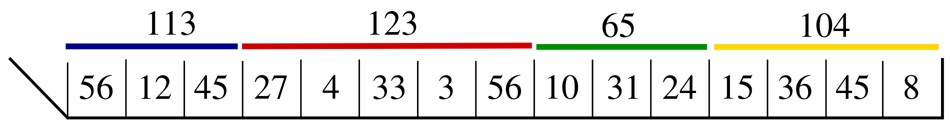


Figure 2.2: Optimal cranesplit with 4 cranes

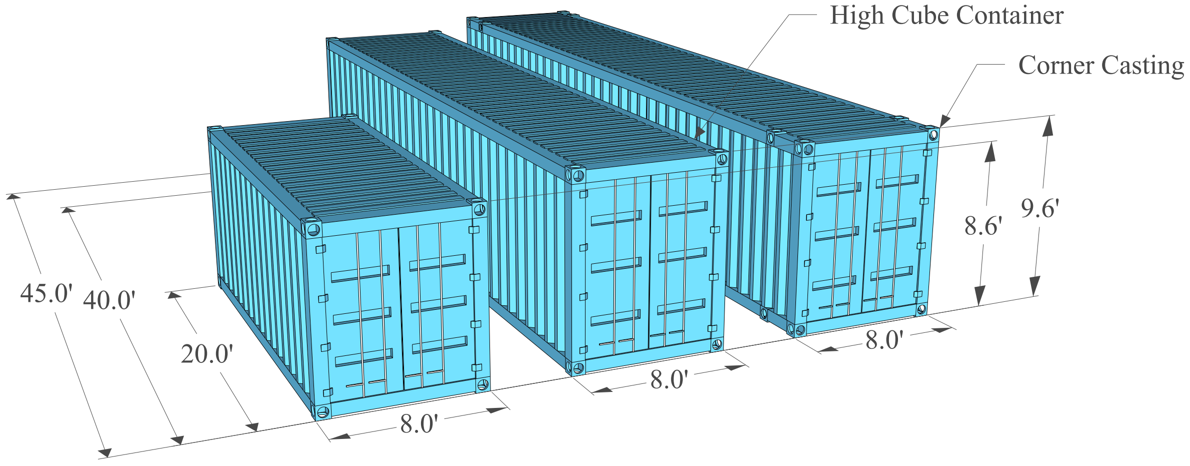


Figure 2.3: Dimensions of most common standard ISO containers

optimal crane split for 4 cranes for a vessel with 15 storage areas is shown in Figure 2.2.

It is important to discuss crane productivity since a vessel not only has to pay a fee for each *move* (load or discharge operation), but also for the team operating the cranes and the cost of the vehicles feeding the cranes. The total move cost is thus linear in the number of cranes utilized. The move cost, however, is often overweighed by the need for reducing the time at port.

2.2 Standard Containers

Containers are metallic boxes designed to withstand significant outer forces. They are particularly robust to high vertical compression, which allows the creation of high stacks. All containers are fitted with corner castings designed to support the container's weight, and to which security fittings can be attached. ISO standard containers are usually 20', 40' or 45' long. In the US trade it is possible to also find 48' or 53' containers that are non-standard in liner shipping.

ISO containers are 8' wide and 8.6' tall with the exception of *high-cube* containers which are 1 foot taller. Figure 2.3 shows an illustration of a 20' container, a 40' container, and a 45' high-cube container. Longer containers, such as the 45' containers are equipped with two extra set of castings at a 40' distance. The extra castings allow longer containers to be stacked on top of 40' containers. No castings, however, exist at the 20' position, which means that 20' containers cannot be stacked on top of longer

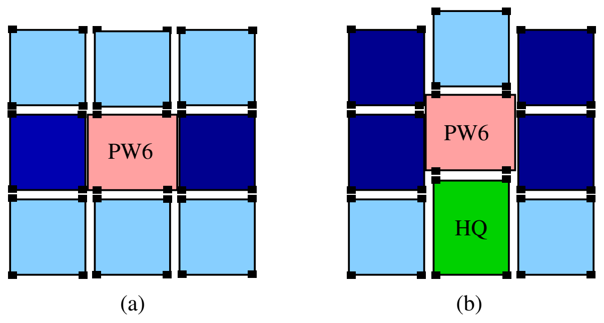


Figure 2.4: Valid palletwide container stowage, (a) without high-cubes (b) with high-cubes

containers.

Note, however, that it is often not allowed on-deck to stack 40' containers on top of longer ones, since it becomes difficult to attach security fitting.

2.2.1 Container Types

Aside from the standard and high-cube containers discussed above, there are a number of specialized containers for different kinds of cargo. Fruits and vegetables, for example, must be transported in refrigerated containers. There are two kinds of refrigerated containers, *reefer*, which have an integrated refrigeration unit and thus must be connected to power supplies, and *insulated* containers, which have more internal capacity than a reefer container, since they do not have an integrated refrigeration unit and thus must be connected to the vessel's internal cooling plant when on board. *Ventilated* containers have small opening allowing air circulation and do not require external power. Fluids, foodstuffs, chemicals and hazardous cargo are often transported in *tank* containers. Some of the cargo in tank containers might need to be kept heated and therefore those containers might also need to be plugged into power supplies. Foodstuffs, cereals and spices are also often transported on *bulk* containers where cargo can be loaded through three hatches on the top. Cargo with non standard dimensions, such as overweight or tall cargo is transported on *Out-Of-Gauge* (OOG) containers. Within this category we find *hard-top* containers with a removable roof, *open-top* containers without roof, *fla-track* containers with only two end-walls that can be collapsed and *platforms* for heavy and oversized cargo.

In order to better fit Euro-pallets, *pallet-wide* containers use the 2 inches (ca. 5 cm) space between container stacks to enlarge their capacity. Based on the stacking rules that those containers follow, they are divided into three categories: *PW0* which do not need any special stacking rules, *PW3* which can be stacked side by side but require a standard container every two PW3s and *PW6* which cannot be stacked side by side. Figure 2.4 shows two valid stacking configurations including a PW6 container. Notice how the stacking rules involve more containers as palletwides are mixed with high-cubes.

Another classification of containers that require special stacking rules are the *IMO* containers. All containers transporting hazardous materials fall under this category. IMO containers are further classified by the numbers 1-4 depending on the level of risk. Containers in the IMO-1 group do not require any rules. A one container separation is required between IMO-2 containers. IMO-3 and IMO-4 require more complex separation patterns (see IMO (2010)). Most vessels have special storage areas for IMO containers, often towards the bow below-deck.

2.3 Container Vessels

The first vessel specifically designed for the transportation of containers was build in 1960 an had a capacity of 610 TEUs (MAN Diesel & Turbo, 2008). Since then, container vessels have been increasing in size and quality. Nowadays the world fleet is composed of about 6,000 ships, transporting over 16 million TEU with the largest container vessel having a capacity of 18,000 TEUs (Alphaliner, 2012). Container vessels are usually classified according to their capacity and size into the following main groups:

Small Feeders with capacity $\leq$ 1000 TEUs are used for short sea transport

Feeders with a capacity of 1000 - 2,800 TEUs are usually applied to feed very large vessels or service markets that are too small for larger vessels

Panamax with a capacity of 2,800 - 5,100 TEUs are vessels that can sail through the existing Panama Canal

Post-Panamax with a capacity of 5,100 - 10,000 TEUs are vessels that exceed the current Panama Canal beam

New Panamax with a capacity of 12,000 - 14,500 TEUs are vessels that exceed the current Panama Canal beam, but that will be able to sail through the new lane and lock chambers that will be fully operative in the canal by 2015

ULCV Ultra Large Container Vessels (ULCVs) have a capacity $\geq$ 14,500 TEUs and are already larger than the new Panama Canal beam (MAN Diesel & Turbo, 2008).

The layout of a container vessel is shown in Figure 2.5. The figure shows how containers are arranged into storage areas called *bays* (or *holds*) through-out the entire vessel length. Each bay is composed of a number of container *stacks*, and since load and discharge operations only happen from the top, it is only the top container in a stack that can be accessed at any point in time. A vertical position in a container stack is called a *cell* and usually has a capacity of two TEUs, meaning that we can either stow two 20' containers or one 40' (or 45') container. Each TEU position within a cell is referred to as a *slot*. Slots toward the bow of the vessel are called *Fore* slots and those towards the stern are called *Aft* slots. In general there is a distinction between

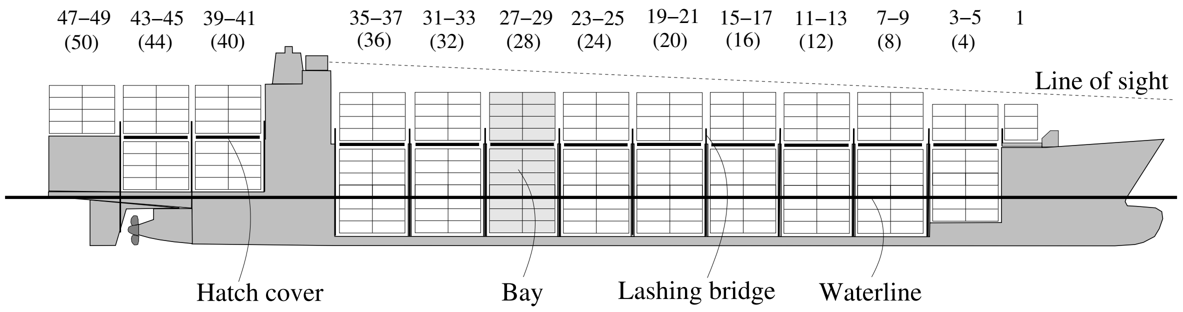


Figure 2.5: The arrangement of bays in a small container vessel.

on-deck and *below-deck* areas of a bay. The below-deck areas are closed by *hatch-covers* (or *hatch-lids*), which are tight metallic structures that prevent water from coming in. Containers below-deck can only be accessed once all containers on top of the hatch-cover and the hatch itself are removed.

Figure 2.6 shows a graphical representation of a bay layout. In the figure it is possible to see how the hatch-cover divides the on-deck and below-deck containers, and also shows the standard indexing system used by the industry, where stacks are enumerated from the middle out and *tiers* (vertical indexes) start from the bottom and distinguish on-deck from below deck by starting at 82 (in most cases) after the hatch-cover. The illustration also shows how, due to high-cubes, containers can be out of sync with the tier indexing. As previously mentioned this can be problematic, for example, in correlation with the stowage of pallet-wide containers.

2.3.1 Vessel Stability and Stress Limits[1]

When a container vessel has to leave a port it must be declared seaworthy, which means that its initial stability must be correct and that all stress forces must be within limits.

When a vessel floats in the water, its immersion depth is called *draft*. More precisely, the draft of a vessel is the distance between the *keel* (the bottom of the vessel) and the *waterline*. The waterline is a line following the level that the water reaches around the hull of the vessel. The plane cutting the vessel at the water level is called a *waterplane*. If the total weight of the vessel and its cargo, the *displacement*, is evenly distributed the keel of the vessel would be parallel to the waterline in both the longitudinal and transversal axis. A vessel is said to be at *even keel* when its weight is evenly distributed around its transversal axis, which is a necessary condition for a vessel to be seaworthy. It is, however, not uncommon for container vessels to have an uneven longitudinal weight distribution (this can also be the case for empty vessels). In this case, since the keel is not parallel to the waterline, the vessel is inclined by a angle called the *trim angle*. Figure 2.7 shows a longitudinal section of a container vessel where W_1 is the waterline parallel to the keel and W_2 is the waterline for a trim angle δ. Container

[1]Inspired by Tupper (2009)

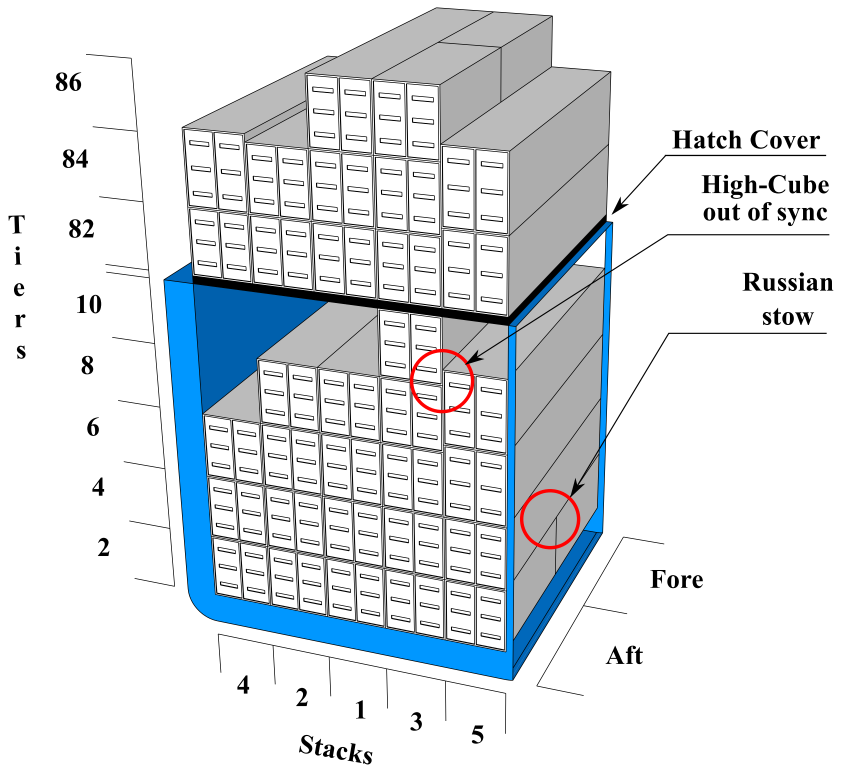


Figure 2.6: The layout of a bay with container on and below-deck

vessels have two perpendiculars to the keel used as reference for different calculations. The perpendicular toward the stern is the *aft perpendicular* (AP) and the one toward the bow is the *fore perpendicular* (FP). Vessels with a non zero trim angle distinguish between the two drafts at the perpendiculars, namely the *aft draft* and the *fore draft*. The difference between these two draft is called *trim*. Each vessel has specific limits within which the trim must be in order for the vessel to be seaworthy. Notice that some vessels prefer non-zero trim values, which make it is possible to achieve better propulsion efficiency. In order to make sure that the propeller is sufficiently immersed, a minimum aft draft is often required. Draft values are not only important during the initial stability calculations but also during the sailing period. The route the vessel sails and the ports and canals it will visit, present restrictions to the maximum draft of the vessel. These requirements must be satisfied in order for the vessel to avoid the risk of running aground.

When a vessel's floating equilibrium is disrupted by an external force, the tendency of the vessel of returning to its original position once this force is lifted is called vessel stability. Vessel initial stability is calculated when the vessel floats in still waters and can be affected by small external forces. Consider Figure 2.8, where a transversal section of a container vessel is shown. The figure shows the center of gravity of the vessel G, and the center of buoyancy B_1. The center of buoyancy is the center of gravity of the

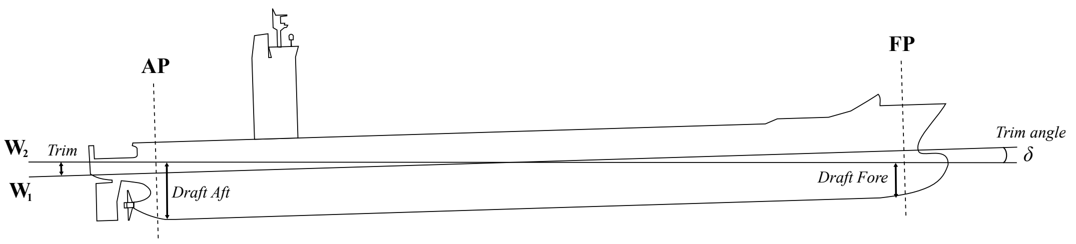


Figure 2.7: Longitudinal section of a vessel showing trim, draft and waterlines

volume of water displaced by the vessel. When an external force is applied, assuming constant displacement and no free weights, it is safe to assume that the rotation will not affect the center of gravity, which thus remains constant. The center of buoyancy, on the other hand, changes as the underwater volume changes its form. The gray wedges between the original waterline W_1 and the waterline after the inclination W_2 represent the movement of the volume of water displaced by the vessel. The center of buoyancy of the vessel thus changes to B_2. For small *heeling* angles (θ), it is safe to assume that the volume of the wedges will grow linearly with θ. The *metacenter* of the vessel is the intersection M between the center line going though G and the center line going through B_2, and for small angles can be considered a fixed point. Once the external force inclining the vessel is removed and equal and opposite force δ will bring the vessel back to its original position. The perpendicular from G to B_1M is called the *righting level* (z), thus the moment with which the vessel rolls back to is original position is $M_r = \delta z = \delta GM \sin\theta$, where GM is called the *metacentric height*. Vessels have a minimum required GM before they can be considered stable. The larger the GM the more stable the vessel is. Too large values might, however, result in a too "stiff" vessel.

The weight distribution on a container vessel does not only affect its stability. The forces at play on a vessel stress its physical structure and must be kept within the vessel's structural limits. The most basic structural stresses are those regarding the weight of the container stacks, which must not exceed the maximum capacities. Nowadays ships are designed to be light in order to reduce bunker consumption. It is thus important to take structural weight limitations into account. Two weight limits exists for each stack, one regarding the outer container supports and one regarding the inner supports. Limits on the inner supports are often the smallest as the vessel structure in the middle of a stack is weaker. The inner supports are used only when 20' containers are stowed. When 20' and 40' containers are mixed in the same stack, only half of the 20' weight is considered to be supported by the outer supports, since the other half sits on the inner supports.

Other structural stresses come from the distribution of the upward and downward forces that act on the vessel. When a body floats in still water, it experiences two acting forces: a downward force due to gravity (the weight) and an equal and opposite (upward) force due to hydrostatic pressure. The upward force is the buoyancy and its magnitude is equal to the mass of water displaced by the object. This is also true

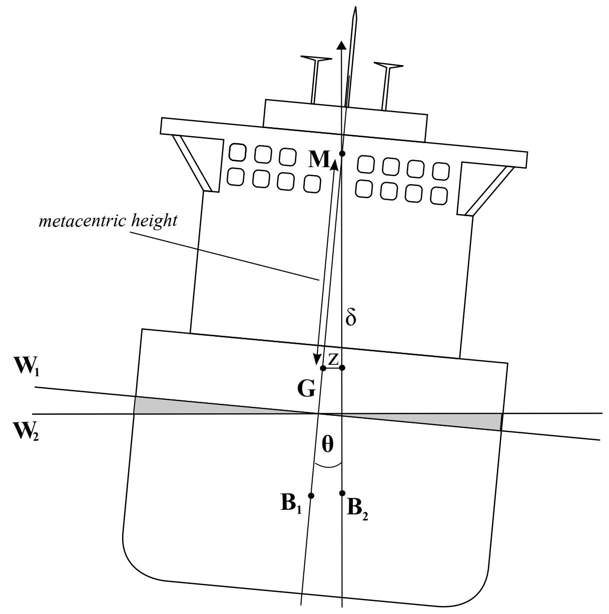


Figure 2.8: Transversal section of a vessel showing initial stability components.

for container vessels. Even though the total weight and buoyancy forces are equal, the shape of the hull and the uneven distribution of weight results in an uneven distribution of forces along the vessel which stresses its structure. Each vessel has a fixed set of calculation points called *frames* for which stress limits are known. Consider Figure 2.9. It shows a longitudinal section of a container vessel with 14 frames. The W curve represents the weight distribution along the vessel and the O curve is the corresponding buoyancy. For each of the sections between each frame, the arrows denote the resulting forces acting on each section. Should each one of these sections be allowed to freely move, sections with a stronger downward force will sink more into the water while the opposite would be true for those with stronger upward forces. Since the sections are not allowed to move, they cause shearing and bending stresses over the vessel structure. The *shear force* on a vessel, at a specific frame, is the integral of the forces on either side of the frame. Similarly, the *bending moment* at each frame is the integral of the moments on either side of the frame; another way of seeing this is using the *load curve*, which shows the resulting force per length unit where moment is defined as force times distance. Shear force is then the area under the load curve, from either the bow or the stern to the frame of calculation. Shear force is shown in Figure 2.9 as the curve S. The image also shows the bending curve B for the bending moment of the vessel, which can also be computed as the area under the shear force curve S.

Modern container vessels are equipped with *ballast tanks*. These tanks can be used to load water, and thus distribute extra weight along the vessel to, for example, improve vessel stability or adjust weight distribution to reduce stress forces. Automatic pumps can move water between the tanks, which is particularly useful during load and

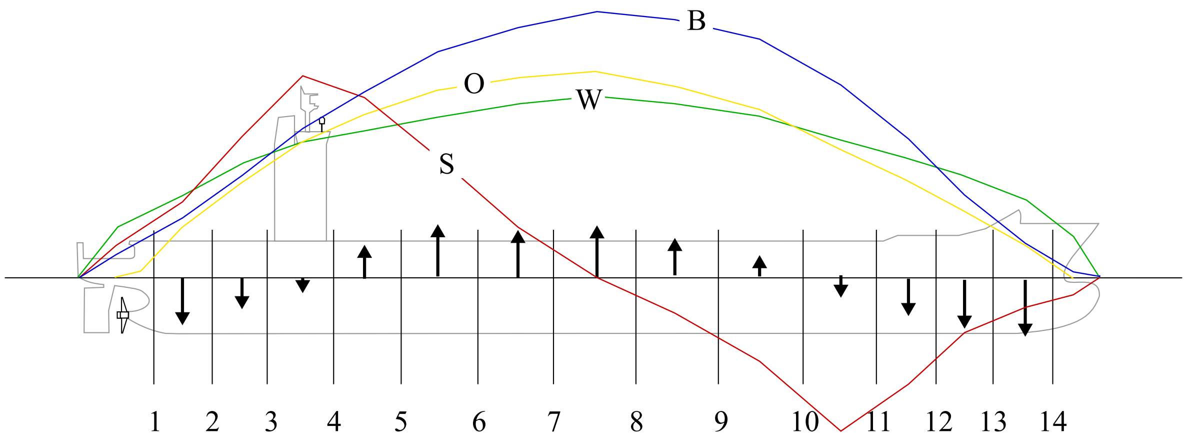


Figure 2.9: Longitudinal section of a vessel and stress curves.

discharge operations.

2.4 Container Stowage

Previous sections described general conditions that are necessary for a vessel to be seaworthy. Containers, however, cannot be stowed on a vessel freely. A number of rules and constraints must be followed when stowing containers. Some of these rules have already been presented in Section 2.2 in relation to the restrictions that must be observed when stowing special containers such as IMOs and pallet-wides. Aside from these last rules, which have an impact between stacks, most stowage rules can be explained from a single stack prospective and then extended to all the stacks in a bay. *Holds*, the below-deck parts of a bay, are mainly 40' long and can stow either one 40' container or two 20' containers (there are 45' long holds but they are not common). Below-deck containers are transversally secured by *cell guides*, which are found on each end of a stack. In between each tier of containers, four *stacking cones* are placed in each of the corner fittings, thus minimizing the possible movements of the containers. The same stacking rules between containers apply also here, thus 20' containers cannot be stowed on top of 40' containers. Notice, however, that it is not allowed to stow only one single 20' container on the top tier because the container would not be sufficiently secured. Internal company rules might also require a 40' container on top of a stack of 20'containers should it reach a given tier level. Due to the shape of the hull and the presence of tanks and other structural volumes, it is possible that some cells, in stacks in lower tiers, only have one TEU capacity. Cells with one TEU capacity are called *odd-slots* and force the stowage of one 20' container before any other container can be stowed in the stack. Some holds are provided with cell guides in the middle of the stacks, and can thus only stow 20' containers. All data regarding vessels and their layouts is collected in one document called the *vessel profile*. Vessel profiles are important during stowage operations as they describe the capacities of each slot. Vessel profiles can be used to check where power outlets are available, such that it is possible to know where, for example, reefer containers can be stowed.

Container stowage on-deck has the same basic stacking restrictions as in stowage below-deck. It is, however, uncommon to have *Russian stacks*, stacks of mixed 20' and 40' containers, on-deck. Container stacks on-deck are not supported by the vessel structure as the ones below-deck, thus containers must be secured in a different way. The stacking cones used below-deck are replaced by *twist-locks*, which similar to cones, fit into the corner castings, but that can lock containers between each other when the cone heads are twisted. Nowadays it is common to use automatic twist-locks which release the containers automatically once a crane lifts them. The first two tiers of containers are secured using *lashing-rods*. The use of lashing-rods presents, however, a restriction on the positioning of 20' containers. Ship personnel must be able to reach in between two 20' containers in order to tighten the lashing rods. Should a 40' container be stowed in the path of 20' containers it would not be possible to lash those last containers. All containers above the second tier, that cannot be lashed, are only supported by the twist-locks. For that reason, stacks on-deck with more than two tiers must have at most a tier difference of one in order to support each other. Some vessels are equipped with a *lashing bridge* (see Figure 2.5), a metallic structure similar to the one found below-deck. Containers within the lashing bridge are firmly secured, and the bridge gives the possibility of lashing the next two tiers on top of it. In that way, it allows much higher and secure stacking of containers. If a lashing bridge is present, it must be noted that long containers such as 45' containers are only allowed above the bridge. It is not allowed, even though physically possible, to stow 40' on top of 45' containers since the 40' cannot be lashed. The rolling moment of a vessel also influences the way containers should be stacked on-deck in terms of security. Vessels having a short rolling period, and thus a large *GM* (see Section 2.3.1 for the definition of *GM*) experience strong forces on the top tiers. For that reason, it is preferable to have light containers at the top of a stack and heavy at the bottom. If heavy containers are at the top, the resulting forces might collapse the entire stack. High container stacks cannot be freely stowed on-deck due to wind forces. Shipping companies have different *wind-stacking* rules. An example of such a rule is to not allow stacks to be more than one tier higher than the ones supporting them on the side. The height that container stacks can reach is also governed by the *line-of-sight* of the pilot house. The line-of-site is a fictitious line from the pilot house to the sea at a distance equal to double the length of the vessel, or 500 meters, which ever is smallest. Containers are not allowed above this line or they will obstruct visibility, an example of which can be seen in Figure 2.5.

2.5 Stowage Plans

A stowage plan is an assignment of containers to slots in a vessel. Stowage plans are generated by stowage coordinators before a vessel reaches its destination port. Stowage plans are then used in the terminal to coordinate the load and discharge operations. Section 2.1 gave a description of the main costs of terminal operations, and mentioned that those, and the tight schedule of liner vessels, result in a wish to minimize the time at port. It was already shown in Section 2.1 how the distribution of moves on each of the vessel's bays has a direct impact on the crane-split and thus the total time that terminal operations will take. Another, and very important, factor to the minimization of time at port, is *overstowage*. Consider a container a to be discharged in an arbitrary port, and container b to be discharged at a later port. Once the vessel reaches the port of discharge of container a, if container b is stowed on top of container a, b will need to be discharged so that container a can be reached. Container b will then have to be loaded again into the vessel to continue to its port of destination. Container b is said to *overstow* container a and the extra moves caused by this overstowage are called *shifts*. There also exists a more costly kind of overstowage, one that involves hatch-covers. Consider again container a, but this time assume it is stowed below-deck. Once container a has to be discharged, all the containers over the hatch-cover, that do not have the same discharge port as a, will be overstowing a independently of which stack they are stowed in since it is necessary to remove the hatch-cover. All these containers will results in shifts. Clearly, stowing containers with a later discharge port first will reduce this problem. However, such stowage would result in a horizontal layer distribution of discharge ports which could have a disastrous impact on the stowage plans at later ports. It is for this reason, that stowage coordinators must take future ports into account when decisions about the stowage plans are taken. Normally stowage coordinators have about six hours to produce a stowage plan. The information at their disposal is: a *loadlist* containing information (weight, size and discharge port) about the containers to be loaded on the vessel, a *release* containing the same information about the containers already onboard the vessel, a *forecast*, based on historical data, giving information about the containers to be expected in down stream ports, and port data about the string, such as number of assigned cranes and port depths. In order to help stowage coordinators produce valid plans, they are often given access to specialized software that can, in real time, calculate and give feed back on the state of a vessel given a specific stowage plan. Stowage planning then becomes a complex puzzle governed by expert guidelines, experience and the trial and error feed back of the currently available software packages. One of the most successful stowage guidelines is *block stowage*. When block stowing, stowage coordinators attempt to assign discharge ports to section of bays to minimize crane workloads and overstowage at current and future ports. The containers in the loadlist are then loaded within those blocks if possible. Stowage coordinators also use a number of rules-of-thumb that allow them to produce robust plans, such as making sure that reefer plugs are correctly utilized such that they can be available when containers have to be stowed in future ports.

Chapter 3

Related Work

Previous academic work is rich of innovative solution approaches, but has been challenged by the inaccessibility of the problem. This has led to limited representative power of the developed models and a lack of test instances. The work can be clustered into two main groups: approaches based on a single model of the entire problem and approaches based on a decomposition of the problem into several optimization models. Between the methods aiming at optimality, the latter approaches are the most successful, however, most of the work in the literature focuses on heuristic approaches using a single model. To the best of the authors knowledge, this chapter presents a complete review of the academic work for the container stowage problem.

3.1 Single Model Approaches

Mathematical Programming

Within the realm of single model optimization, mathematical modeling has been applied by Ambrosino et al. (2004) and Li et al. (2008). Ambrosino et al. (2004) use a 0-1 Integer Programming (IP) formulation that considers 20' and 40' containers within three weight classes. The model indirectly deals with lashing and *GM* constraints by forcing containers to be ordered ascending by weight from the bottom to the top in stacks. Containers with special requirements, such as reefers, are not modeled and overstowage is modelled as a constraint rather than an objetive. The goal of the model is to minimize time at port. Using a preprocessing procedure, the proposed IP model can solve stowage instances for a single vessels of 188 TEUs within 33 minutes. The model, however, only considers the current port into account. Li et al. (2008) also propose a 0-1 IP model, but take a multi-port scenario into account. Similar to Ambrosino et al. (2004) only standard 20' and 40' containers are modelled. This work, however, does not consider weight limitations but does represent overstowage as an objective. The authors claim to solve random generated problem instances for a single vessels of 800 TEUs but present no details of the runtime results.

Placement Heuristics

Pure placement heuristics have been explored by Avriel et al. (1998) and Yoke et al. (2009). Avriel et al. (1998) presented the *Suspensory Heuristic Procedure*, which is restricted to a single bay vessel and solves the stowage problem of a single size container without taking into account the stability of the vessel. The heuristic is aimed at minimizing overstowage in a multi-port problem where only loading is allowed. Randomly generated problem instances with vessels of up to 1700 TEUs can be solved within 30 seconds. In the same paper, the authors also present an IP model that can solve this problem to optimality for very small instances. Yoke et al. (2009) proposes a placement heuristic based on the work practice of stowage coordinators. Containers are initially grouped by discharge port, length, and type of special requirements. The vessel is then partitioned in so called blocks to which discharge ports are assigned. After this, containers are stowed into each group using heuristic rules. These rules minimize overstowage but do not embed stability considerations. The authors show evidence that one problem instance for a single vessels of 5000 TEUs can be stowed efficiently. The feasibility of the plan is, however, not guaranteed. Some improvements to the heuristic rules are presented by Fan et al. (2010) in order to take stability into account.

Placement Heuristics with Local Search

Some of the earliest work combines a placement heuristic placement with local search. Scott and Chen (1978) propose a grouping of containers into classes based on special requirements, leaving the last class to hold all the standard containers which are then further classified according to their weight. Containers with special requirements are assigned to slots first based on some heuristic rules. Two IP models are then solved, one to assign the standard containers to bays and maximize intake, and one to optimize ship stability. A local search using container swaps as a neighborhood is used to solve the trim and GM. Should it not be possible to find a solution, the process is repeated by removing one container form the loadlist. The method does not take overstowage into account and has long computation times according to the authors. Aslidis (1984) solved a simplified version of the problem with one size of container without special requirements. Overstowage minimization was the only objective and weight capacities were not taken into account. The authors propose a placement heuristic for the containers where trim is solved using ballast water. A local search based on swap neighborhoods is used to solve the GM. Once a feasible solution is found, swaps with no cost on the objective value are performed to maximize GM. Two test cases on a 1500 TEU vessel are solved within 30 seconds. Similar ideas have recently been investigated by Liu et al. (2011) and Low et al. (2011). Low et al. (2011) improves the solutions given by the placement heuristic of Yoke et al. (2009) using a local search with swap neighborhoods that are specifically designed to solve weight limits, trim, and heel requirements. The work of Liu et al. (2011) modifies the heuristic of Yoke et al. (2009) to obtain non deterministic solutions. In a first stage the modified heuristic is used to generate a number of initial solutions. The best of these, in terms of overstowage and crane intensity, are then used in a second stage. The second stage uses a swap neighborhood limited to containers with the same discharge port to solve the stability as a multi-objective problem. For the vessel of 5000 TEUs examined by the authors, 10 non-dominant solutions can be found within 2600 seconds.

Bin Packing Heuristic

The similarity between the stowage problem and the Bin Packing problem has been investigated by Sciomachen and Tanfani (2003) and Zhang et al. (2005). Sciomachen and Tanfani (2003) adapt a 3D-Bin Packing heuristic to a stowage problem which takes into account 20' and 40' containers and high-cube containers. The vessel is divided into sections that are filled by containers according to their destination and length. Main sections have simple forms and are easy to stow, while remaining sections are left for later stowage. Stability and stacking constraints are considered in the loading order of containers and sections. Generated plans are designed for only one port. For a vessel of 1800 TEUs, the two industrial cases can be solved within 33 minutes. Zhang et al. (2005) propose a bin packing model of the multi-port stowage problem. Stability constraints, however, are not modelled. Overstowage minimization and minimization of the number of used bays are optimized by modeling the packing decisions as a binary decision tree traversed using bin packing heuristics. One case study with a vessel of 895 TEUs is presented but no runtime performance is presented.

Genetic Algorithms

Dubrovsky et al. (2002) propose a compact encoding of a container stowage problem similar to that described by Avriel et al. (1998). In this encoding, a single bay vessel can be stowed with one size of container. Stability and container weights are not taken into consideration. The solution process does, however, heuristically handle trim requirements. Random problem instances, with vessels of 500 and 1000 TEUs, have been solved for 10 ports in 30 minutes. Another work that makes use of genetic algorithms is that of Imai et al. (2006) where container stowage and terminal sequencing are combined. Sequencing is introduced by modeling the transportation time needed for each container to be retrieved from the yard. Plans are only produced for the holds and overstowage is modeled as a constraint. Stability is taken into account as well as GM, trim, and heel angles. The algorithm, however, takes over 1 hour to generate plans for random problem instances of vessels in the range of 500-2000 TEUs.

Constraint Programming

The use of Constraint Programming (CP) for container stowage was introduced by Ambrosino and Sciomachen (1998). In this work, a single bay vessel loading 350 containers is modeled. The approach takes 20', 40', and IMO containers into account, and has a heuristic handling of the stability constraints. The CP model is optimized within a branch & bound framework where overstowage and intake are optimized. The model also considers the use of Roll-On Roll-Off vessels. The authors claim to solve problem instances within 10 minutes, but no evidence is presented. Later, Delgado et al. (2009) presented a different CP model which takes into account most of the stacking constraint for vessel holds, but disregards IMO restrictions and stability. A detailed description of the model and its improvements, such as symmetry breaking, channeling constraints and partial evaluations are presented. An extension of this work was later published (Delgado et al., 2012) where an in-depth comparison with an IP model is shown. The approach can solve industrial problem instances of 100 TEUs vessel partitions to optimality within 1 second (in most cases). It must be noted that this model solves the slot planning problem of the decomposition described in this thesis, thus stability is assumed to be handled during the master planning phase. This is also the approach used to generate the optimal solutions for the evaluation of the CBLS algorithm presented in this thesis.

Simulation

Simulation in stowage planning is found in one of the earliest (Shields, 1984) and one of the latest (Azevedo et al., 2012) contributions. Shields (1984) uses a Monte Carlo model with the aim of handling the uncertainty of cargo in the loadlists. For each port, container groups are loaded into the vessel using heuristic guidelines that try to prune slots where the analysed containers cannot be stowed and thus aim to find bays that can stow the entire group. Penalty functions are used to rank solutions and the three best ones are selected for each port. An overall best solution between all ports is then selected. No performance results were published. Azevedo et al. (2012) uses simulation in a similar fashion, however, stochasticity is not taken into account. Azevedo et al. (2012) proposes a solution search based on the selection of the loading and unloading rules applied by stowage coordinators. The idea is to decide which overall rules to use, and simulate them in order to generate a plan. The decision tree is then traversed using beam search in order to reduce the search space. Stability and standard one size containers (with unitary weight) are taken into account. The model optimizes overstowage and stability with a runtime between 400 and 18000 seconds for a 1500 TEU vessel.

Other Approaches

Giemsch and Jellinghaus (2004) present an alternative approach to the uncapacitated single bay problem, where a 2-step heuristic stows containers by their discharge ports into stacks. The remaining containers, those for which a stack that does not force rehandles cannot be found, are assigned to slots using a branch & bound search. The authors claim to achieve better results than the Suspensory Heuristic Procedure by Avriel et al. (1998), but no detailed results have been published. The same problem is solved by Yanbin et al. (2008) using a multi agent system, which, however, only achieves results comparable to those of the Suspensory Heuristic Procedure. Nugroho

(2004) presents a very different approach, a case-based system. The idea is to find a stowage plan similar to the one in question within a knowledge base. The database of cases will then grow and be more accurate the more stowage plans are solved. No sufficient experimental evaluation of the idea is presented.

3.2 Decomposition Models

To the best of the author's knowledge, the first decomposition of the stowage problem was proposed by Botter and Brinati (1992). In this work, a detailed mathematical model of the container stowage problem was presented. The model also included decisions over the sequence in which containers should be loaded or discharged. Botter and Brinati (1992) proposed to solve the problem using a decomposition which first solves the assignment problem of containers to slots, and then the sequencing problem. Given a solution to the assignment problem, the set of sequencing variables was greatly reduced. The decomposition, however, was too computationally expensive and was replaced by a branch & bound search with domain specific branching heuristics. The authors claim that the search could be stopped within acceptable computational times and result in good quality stowages. No evidence of these results was presented.

The first decomposition approach that presented promising results was the work of Wilson and Roach (1999)(Wilson and Roach, 2000; Wilson et al., 2001) where the block based decomposition was introduced. Wilson and Roach (1999) divided each bay of the vessel into blocks. Blocks are logically distributed such that they are either on or below-deck and often follow the pattern of the hatch covers. The proposed decomposition first solves an assignment problem from container groups to blocks. This problem is solved using an enumeration algorithm where solutions are graded by a fitness function. The algorithm optimizes overstowage, crane utilization, and discharge port clustering within blocks. Given this assignment, a tabu search is run to find the assignment of containers within each block. Here overstowage is minimized, container weights are ordered, and stacks with the same discharge port are preferred. Without presenting any evidence, the authors claim to solve problem instances for 688 TEU vessels. A total of 90 minutes is necessary for the block assignment problem, and less than 1 hour for the tabu search to be completed.

Kang and Kim (2002) presented an iterative decomposition, where information is passed between the master and the sub-problem. Kang and Kim (2002) have also adapted the concept of block decomposition, and, as Wilson and Roach (1999), the first stage of the decomposition assigns container groups to blocks. Only one size of container with no special requirements is taken into consideration. The assignment problem is not solved for all ports simultaneously but for one port at a time with the objective of reducing overstowage and stability violations. Solutions are found using a modified version of the transportation simplex that uses specialized pivoting rules. The classes of containers are then assigned to the slots using a tree search enumeration for each block. Information from the slot assignment is after this passed to the block assignment algorithm in order to find improving solutions. The approach can solve randomly generated problem instances for 4,000 TEU vessels using three weight classes for the containers in 640 seconds, while planning for eight ports.

A different decomposition approach is proposed by Ambrosino et al. (2006) that is based, however, on the assumption that overstowage is a hard constraint in the model. The authors present a 3-phase decomposition. In the first phase, containers are grouped according to their discharge port and are distributed among the bays (which are devoted to one discharge port) taking into account capacities. For each defined partition of the vessel, an IP model is solved that assigns containers to specific slots. Once all partitions are solved, a local search uses a swap neighborhood to improve the stability of the vessel. The approach generates stowage plans for a single port, and with a vessel of 198 TEUs, it runs in 3.5 minutes and finds solutions with a one percent optimality gap compared to the IP model of Ambrosino et al. (2004). Later, the authors propose a modification of the decomposition (Ambrosino et al., 2009) where a new IP model for the solution of the single destination partitions is presented and the simple local search is changed to a tabu search. The new modifications allow the decomposition to scale up to a 2,124 TEUs vessel with a runtime of 74.7 seconds. Further research on the decomposition model was presented by Ambrosino et al. (2010), where a constructive heuristic was developed and used after the initial bay assignment. The heuristic is based on Ant Colony Optimization where the first decision is the assignment of containers to stacks and the successive decision is the assignment of containers to slots. The found solution is then improved using local search. The ant system is updated (pheromone update) and a solution is retuned once the ants start converging. The new heuristic was shown to solve problem instances for a 5,632 TEU vessel in 139.4 seconds. Multi-port solutions and overstowage have not yet been modeled.

Gumus et al. (2008) presents a multistage decomposition, where containers are grouped into types. In a first stage, fractions of bay capacities are assigned to specific discharge ports. The assignment is then refined at the tier level such that hatch covers can be taken into account. Then, further refinement is made at the slot level. The fourth, and last, stage assigns containers to the slots according to the assigned port of discharge and uses a heuristic to handle vessel stability. Aside from the first stage, solved with a mixed-integer program, the authors do not provide details of the other stages. The decomposition is claimed to scale to large instances but no evidence is presented.

Chapter 4

Representative Problem and Algorithmic Framework

Chapter 2 gave an in-depth overview of all the details and constraints that surround the container stowage problem. From a research point of view, it would be unpractical to study models of the problem that include all those details. Instead we propose to study a representative problem. This problem should include all the core computational components of the container stowage problem, and, at the same time, be detailed enough for the industry to be able to evaluate the value of the obtained results.

4.1 A Representative Problem

The definition of a representative problem requires balanced decisions about which parts of the original problem should be relaxed and which should be always taken into account.

4.1.1 Container Types

The computational complexity of the container stowage problem does not depend on the type of containers used, as the assignment problem itself is nontrivial (see Appendix A). Such an observation could lead to the representation of only one container type. Stowage coordinators, however, know that it also can be hard to find stowage plans where 20' and 40' containers are mixed. This is due to the fact that 20' containers cannot be stowed on top of 40' containers and that the vessel arrives non-empty. For this reason a representative model should at least include containers of these two length's. Containers with other lengths are not very common, with exception of the 45' containers which, however, are not involved in complex constraints.

Containers with special requirements, such as reefers, IMO, OGG and pallet-wides, can also have a significant impact on the complexity of the problem. Pallet-wides are usually used on specific routes and thus are not representative of the general problem. OGGs, such as bulk cargo, are not very common and one can immagine them as being handled by the stowage coordinator and thus simply resulting in a capacity reduction. The same can be said about IMO containers since vessels often have dedicated bays for those containers towards the bow. One could easily imagine models where IMOs are forced to be stowed on such bays and where the final decision of their positioning is left to the stowage coordinators. Reefer containers are more common, and represent an everyday challenge for stowage coordinators. Due to the fact that vessels have a limited number of reefer plugs, the positioning of those containers can greatly influence a stowage plan. For this reason a representative stowage problem should include reefer containers. It is debatable whether high-cube containers should be included into a representative problem. It is true that the extra height of those containers might lead to a reduction in the capacity of a vessel, but a the same time from a computational stand point they can be seen as extra capacity constraints, and thus redundant from a computation prospective.

4.1.2 Vessel Layout and Routes

Considering the shape of the vessel is important for solving the container stowage problem as it posts major constraints on the possible distribution of containers. Also from the prospective of the industry, it would be very hard to evaluate solutions based on rectangular vessels, as there would be a good chance that the produced stowage plans will not look anything like the ones stowage coordinators are used to seeing. For a stowage problem to be representative it should consider the layout of the vessel, including the separation between on and below-deck with hatch covers. A problem that does not take hatch covers into account cannot model the hatch-overstowage, which is a significant part of the overstowage calculation. We, however, do not regard *odd-slots* (cells that only allow the stowage of one TEU) as part of the representative problem since it would only result in extra capacity constraints that do not have an impact on the computational complexity of the problem or on the way stowage coordinator would evaluate the results.

Vessels should also be considered to have an initial load of containers. This initial load forces stowage plans toward specific configurations, which is a property that should be exploited from a computational point of view. Moreover, a stowage problem that includes already onboard containers is more representative of the everyday planning that a stowage coordinator has to perform. All ports of the plan should also be considered. Besides the fact that stowage coordinators judge stowage plans also on their robustness, a multiport problem has a higher level of complexity which cannot be ignored.

4.1.3 Vessel Stability

It is clear that the stability of the vessel is of high importance for the industry, and thus automated stowage plans should be as accurate as possible in that regard. Stowage plans that do not take stability into account, or that are too inaccurate, might force stowage coordinators to make big changes to the plan, which defeats the purpose of solving the problem. To give an example, due to the shape of the vessel it can be hard to handle bending moments as there is not enough capacity at the extremities of a vessels. A solution to a container stowage problem that disregards bending might require the stowage coordinator to change its container distribution such that less weight is assigned at the vessel extremities. With regard to stress forces and ship stability, ballast water can have a significant impact as well. Since the release of ballast water can have significant environmental impacts, it is debatable whether the amount of water in the tanks will change drastically within a stowage planning horizon. Nevertheless a representative stowage planning problem should at least take the weight distribution of the water into account.

4.1.4 Container Stowage

Clearly, basic container stacking rules should be part of the representative problem. Vessels tend to be loaded close to their maximum capacities. When combining this fact with the optimized construction of the vessels, it is clear that it is not possible to disregard the structural limitations of a ship such as the weight constraints of the stacks. Due to the presence of hatch-covers and the requirement for line-of-sight, it is also unreasonable to model problems that consider uncapacitated stacks. Representative models should also consider on-deck wind stacking rules (at least heuristically) since inter-bay container arrangements can be restrictive, especially in the cases where containers with different discharge ports are stowed in the same bay.

4.1.5 Container Stowage Objectives

Overstowage has the largest impact on the efficiency of a stowage plan as it results in extra moves. Any representative model should include overstowage both for the current and down-stream ports. It is also important to include hatch overstowage in the objective calculation due to the significant number of extra moves that it could enforce. Different ways of defining crane utilization exist, and those ways are most likely company and contract dependent. One thing that is common is that the arrangement of containers in a vessel will be heavily affected by it, thus a representative model should in some way take those concepts into account. Ballast water minimization is often required for optimizing bunker consumption. As previously mentioned, releasing ballast water can have environmental consequences thus it is possible that only a limited amount of water can be released. For that reason we do not believe that ballast optimization should be part of the representative problem. The handling of the current condition of the tanks should, however, still be taken into consideration. When discussing with stowage coordinators, one can find a large number of characteristics that each stowage coordinator thinks a stowage plan should have. We believe that such considerations, can be left out of a representative model.

4.1.6 The proposed problem

In summary, we propose the following representative container stowage problem.

Definition 1. *Given a vessel profile that includes hydrostatic data, capacity and layout of bays, and given data about the ports in the planned route including a loadlist of containers at the current port and a forecast for the containers in the down-stream ports, generate a stowage plan for the containers in the loadlist such that:*

1. *Both 20' and 40' containers are modelled*
2. *Reefer containers are stowed near power plugs*
3. *Already onboard containers are considered*
4. *Container stacking constraints are satisfied*
5. *Stack weight and height limits are respected*
6. *Trim, heel, draft and GM are within limits*
7. *Shear forces and bending moments are within limits*
8. *Wind stacking rules are taken into account*
9. *Loading and discharge of containers is allowed for all planned ports*

The generated stowage plans should then minimize overstowage, hatch overstowage, and maximize crane utilization.

4.2 The Quad Framework

The work in this thesis is part of a research project that uses an algorithmic framework called *Quad*. This framework assumes a block decomposition similar to that of Wilson and Roach (1999) and Kang and Kim (2002). The problem is decomposed into a *master planning problem* and a *slot planning problem*. During master planning we solve the problem of distributing groups of containers to sub-sections of bays called *locations*. The containers are grouped based on their weight, length, and special requirements (e.g. reefers). In order to achieve a robust plan, the framework assumes that the master planning problem is solved for the current port and a number of down-stream ports. The input of master planning is thus the loadlist of the current port and a forecast of the containers to be transported in the future ports. The input also includes the vessel profile and data about the ports the stowage is planned for (e.g. maximum draft, cranes available, etc.). It is in this phase that stability constraints are handled and the hatch-overstowage and crane utilization are optimized. The output of the master planning phase is an assignment of container groups to locations for the current port, which is the port we want to generate the stowage plan for. This output becomes the input of slot planning. The Quad framework assumes that the way locations are defined in the vessel is such that changes in how the containers are distributed within a location does not affect the stability calculations significantly. In other words, as long as the distribution of container groups between the locations is maintained, stability and stress calculations can be discarded during slot planning. Slot planning thus only handles stacking constraints while deciding the slot assignent of each container in the container groups associated with each location. The assignent of containers, during slot planning, is optimized with respect to overstowage within each location. The generated output is a complete stowage plan for the current port.

The Quad framework is aimed at capturing the strict time requirement imposed by the industry. It does so by first attempting to solve each of the decomposition problems using complete methods, and then resorts to heuristic search should the method of choice fail to find solutions in time. Figure 4.1 shows a graphic representation of the Quad framework, where the dotted arrows show the possible ways that complete and heuristic methods can interact within the framework.

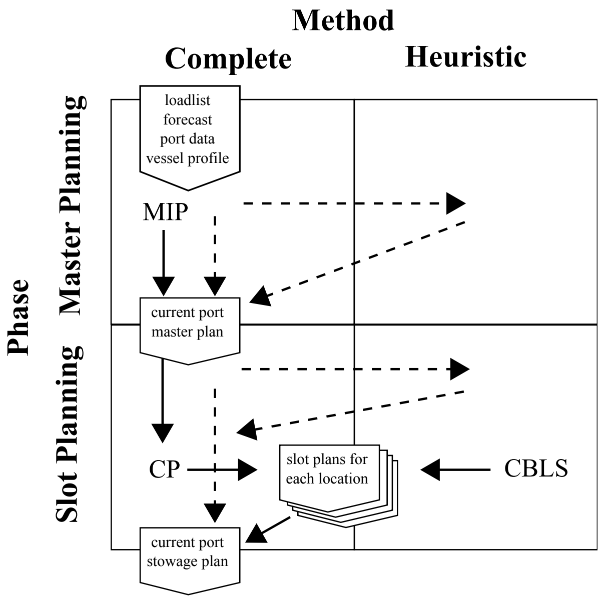

Method
Complete
Heuristic
Phase
Master Planning
Slot Planning
loadlist forecast port data vessel profile
MIP
current port master plan
CP
slot plans for each location
CBLS
current port stowage plan

Figure 4.1: The Quad framework

Using an arrow with a solid line, the figure also shows the current state of the implemented Quad framework. Master planning is successfully implemented as a mixed-integer problem, the details of which are described in Section 5. Slot planning has been implemented using different solution approaches. Complete methods have been explored by Delgado et al. (2009) where CP was shown to outperform both an IP model and a column generation approach. The heuristic search has been implemented using CBLS, the details of which are presented in Section 6. Once a solution to master planning is found, the current implementation uses the CP model to solve a slot planning problem for each location. The CBLS model is used in all the cases where the CP model fails to find solutions. The mixed-integer programming model for master planning and the CBLS model for slot planning are part of the work behind this thesis.

Chapter 5

Master Planning

The master planning problem has turned out to be the computationally most challenging part of the employed decomposition. It is in this phase that the general distribution of containers is decided. During master planning, high level constraints such as stability and stresses are handled. A good approximation of the stability and stress calculations can be obtained by only considering the weight of the cargo in each vessel location. Thus, the exact position of the containers is not important in this phase. Master planning takes into consideration all the ports in the planned route and for each port optimizes hatch-overstowage and crane utilization. The following sections will present first an efficient master planning model that can be used to solve industrial strength master plans and subsequently a refined model for the stability and stress calculations with an analysis of its accuracy.

Before describing the master planning problem it is important to have a better understanding of what locations are. Figure 5.1 depicts how locations are distributed in bays. Locations are either on-deck or below-deck, and coincide with the position of

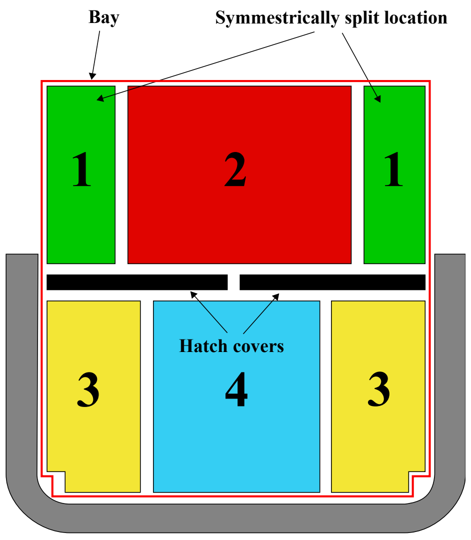


Figure 5.1: Locations in a bay

the hatch-covers, which makes it easier to model hatch overstowage. Some locations are not composed of a continuos row of stacks. This is the case of locations representing the extremities of a bay. This split and symmetric distribution of stacks is a useful tool for the modeling of the transversal stability. With the assumption that containers and their weights are symmetrically distributed in the locations, it is possible to disregard stability constraints about the heel angle.

5.1 A Model for Master Planning

The master planning problem described in this thesis considers both 20' and 40' containers including, as an example of containers with special requirements, reefer containers. Containers are grouped into types and are assigned to locations. Locations are then used as basic elements for the stability and stress calculations. We model trim, GM, draft, and shear forces as an example of stresses. Assuming that stacking constraints are handled during slot planning, this model only lacks the handling of the bending moment and the wind-stacking rules compared to the representative problem describe in Chapter 4. Ballast tanks are included in the calculations, but are considered constants. Thus it is debatable to consider them as part of the problem.

More formally, master planning assigns container types to locations. For 20' and 40' containers, we consider a set of four mutually exclusive container types $T = \{L, H, RL, RH\}$, respectively light and heavy containers and light and heavy reefer containers. To produce a robust plan, our model takes into account the current and a set of down-stream ports P. We define transports TR as a set of pairs $\langle p_1, p_2 \rangle$, where $p_1, p_2 \in P$ are the load and discharge port of a container type. Notice that the container types T are only a classification and thus are not bound to fixed weight ranges. For each type $t \in T$, the average weight of the containers on a specific transport $t \in TR$ is calculated and represented by the constants $W_t^{20\tau}$ and $W_t^{40\tau}$. This results in a more refined weight categorization than taking the average weight of the containers in each container class. We define two sets of decision variables $x_{tl}^{20\tau}$ and $x_{tl}^{40\tau}$ representing respectively the number of 20' and 40' containers of type $\tau \in T$ to be stowed in location $l \in L$ during transport $t \in TR$, where L is the set of all locations. Following is the proposed IP model:

minimize

$$\sum_{p \in P} \sum_{l \in L} \left(\mathcal{C}^O y_{pl}^O + \mathcal{C}^P y_{pl}^P + \mathcal{C}^R y_{pl}^R \right) + \sum_{p \in P} \mathcal{C}^T y_p^T \tag{5.1}$$

subject to

$$\sum_{t \in TR_p^{ON}} \sum_{\tau \in T} \left(x_{tl}^{20\tau} + 2x_{tl}^{40\tau} \right) \leq C_{pl} \quad \forall p \in P, l \in L \tag{5.2}$$

$$\sum_{t\in TR_p^{ON}} \sum_{\tau\in\{RL,RH\}} \left(x_{tl}^{20\tau} + 2x_{tl}^{40\tau}\right) \leq C_{pl}^{R} \quad \forall p \in P, l \in L \tag{5.3}$$

$$\sum_{t\in TR_p^{ON}} \sum_{\tau\in T} x_{tl}^{\alpha\tau} \leq C_{pl}^{\alpha} \quad \forall p \in P, l \in L, \alpha \in \mathfrak{L} \tag{5.4}$$

$$\sum_{l\in L} x_{tl}^{\alpha\tau} = LD_t^{\alpha\tau} \quad \forall \alpha \in \mathfrak{L}, \tau \in T, t \in TR \tag{5.5}$$

$$\sum_{t\in TR_p^{ON}} \sum_{\tau\in T} \sum_{\alpha\in\{20,40\}} W_t^{\alpha\tau} x_{tl}^{\alpha\tau} \leq W_{pl} \quad \forall p \in P, l \in L \tag{5.6}$$

$$G_p^{-\rho} \leq \sum_{l\in L} G_l^{\rho} \sum_{t\in TR_p^{ON}} \sum_{\tau\in T} \sum_{\alpha\in\mathfrak{L}} W_t^{\alpha\tau} x_{tl}^{\alpha\tau} \leq G_p^{+\rho} \quad \forall p \in P, \rho \in \{L, V\} \tag{5.7}$$

$$\mathcal{S}_{pf}^{-} \leq \sum_{l\in L_f^{Aft}} \sum_{t\in TR_p^{ON}} \sum_{\tau\in T} \sum_{\alpha\in\mathfrak{L}} W_t^{\alpha\tau} x_{tl}^{\alpha\tau} \leq \mathcal{S}_{pf}^{+} \quad \forall p \in P, f \in F \tag{5.8}$$

$$\sum_{i\in L_l^U} \left(R_{pi}^{D} + \sum_{t\in TR_p^A} \sum_{\tau\in T} \left(x_{ti}^{20\tau} + x_{ti}^{40\tau}\right) \right) \leq M\delta_{pl} \quad \forall p \in P, l \in L^O \tag{5.9}$$

$$R_{pl}^{OV} + \sum_{t\in TR_p^{OV}} \sum_{\tau\in T} \left(x_{tl}^{20\tau} + x_{tl}^{40\tau}\right) - M(1-\delta_{pl}) \leq y_{pl}^{O} \quad \forall p \in P, l \in L^O \tag{5.10}$$

$$R_{pl}^{D20} + \sum_{t\in TR_p^A} \sum_{\tau\in T} x_{tl}^{20\tau} \leq M\phi_{pl} \quad \forall p \in P, l \in L \tag{5.11}$$

$$R_{pl}^{OV40} + \sum_{t\in TR_p^{OV}} \sum_{\tau\in T} x_{tl}^{40\tau} - M(1-\phi_{pl}) \leq y_{pl}^{P} \quad \forall p \in P, l \in L \tag{5.12}$$

$$C^T \sum_{t\in TR_p^A} \sum_{l\in L_b} \left(R_{pl}^{A} + \sum_{\tau\in T} \left(x_{tl}^{20\tau} + 2x_{tl}^{40\tau}\right) \right) \leq y_p^T \quad \forall b \in B, p \in P \tag{5.13}$$

$$\sum_{t\in TR_p^{ON}} \sum_{\tau\in T} F_{pl}^{\tau} \left(x_{tl}^{20\tau} + 2x_{tl}^{40\tau}\right) - C_{pl}^{R} \leq y_{pl}^{R} \quad \forall p \in P, l \in L \tag{5.14}$$

The TEU capacity of each location $l \in L$ at each port $p \in P$, is restricted by constraints (5.2-5.4), where TR_p^{ON} is the set of all the transports on the vessel at departure from port p. Maximum TEU and reefer capacity limits are given respectively by the constants C_{pl} (5.2) and C_{pl}^{R} (5.3). Locations can have restrictions about the kind of containers they can hold. Constraint (5.4) limits the number of 20' and 40' containers to their respective limits C_{pl}^{α} where $\mathfrak{L} = \{20, 40\}$.

From the loadlist of the current port and the cargo forecasts, we derive the constants $LD_t^{20\tau}$ and $LD_t^{40\tau}$, holding the number of 20' and 40' containers of each type $\tau \in T$ to stow on each transport $t \in TR$, respectively. These containers are then enforced to be stowed by constraint (5.5).

As described above, for each of the light and heavy types $\tau \in T$, the average weight of all the containers in transport $t \in TR$ is given by the constant $W_t^{\alpha\tau}$. The total weight of a location is then calculated and kept within its maximum limit W_{pl} (reflecting vessel capacities) by constraint (5.6).

Hydrostatic data, specific to each vessel, is used to calculate its trim, draft, buoyancy and stability. Those calculations can be expressed as functions of displacement and center of gravity of the vessel. Due to our assumption about constant displacement (since no ballast can be added), hydrostatic calculations only depend on the center of gravity of the vessel. This allows us to transform the stability, trim and draft restrictions to bounds on the center of gravity. In constraint (5.7), for each port $p \in P$, the center of gravity limits, $G_p^{+\rho}$ and $G_p^{-\rho}$, have been calculated, where $\rho \in \{L, V\}$ represent the longitudinal and vertical components. The center of gravity of each location is defined by the constant G_l^{ρ}. Due to the symmetrical position of the stacks in outer locations, it is possible to distribute the weight of the containers in these stacks such that the vessel is at even keel. For that reason we do not need specific constraints to ensure this.

Given a set of frames $f \in F$, where each frame is a calculation point as described in chapeter 2, constraint (5.8) calculates the downward force created by the cargo aft of each frame f, where L_f^{Aft} is the set of locations aft of frame f, and $\mathcal{S}_{pf}^{+}$ and $\mathcal{S}_{pf}^{-}$ are the maximum and minimum shear limits at frame $f \in F$ for port $p \in P$, respectively. Since the weight of the vessel is constant, we pre-compute the buoyancy forces and thus only restrict the weight distribution over the vessel accordingly.

Constraints (5.9 - 5.14) define the various objective components minimized in (5.1). A major objective is to minimize hatch-overstowage. This is modeled by the binary variables $\delta_{pl} \in \{0, 1\}$, indicating the presence of containers to load or unload at port $p \in P$ under on-deck locations L^O. This is accomplished with constraint (5.9), where L_l^U is the set of locations under $l \in L^O$, TR_p^A is the set of transports that are either loaded or unloaded in port p (we refer to such containers as *active* containers), and R_{pi}^D are the containers already on board the vessel when arriving at the first port (hereafter referred to as the *release*) that are discharged from location i in port p. In constraints (5.9-5.12) we make use of a BigM constant M. In our implementation of the model we tighten these constants to upper bounds of the constraints. Variable y_{pl}^O is defined in (5.10) to be the number of hatch-overstowing containers in on deck location l in port p given the set of transports TR_p^{OV} (and release containers R_{pl}^{OV}) that overstow containers to load or unload in port p.

Overstowage can happen also within locations. One of the major constraints that can force overstowage within a location, is the fact that 20' containers cannot be stowed over 40' containers. Having many 20' containers with earlier discharge port than the 40' containers stowed in a location is thus an undesirable configuration. An estimate of the potential overstowage between 20' and 40' containers within each location is defined by constraints (5.11) and (5.12). Constraint (5.11) introduces a new set of Boolean variables ϕ_{pl} for each port $p \in P$ and location $l \in L$, indicating the presence of 20' containers to load or unload. These indicator variables define the cost variable y_{pl}^P in constraint (5.12). This variable holds the number of potential overstows between 40' and 20' containers within a location. The constants R_{pl}^{D20} and R_{pl}^{OV40} are the number of containers in the release discharged and potentially overstowing in port p, respectively.

Another important objective is optimization of crane utilization. In this work, we define crane utilization as the makespan of the quay side operations, in other words the total amount of time needed for the cranes to perform all the load and discharge operations. Given that any two cranes must work at least two bays apart, we know that crane moves in adjacent bays only can be carried out by a single crane at the time. Thus, a lower bound of the makespan of cranes is the maximum work time for a single crane over the set of moves in adjacent bays, and this is the objective considered in this work. In the crane split example shown in Figure 2.2 this lower-bound would correspond to 123 moves. Constraint (5.13) represents this calculation, where the variable y^T_p defines the lower bound for each port $p \in P$. The index set B represents the set of adjacent bays. L_b is the set of locations of the pair of adjacent bays $b \in B$, R^A_{pl} is the number of active TEU in location l and port p, and C^T is the average time needed to move one TEU by a crane.

During master planning, stacking rules are not taken explicitly into account. For that reason some generated slot planning instances may be infeasible. A possible source of unfeasibility is the reefer containers since they can only be stowed in slots with power plugs, which might be occupied by other containers. Constraint (5.14) alleviates this issue by reducing the maximum capacity of reefer containers within a location by a proportional factor F^{τ}_{pl}, where $F^{\tau}_{pl} = C^R_{pl}/C_{pl}$ for all non-reefer containers and a factor 1 for all reefer containers. The reduction is then captured in the variable y^R_{pl}.

The master planning problem is, due to preference objectives such as reefer capacity reductions, strictly a multi-objective optimization problem. We, however, minimize a weighted sum of the cost variables where the weights $(\mathcal{C}^O, \mathcal{C}^P, \mathcal{C}^R, \mathcal{C}^T)$ correspond to the preferences of our industrial partner and of stowage coordinators.

5.1.1 The Complexity of the Master Planning Problem

From a bin-packing reduction, it is easy to show that slot planning is NP-Hard (e.g. Sciomachen and Tanfani (2003)). The same reduction, however, does not apply to the master planning problem since it abstracts the slots away. Here we show that master planning nevertheless is NP-Hard by a reduction from the Hatch Overstow Problem (HOP). A HOP instance is a tuple $\langle C, in, out, H, r, m, k\rangle$, where C is a set of containers and $c \in C$ is loaded at port $in(c)$ and discharged at port $out(c)$, H is a set of hatch covers where each $h \in H$ must be lifted in the set of ports $r(h)$, and m is the capacity of each stowage area above a hatch cover. The question is whether it is possible to stow all the containers with at most k hatch overstows in the stowage area above the hatches. The HOP is shown to be NP-Complete in Section A.4.

Proposition 1. *Master planning is NP-Hard.*

Proof. We reduce HOP to master planning by creating a vessel with $2|H|$ locations such that for each on deck location there is a corresponding counterpart below deck. The set C of containers are created as 40' reefer containers, where $c \in C$ has load port $in(c)$ and discharge port $out(c)$ for $c \in C$. For each hatch $h \in H$ the forced hatch lifts can be modelled by adding $|r(h)|$ onboard containers in a location associated with h below deck with discharge ports equal to those in $r(h)$. In order to disallow stowing other containers below deck, we make sure that the reefer capacity below deck is zero. All locations will have 40' capacity equal to m, and we increase height, weight, shear and stability limits to infinity. Solving the master planning problem will now give us k' as the minimum number of hatch overstows which will result in a "yes" instance if $k' \leq k$ and in a "no" instance otherwise. Since in this way it is possible in polynomial time to reduce HOP to master planning we have proven that master planning is NP-Hard. $\square$

5.2 Computational Results

We evaluate our approach on a test bed of 30 instances generated from a stowage planning optimization tool used by our industrial partner. An overview of the instance characteristics can be seen in Table 5.1. The instances are real stowage problems that coordinators have solved and thus have very high data quality.

All experiments were run on a Linux machine with two Six Core AMD Opteron processors at 2.0 Ghz and 32 GB of memory. Master planning was implemented in C++ and used CPLEX 12.2 libraries.

The master planning problem has, to the best of our knowledge, not yet been solved to optimality. For that reason we did not expect our IP model to be efficient. The results of our experiments, shown in Table 5.2, are however worth of notice. The table presents the outcome of our model, showing results at 2 and 5 percent gap from the LP bound. Such results are interesting as they are within the range of inaccuracy of the forecasting data wrt. cargo in the down stream ports. Two findings are worth of notice. First the model is able to solve to optimality a total of 7 out of 30 instances within a 5 hour time limit. Second, within a 5 percent LP bound gap, we were able to solve 14 out of the 30 instances in less than 10 minutes. These results show that IP models for master planning can be partially applied, and thus further effort on improving them should be made.

MIP relaxation The results, however interesting, are not strong enough to be of practical use for stowage planning decision support. A much higher rate of success is required. Our proposition is to tackle the weaknesses of the IP model by relaxing the integrality constraint on the decision variables $x_{tl}^{20\tau}$ and $x_{tl}^{40\tau}$ and thus solving a MIP problem. We experimentally evaluate this MIP relaxation by comparing the results to the optimal ones from the IP model.

Table 5.3 presents results for optimal runs and for 2 and 5 percent gap from the LP relaxation of the MIP model. In terms of objective value, MIP and IP solutions are very similar.

In contrast, runtime results are drastically different. It is now possible to generate complete stowage plans for nearly all instances within 10 minutes. Only 4 of the 30 instances could not be solved within this time frame, and 12 could be solved to optimal-

Instances Characteristics										
ID	Vessel		Route						Encoding	
	Cap. (TEU)	Loc.	Ports	Util. (%)	Weight (%)	Loadlist & forecast	Release	Moves	Transports	Bools
1	5802	58	7	58	56	11194	0	8163	16	425
2	9112	100	7	80	65	27648	1515	26801	11	450
3	9112	100	7	80	65	27648	1515	26801	11	450
4	9882	87	7	66	65	14102	1438	16331	14	448
5	5802	58	8	52	21	10268	1545	12988	18	408
6	9078	100	10	65	48	39936	1409	26207	20	571
7	8334	80	11	58	40	14862	2743	26702	37	607
8	2480	40	8	42	38	4444	388	5130	20	249
9	9958	87	8	58	51	15037	1741	16332	19	585
10	3354	47	7	40	36	5717	139	5108	13	267
11	2480	40	6	61	55	5306	191	5937	11	137
12	3354	47	5	36	11	1831	111	3036	7	182
13	8434	100	6	69	58	22127	525	9121	4	350
14	7474	90	6	83	86	16985	1956	23131	8	206
15	7464	90	9	71	48	28436	1754	19824	9	320
16	4988	71	8	74	37	12292	289	10863	19	603
17	7464	90	5	82	68	12297	1816	19183	6	146
18	4434	61	4	68	30	4294	433	7239	6	153
19	4978	70	9	50	57	12651	132	8678	12	646
20	4672	67	7	54	42	10006	182	7491	15	417
21	7170	68	11	81	33	33037	1317	27296	15	387
22	7470	90	7	80	75	20974	1878	24244	9	353
23	5004	70	12	61	48	24093	582	13514	23	549
24	8292	80	10	63	67	23440	3220	18287	26	622
25	2480	40	9	44	59	6117	433	5190	30	353
26	8292	80	6	86	62	18484	1433	19275	13	346
27	2748	44	7	31	52	2548	402	3269	17	261
28	6966	72	7	69	61	20981	648	15606	15	474
29	6966	72	4	47	50	4185	1457	6185	6	120
30	9090	80	11	63	58	31250	2624	19513	20	386

Table 5.1: *Problem instances overview.* Columns under *Vessel* indicate ship dependent data: *Cap.*. is the maximum TEU capacity of the ship and *Loc.* is the total number of locations. Notice that given the same number of locations, different vessels can have different capacities. Columns under *Route* show information about the route, *Ports* indicates the number of calls in the route, *Util.* and *Weight* are the maximum percentage of utilization during the voyage in terms of TEU and weight. *Loadlist* and *Release* are respectively the total number of containers to be loaded in the vessel and the number of containers already onboard in TEUs. *Moves* is the total number of crane moves on the route. The *Encoding* columns present the number of Boolean variables (*Booleans*) needed by the master planning phase after preprocessing, while *Transports* is the total number of active transports.

ity. Experiments have shown nearly no difference in objective value w.r.t. overstowage. Also we see a very small difference in crane utilization, clearly due to the increased flexibility of the decision variables. This important observation can be used to advocate the use of the MIP model in exchange of IP. Notice that the last column of Table 5.3 shows the runtime including the slot planning phase. Part of the success of the MIP master planning model is due to the high efficiency of the slot planning algorithms. As shown in chapter 6, slot plans can be generated in less the one minute for the entier vessel, leaving a large portion of the 10 minutes time-limit to master planning.

IP Results							
	Optimal		2% Gap		5% Gap		
ID	Objective (10^5)	Time (sec.)	Gap. (%)	Time (sec.)	Gap (%)	Time (sec.)	Total (sec.)
1	0.61	*timeout*	-	*timeout*	4.88	17969.09	17970.92
2	7.97	*timeout*	1.9	**227.74**	3.45	**47.56**	**58.78**
3	7.97	*timeout*	1.9	**231.82**	3.45	**47.75**	**58.90**
4	19.13	5430.72	0.79	**112.19**	3.25	**106.60**	**108.87**
5	35.31	10210.36	0.38	**255.10**	0.38	**255.10**	**256.33**
6	13.95	*timeout*	1.42	5511.46	2.2	4533.47	4548.34
7	63.90	*timeout*	1.88	899.84	4.96	**177.22**	**179.22**
8	2.55	8520.11	0.64	**27.15**	0.64	**27.15**	**28.37**
9	73.12	16658.4	0.13	2358.92	4.69	1026.21	1030.25
10	0.69	*timeout*	1.84	11025.54	5	4730.26	4734.02
11	50.49	*timeout*	-	*timeout*	-	*timeout*	*timeout*
12	1.91	*timeout*	1.39	**8.02**	2.31	**6.92**	**19.11**
13	22.62	*timeout*	2	6096.07	4.15	602.4	619.42
14	-	*timeout*	-	*timeout*	-	*timeout*	*timeout*
15	14.82	*timeout*	0.5	**190.53**	0.5	**190.53**	**199.97**
16	5.41	*timeout*	-	*timeout*	-	*timeout*	*timeout*
17	4.66	**4.45**	0.01	**4.45**	0.01	**4.45**	**6.51**
18	2.80	**69.61**	0.56	**6.11**	0.56	**6.11**	**11.13**
19	0.79	*timeout*	-	*timeout*	4.89	16305.32	16311.27
20	7.10	*timeout*	1.96	4229.4	2.69	3345.58	3349.62
21	2.35	*timeout*	1.21	2740.46	2.15	1747.86	1757.61
22	13.38	*timeout*	0.7	2082.79	0.7	2082.79	2087.89
23	41.38	*timeout*	2	8073.32	4.96	1149.13	1154.83
24	322.55	*timeout*	-	*timeout*	-	*timeout*	*timeout*
25	24.16	*timeout*	0.74	3326.15	0.74	3326.15	3328.01
26	4.76	952.36	0.56	**53.21**	0.56	**53.21**	**53.21**
27	0.36	*timeout*	1.9	**260.49**	2.05	**84.01**	**88.01**
28	2.23	*timeout*	-	*timeout*	4	16429.07	16435.85
29	0.53	*timeout*	1.46	**10.89**	3.98	**10.43**	**13.39**
30	76.44	*timeout*	1.99	**284.30**	2.18	**82.99**	**85.33**

Table 5.2: *Master Planning with IP.* The first column is the instance number. The next columns present grouped results of three runs of the model: the first for optimality and the others ending respectively at 2 and 5 percent gap from the LP relaxation. Column *Objective* is the optimal value, and column *Gap* the distance to optimality w.r.t. *Obj.* Times are reported in *Time*, while time to generate a complete stowage plan is shown in column *Total* which includes the runtime of the slot planning phase. Instances that could not be solved within 5 hours are marked with *timeout*. The bold face shows results obtained within 10 min.

This model enables shipping companies to use standard solvers and eases the process of adding side constraints. We also believe the MIP model can be used in practice as a decision support tool.

Further details are shown in Table 5.4, where the experiments between the IP and MIP models at 5% gap are compared. What is important to notice is that the difference in objective value and all of its components is very small between the two models. This further confirms that the MIP model can effectively be used in exchange of the IP model, exchanging very little quality loss for a larger gain in runtime performance.

Figure 5.2 is a high-level visualization of the optimal master planning solution of a small stowage planning problem, in this case instance 18. The figure depicts the status of the vessel upon arrival at each port. The top shows the state of the vessel as it arrives

MIP Results							
	Optimal		2% Gap		5% Gap		
ID	Gap (%)	Time (sec.)	Gap. (%)	Time (sec.)	Gap (%)	Time (sec.)	Total (sec.)
1	-	*timeout*	-	1718.57	0.011	782.93	784.75
2	-	**105.24**	0.012	**22.37**	0.007	**17.99**	**29.90**
3	-	**106.86**	0.012	**22.52**	0.007	**18.17**	**30.08**
4	0.000	**31.25**	0.006	**9.71**	0.032	**9.71**	**13.10**
5	0.000	1074.08	0.003	**47.65**	0.003	**47.65**	**48.79**
6	-	*timeout*	0.004	**346.78**	0.020	**332.46**	**346.28**
7	-	*timeout*	0.008	**143.74**	0.008	**92.19**	**94.95**
8	0.002	**16.91**	0.006	**5.68**	0.006	**5.68**	**7.35**
9	0.000	**107.01**	0.001	**101.28**	0.049	**101.28**	**106.57**
10	-	663.15	0.012	**157.82**	0.025	**41.59**	**44.95**
11	-	*timeout*	-	*timeout*	-	*timeout*	*timeout*
12	-	**2.04**	0.016	**0.65**	0.025	**0.65**	**12.17**
13	-	17098.95	0.009	**137.75**	0.008	**58.08**	**76.92**
14	-	645.51	-	**13.09**	-	**13.09**	**13.09**
15	-	1304.77	0.012	**16.05**	0.012	**16.05**	**24.88**
16	-	15844.06	-	8775.7	-	2665.65	2672.48
17	0.001	**1.77**	0.006	**1.54**	0.006	**1.54**	**4.06**
18	0.001	**1.52**	0.005	**1.12**	0.005	**1.12**	**5.29**
19	-	*timeout*	-	**216.28**	0.009	**81.55**	**87.30**
20	-	9050.47	0.010	**70.43**	0.017	**70.43**	**74.34**
21	-	*timeout*	0.006	**84.36**	0.027	**72.88**	**82.91**
22	-	*timeout*	0.011	**61.80**	0.021	**12.14**	**20.42**
23	-	*timeout*	0.005	1543.14	0.010	**304.03**	**309.97**
24	-	2858.98	-	2328.33	-	1609.04	1615.88
25	-	*timeout*	0.004	**63.63**	0.013	**37.92**	**39.64**
26	0.002	**54.59**	0.004	**8.59**	0.004	**8.59**	**8.59**
27	-	**93.86**	0.000	**4.95**	0.013	**4.40**	**6.52**
28	-	*timeout*	-	4624.16	0.017	1794.5	1799.89
29	-	**1.68**	0.012	**0.95**	0.038	**0.95**	**3.06**
30	-	**102.41**	0.002	**37.54**	0.024	**28.02**	**30.85**

Table 5.3: *Master Planning with MIP.* The second column describes the gap between the IP and MIP optimal solution. For the other columns see Table 5.2. Note that the gap in this table is wrt. the corresponding IP solutions.

at the first port before any load or discharge operations are performed. The left side we see depicts the load of each bay with a color coding indicating the port of discharge. On the right side, each of the vertical lines represents a transport assigned to a location of the associated bay. The beginning of the line indicates the loading port and the end the discharge port. The horizontal gray box between the vessels represents the ports. As illustrated, the crane makespan objective imposes a distribution of containers along the ship to maximize concurrent crane operations. Vessel bays, through out the entier route, also tend to hold most containers with the same port of discharge to minimize overstowage.

	5% IP vs. MIP Results											
	Obj		OV		P.OV.		Makespan		Reefer Prop.		Time	
ID	IP	MIP	IP	MIP	IP	MIP	IP	MIP	IP	MIP	IP	MIP
1	0.61	0.60	0.00	0.00	0.00	0.00	0.61	0.60	0.00	0.00	17969.09	782.93
2	8.14	8.20	0.00	0.00	1.00	1.00	1.94	2.00	5.20	5.20	47.56	17.99
3	8.14	8.20	0.00	0.00	1.00	1.00	1.94	2.00	5.20	5.20	47.75	18.17
4	19.77	19.17	0.00	0.00	18.50	18.00	1.17	1.17	0.10	0.00	106.6	9.71
5	35.43	35.34	34.00	34.00	0.00	0.00	1.43	1.32	0.00	0.02	255.1	47.65
6	14.22	14.50	3.00	3.00	9.00	9.00	1.75	1.70	0.46	0.80	4533.47	332.46
7	66.64	66.10	19.00	17.00	43.00	43.00	1.70	1.68	2.94	4.42	177.22	92.19
8	2.56	2.55	2.00	2.00	0.00	0.00	0.56	0.55	0.00	0.00	27.15	5.68
9	76.71	73.14	10.00	10.00	65.50	62.00	1.21	1.14	0.00	0.00	1026.21	101.28
10	0.71	0.69	0.00	0.00	0.00	0.00	0.71	0.69	0.00	0.00	4730.26	41.59
11	-	-	-	-	-	-	-	-	-	-	-	-
12	1.95	1.90	1.00	1.00	0.00	0.00	0.46	0.45	0.49	0.44	6.92	0.65
13	23.00	23.19	0.00	0.00	18.50	18.50	1.14	1.19	3.36	3.50	602.4	58.08
14	-	6.35	-	4.00	-	0.00	-	2.35	-	0.00	-	13.09
15	14.87	15.05	13.00	13.00	0.00	0.00	1.87	1.98	0.00	0.07	190.53	16.05
16	-	5.39	-	0.00	-	0.00	-	1.15	-	4.24	-	2665.65
17	4.66	4.69	1.00	1.00	1.50	1.50	1.90	1.93	0.27	0.26	4.45	1.54
18	2.82	2.81	0.00	0.00	0.00	0.00	0.61	0.60	2.21	2.21	6.11	1.12
19	0.81	0.80	0.00	0.00	0.00	0.00	0.79	0.80	0.02	0.00	16305.32	81.55
20	7.23	7.11	5.00	5.00	0.00	0.00	0.84	0.81	1.39	1.31	3345.58	70.43
21	2.39	2.45	0.00	0.00	0.00	0.00	2.25	2.29	0.14	0.16	1747.86	72.88
22	13.40	13.69	8.00	8.00	3.00	3.00	2.38	2.42	0.02	0.28	2082.79	12.14
23	42.28	41.85	32.00	32.00	0.00	0.00	1.64	1.46	8.65	8.38	1149.13	304.03
24	-	296.60	-	149.38	-	142.29	-	1.69	-	3.24	-	1609.04
25	24.29	24.61	22.00	22.00	1.50	1.50	0.71	0.72	0.09	0.39	3326.15	37.92
26	4.79	4.77	0.00	0.00	0.00	0.00	1.80	1.84	2.99	2.93	53.21	8.59
27	0.36	0.37	0.00	0.00	0.00	0.00	0.36	0.37	0.00	0.00	84.01	4.4
28	2.26	2.22	0.00	0.00	0.00	0.00	1.71	1.71	0.55	0.51	16429.07	1794.5
29	0.55	0.53	0.00	0.00	0.00	0.00	0.55	0.53	0.00	0.00	10.43	0.95
30	78.03	79.97	69.00	70.00	3.00	3.00	1.51	1.49	4.52	5.48	82.99	28.02

Table 5.4: *Comparison between IP and MIP result at 5% gap.* The experiments are presented for each column in a pair of IP and MIP results. The first group column shows the total objective value, the second the overstowage component, the third the location overstowage, the fourth the makespan, the fifth the reefer proportion component, and at last the runtime.

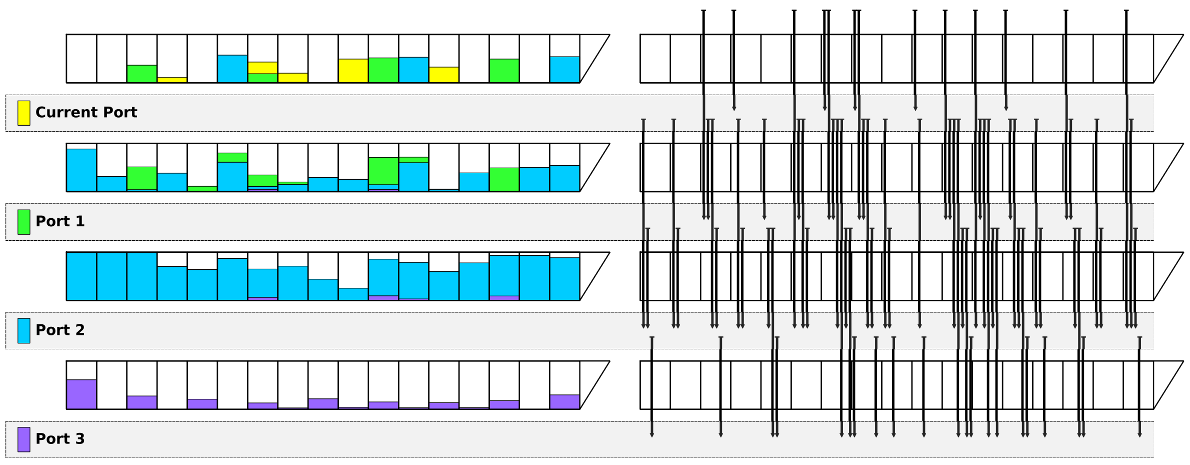


Figure 5.2: Master planning solution of instance 18

5.3 An Improved Stability and Stress Model

Hydrostatic calculations, such as buoyancy, stability, trim and draft restrictions are non-linear functions of the ship's center of gravity and its displacement (the total weight of the loaded vessel). Those can, however, easily be linearized and translated into bounds on the position of the center of gravity when considering a constant displacement, as has been done in previous work and the model presented in the previous section.

In reality, ballast tanks are used by stowage coordinators to better handle the stability of the vessel and allow stowage configurations that are otherwise infeasible. Ignoring ballast water can become a great source of error as it can constitute up to 25% of the ship's displacement. Including tanks in the mathematical models, however, brings forth a number of non-linear constraints due to the now variable vessel displacement.

When variable displacement is taken into account, the above mentioned hydrostatic calculations becomes a function of two variables, the center of gravity and the displacement. The previously trivial linearization now becomes complex and difficult to handle efficiently. The intuition behind this complexity is simple. When the displacement is constant, it is possible to pre-calculate the amount of water the vessel will displace. With variable displacement the amount of displaced water changes, and it does so non-linearly due to the curved shape of the vessel hull.

According to our industrial collaborators, it is possible for stowage coordinators to make an educated guess on the amount of ballast water that a vessel might need within 15% from the actual amount. We use this assumption to define a displacement range within which we are able to define a linearization of the stability contraints with an acceptable error.

5.3.1 Linearization of stability constraints

When ballast tanks are included into the optimization model there are two major non-linearities that need to be dealt with. The first is the calculation of the center of gravity and second is the linearization of the hydrostatic data. Consider the following equation for the calculation of the longitudinal center of gravity (lcg) without ballast tanks:

$$LCG = \frac{LM^o + \sum_{l \in L} G_l^L v_l}{W} \tag{5.15}$$

where LM^o is the constant longitudinal moment of the empty vessel, G_l^L is the longitudinal center of gravity of location $l \in L$, v_l is the weight of location $l \in L$ and W is the displacement. Displacement is given by $W = W^o + \sum_{l \in L} v_l$ where W^o is the constant weight of the empty vessel. Since we load all the containers in the loadlist, the total weight of the locations does not change and we can thus assume W to be constant, which makes equation (5.15) linear. Now let us consider the same equation where we include ballast tanks:

$$LCG = \frac{LM^o + \sum_{l \in L} G_l^L v_l + \sum_{u \in U} G_u^L v_u}{W + \sum_{u \in U} v_u}, \tag{5.16}$$

where U is the set of ballast tanks, G^L_u is their longitudinal center of gravity and v_u is the variable defining the weight of water to be loaded in tank $u \in U$. The displacement of the vessel is now no longer constant, as, in contrast to the cargo, we do not know in advance the amount of water in the tanks. Thus equation (5.16) is non-linear. In order to deal with this, we propose the following approximation:

$$\begin{aligned}
LCG &= \frac{LM^o + \sum_{l\in L} G^L_l v_l + \sum_{u\in U} G^L_u v_u}{W + \sum_{u\in U} v_u} && (5.17)\\
&= \frac{LM^o + \sum_{l\in L} G^L_l v_l + \sum_{u\in U} G^L_u (v_u + \Delta_u)}{W + \sum_{u\in U} v_u + \sum_{u\in U} \Delta_u} && (5.18)\\
&= \frac{LM^o + \sum_{l\in L} G^L_l v_l + \sum_{u\in U} G^L_u (v_u + \Delta_u)}{W + W^T + \sum_{u\in U} \Delta_u} && (5.19)\\
&\approx \frac{LM^o + \sum_{l\in L} G^L_l v_l + \sum_{u\in U} G^L_u (v_u + \Delta_u)}{W + W^T}. && (5.20)
\end{aligned}$$

The idea behind the approximation is to divide the water in the ballast tanks into two parts: the estimated ballast that a stowage coordinator can provide and the error in the estimate. In equation (5.18) we model the estimation error with the variables Δ_u. Given that the water in the tanks can change at most $\sum_{u\in U} \Delta_u$, it is possible to rewrite the equation as in (5.19) where W^T represents the amount of water that remains constant. We then make a linear approximation of the vessel longitudinal center of gravity by removing the allowed changes of ballast water from the denominator of the fraction resulting in equation (5.20). Given the total capacity of the tanks (W^T), the fact that the constant weight of an empty vessel $W^o \approx 2W^T$ and that the weight of the cargo $W^C \approx 6W^T$, we can reasonably assume that the error in the approximation of the longitudinal center of gravity, given that stowage coordinators can estimate the ballast within 15 percent accuracy, is less than $0.15W^T/(2W^T+6W^T+W^T) = 1.7\%$. Note that the same approximation can be used to calculate the vertical and transversal center of gravity.

The assumption that the amount of ballast water lies within a given interval is useful for the linearization of the hydrostatic calculations. Hydrostatic calculations are in practice linear approximations of given data points. When the center of gravity and the displacement of the vessel are known, the linearization is very accurate. For the problem we are going to model, this is, however, not the case since both the center of gravity and the displacement can vary.

Figure 5.3 shows a plot of the hydrostatic data for the trim and metacenter calculation. The functions are clearly non-linear, but it is important to notice that within a small displacement interval it is possible to approximate the functions accurately with a plane. This is only true for displacement levels that are not at the extremes of the data tables, but it is reasonable to assume that the displacement of a stowage plan will be within these extremes.

Given the assumption that the amount of ballast water is only allowed limited changes, it is possible to linearize the hydrostatic calculations with the planes described

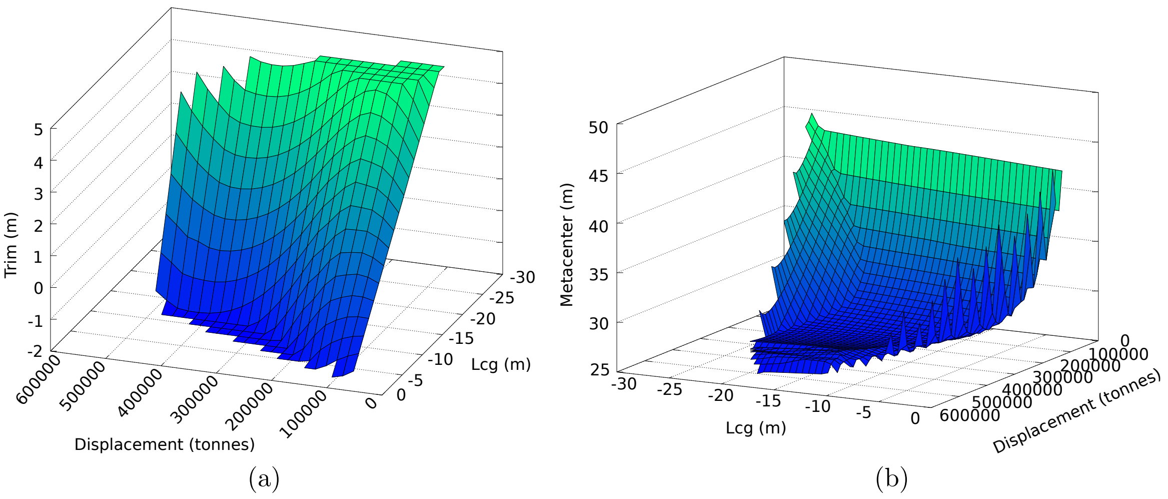


Figure 5.3: (a) Trim as a function of displacement and longitudinal center of gravity (Lcg) (b) Metacenter as a function of displacement and Lcg.

above. Those planes can thus be used in our model. Not all the hydrostatic calculations are, however, as smooth as the ones for trim and metacenter shown in Figure 5.3. The buoyancy of a vessel is the volume of water that the vessel displaces. In order to calculated this volume, it is necessary to know the shape of the vessel hull. For this purpose the hydrostatic data tables provides the possibility of calculating the submerged area of a vessel at a specific point called a *station*. Figure 5.4 shows an example of such areas and how stations are distributed along the vessel.

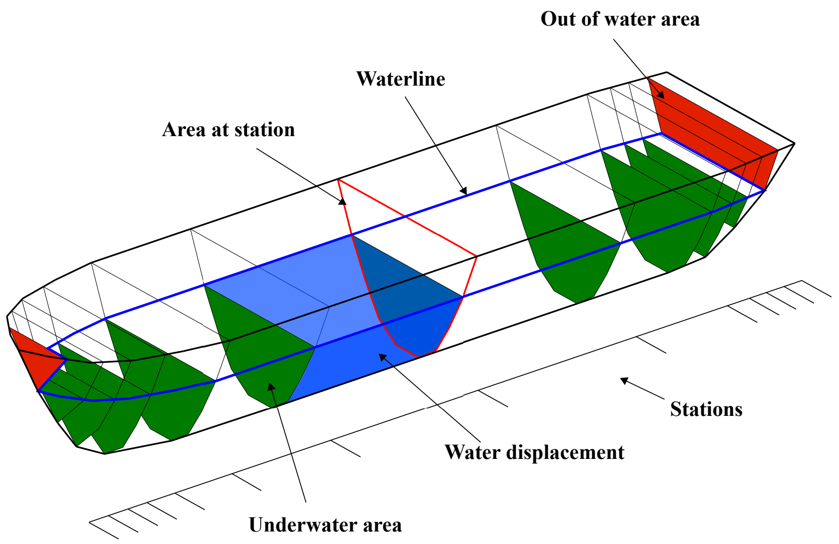
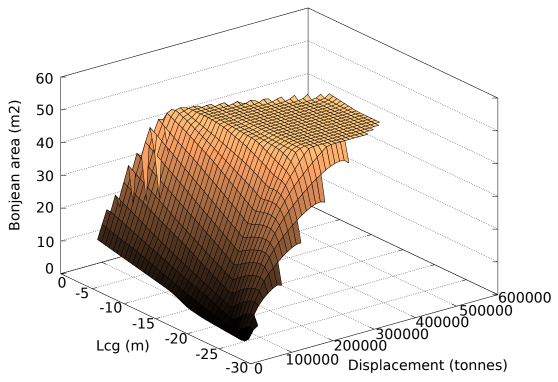
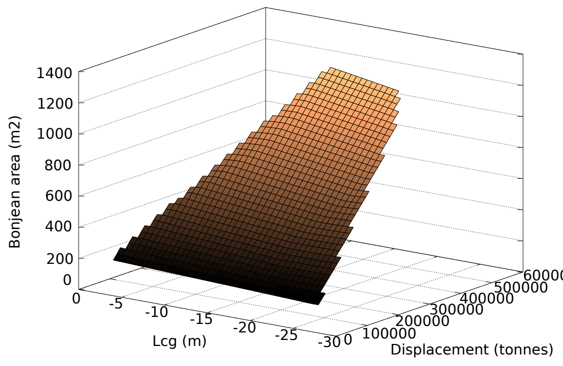


Figure 5.4: Areas for buoyancy calculation and stations distribution

Given two adjacent stations, s_1 and s_2, the buoyancy of the vessel section between the two stations is approximated by $\frac{(A_{s1}+A_{s2})d(s_1,s_2)\delta^W}{2}$, where A_s is the underwater area at station s from now on called *bonjean*, $d(s_1, s_2)$ is the distance between the two stations and δ^W is the density of the water. As Figure 5.4 shows, stations are not evenly distributed along the vessel. A greater concentration is found at the extremities of a vessel where the hull changes the most. Figure 5.5 shows two plots of the hydrostatic data related to the bonjean at one of the first stations and at a station in the middle of a vessel. As expected, the function describing the hull at the bow is highly non-linear since the hull greatly changes, which is not the case for stations in the middle of the vessel. Within specific displacement ranges, however, it is still possible to approximate the function linearly. Should one want to model displacement ranges that include the most non-linear parts, piecewise linear approximations with a few binary variables can be used.



(a)



(b)

Figure 5.5: (a) Underwater area as a function of displacement and longitudinal center of gravity (Lcg) at a bow station (b) Underwater area as a function of displacement and Lcg at a middle station.

5.3.2 A Revised Model with Ballast Tanks

Using the linear approximations described in the previous section, we propose a refined LP model for master planning that includes ballast tank modeling. For the sake of simplicity, the model reuses variables, sets and constants definitions from Section 5.1. Additional constants are defined as needed. We also constrain ourself, without loss of generality, to analyse the model for one port. In that regard, we define $p_0 \in P$ to be the port in question. The objectives from the original master problem (see Section 5.1) are not included in the model under analysis as they do not have any influence on the stability calculations, and thus are irrelevant to this study.

Given the set U of ballast tanks in a vessel, and the decision variable $x_{p_0u} \in \mathbb{R}$ representing the weight of water present in the tanks, we propose the following LP model:

minimize

$$\sum_{u \in U} y_u \tag{5.21}$$

subject to

$$\sum_{t \in TR_{p_0}^{ON}} \sum_{\tau \in T} \left(x_{tl}^{20\tau} + 2x_{tl}^{40\tau} \right) \leq C_{p_0 l} \quad \forall l \in L \tag{5.22}$$

$$\sum_{t \in TR_{p_0}^{ON}} \sum_{\tau \in T} x_{tl}^{\alpha\tau} \leq C_{p_0 l}^{\alpha} \quad \forall l \in L, \alpha \in \{20, 40\} \tag{5.23}$$

$$\sum_{t \in TR_{p_0}^{ON}} \sum_{\tau \in \{RL, RH\}} \left(x_{tl}^{20\tau} + x_{tl}^{40\tau} \right) \leq C_{p_0 l}^{RS} \quad \forall l \in L \tag{5.24}$$

$$\sum_{t \in TR_{p_0}^{ON}} \sum_{\tau \in \{RL, RH\}} \left(0.5x_{tl}^{20\tau} + x_{tl}^{40\tau} \right) \leq C_{p_0 l}^{RC} \quad \forall l \in L \tag{5.25}$$

$$\sum_{t \in TR_{p_0}^{ON}} \sum_{l \in L} x_{tl}^{\alpha\tau} = LD^{\alpha\tau} t \quad \forall \tau \in T, \alpha \in \{20, 40\} \tag{5.26}$$

$$\sum_{t \in TR_{p_0}^{ON}} \sum_{\tau \in T} W_t^{\alpha\tau} x_{tl}^{\alpha\tau} = v_{p_0 l}^{W\alpha} \quad \forall l \in L, \alpha \in \{20, 40\} \tag{5.27}$$

$$v_l^{W\alpha} \leq W_{p_0 l}^{\alpha} \quad \forall l \in L \tag{5.28}$$

$$0.5v_l^{W20} + v_l^{W40} \leq W_{p_0 l}^{40} \forall l \in L \tag{5.29}$$

$$v_l^{W20} + v_l^{W40} = v_{p_0 l}^{W} \quad \forall l \in L \tag{5.30}$$

$$\sum_{u \in U} x_{p_0 u} + \sum_{l \in L} v_{p_o l}^{W} + W_{p_0}^{o} = v_{p_0}^{W} \tag{5.31}$$

$$x_{p_0 u} \leq C_u \quad \forall u \in U \tag{5.32}$$

$$(E_u - \epsilon) \leq x_{p_0 u} \leq (E_u + \epsilon) \quad \forall u \in U \tag{5.33}$$

$$\frac{\sum_{l \in L} G_{p_0 l}^{L} v_{p_0 l}^{W} + \sum_{u \in U} G_{p_0 u}^{L} x_{p_0 u} + LM^{o}}{W} = v_{p_0}^{Lcg} \tag{5.34}$$

$$\frac{\sum_{l \in L} G_{p_0 l}^{V} v_{p_0 l}^{W} + \sum_{u \in U} G_{p_0 u}^{V} x_{p_0 u} + VM^{o}}{W} = v_{p_0}^{Vcg} \tag{5.35}$$

$$\mathcal{L}^{Trim-} \leq A_T^W v_{p_0}^{W} + A_T^{Lcg} v_{p_0}^{Lcg} + A_T \leq \mathcal{L}^{Trim+} \tag{5.36}$$

$$\mathcal{L}^{DraftA-} \leq A_{DA}^{W} v_{p_0}^{W} + A_{DA}^{Lcg} v_{p_0}^{Lcg} + A_{DA} \leq \mathcal{L}^{DraftA+} \tag{5.37}$$

$$A_{DF}^{W} v_{p_0}^{W} + A_{DF}^{Lcg} v_{p_0}^{Lcg} + A_{DF} \leq \mathcal{L}^{DraftF+} \tag{5.38}$$

$$A_M^W v_{p_0}^{W} + A_M^{Lcg} v_{p_0}^{Lcg} + A_M = v_{p_0}^{M} \tag{5.39}$$

$$v_{p_0}^{M} - v_{p_0}^{Vcg} \geq \mathcal{L}^{GM-} \tag{5.40}$$

$$\delta^W D_{(i,j)} \frac{\sum_{s \in \{i,j\}} A_{Bs}^{W} v_{p_0}^{W} + A_{Bs}^{Lcg} v_{p_0}^{Lcg} + A_{Bs}}{2} = v_{p_0(i,j)}^{B} \quad \forall (i,j) \in S \tag{5.41}$$

$$W_f^\alpha + \sum_{l \in L} p_{lf}^\alpha v_{p_0 l}^W + \sum_{u \in U} p_{uf}^\alpha x_{p_0 u} - \sum_{s \in S} p_{sf}^\alpha v_{p_0 f}^B = v_{p_0 f}^{S\alpha} \quad \forall f \in F, \alpha \in \{Aft, Fore\} \tag{5.42}$$

$$M_f^\alpha + \sum_{l \in L} a_{lf}^\alpha p_{lf}^\alpha v_{p_0 l}^W + \sum_{u \in U} a_{uf}^\alpha p_{uf}^\alpha x_{p_0 u} - \sum_{s \in S} a_{sf}^\alpha p_{sf}^\alpha v_{p_0 s}^B = v_{p_0 f}^{B\alpha} \quad \forall f \in F, \alpha \in \{Aft, Fore\} \tag{5.43}$$

$$\mathcal{L}_{p_0 f}^{Shear-} \leq w_f v_{p_0 f}^{sFore} + (1 - w_f) v_{p_0 f}^{SAft} \leq \mathcal{L}_{p_0 f}^{Shear+} \tag{5.44}$$

$$\mathcal{L}_{p_0 f}^{Bending-} \leq w_f v_{p_0 f}^{BFore} + (1 - w_f) v_{p_0 f}^{BAft} \leq \mathcal{L}_{p_0 f}^{Bending+} \tag{5.45}$$

$$E_u - x_u \leq y_u \quad \forall u \in U \tag{5.46}$$

$$x_u - E_u \leq y_u \quad \forall u \in U \tag{5.47}$$

Constraint (5.22), (5.23), (5.24) and (5.26) are equivalent to constraint (5.2), (5.4), (5.3) and (5.5) of the previous model, and are therefore no further discussed.

Once a 20' container is placed on a reefer cell, if that cell has two reefer plugs, the cell-wise capacity is reduced as the extra plug cannot be used for a 40' container. For that reason an extra constraint for the reefer cells capacity ($C_{p_0 l}^{RC}$) is required (constraint (5.25)). Chapter 2 discussed how the weight of 20' and 40' containers differently affect the structure of the vessel. To achieve a more accurate model, we take this information into account with constraint (5.27) that defines the variables $v_{p_0 l}^{W20}$ and $v_{p_0 l}^{W40}$, holding the weight of the 20' and 40' containers in location $l \in L$ respectively. With contraint (5.28) we constraint each of the variables to the respective limits $W_{p_0 l}^{20}$ and $W_{p_0 l}^{40}$. The combined total weight is constrained in (5.29). All stability calculations are based on the weight of containers in a location. Constraint (5.30) defines the auxiliary variable $v_{p_0 l}^W$ representing the weight of containers in location $l \in L$. The displacement (total weight) of the vessel is represented by the auxiliary variable $v_{p_0}^W$ with constraint (5.31) where $W_{p_0}^o$ is the constant weight of the empty vessel and onboard containers. Constraint (5.32) defines the capacity (C_u) of the tanks, while given E_u as the initial condition of the tanks, constraint (5.33) defines the allowed ϵ ballast change. Variable $v_{p_0 l}^{Lcg}$ represents the longitudinal center of gravity of location l and is computed in constraint (5.34) using the approximation defined in (5.20). The constant LM^o is the constant longitudinal moment of the vessel, $G_{p_0 u}^L$ is the longitudinal center of gravity of the tank $u \in U$ and W is the approximated constant displacement. The same approximation is used in constraint (5.35) for the calculation of the vertical center of gravity represented by the variable $v_{p_0}^{Vcg}$. Given the displacement of the vessel $v_{p_0}^W$ and its approximated longitudinal center of gravity $v_{p_0}^{Lcg}$, constraint (5.36) approximates trim with a plane given by the coefficients A_T^W, A_T^{Lcg} and A_T. The calculated trim is then required to be within the limits $\mathcal{L}^{Trim-}$ and $\mathcal{L}^{Trim+}$. Changing the coefficients accordingly, constraint (5.37) and (5.38) approximate the aft and fore draft of the vessel. Both drafts are kept within the maximum limits $\mathcal{L}^{DraftA+}$ and $\mathcal{L}^{DraftF+}$. Due to the propeller it is also necessary to constrain the draft aft to be at least $\mathcal{L}^{DraftA-}$. The metacenter is also calculated using a linear approximation and it is defined in constraint (5.39) by the variable $v_{p_0}^M$. The GM is then calculated in constraint (5.40) and required to be above the limit $\mathcal{L}^{GM}$. The buoyancy of the section of a vessel between two adjacent stations is defined in constraint (5.41) by the variable $v^{B}_{p_o(i,j)}$. Set S is the set of adjacent station pairs (i,j), $D_{(ij)}$ is the distance between the two stations and δ^W is the density of the water. Again, the hydrostatic linearization is used in this calculation. In shear and bending calculations we take into account the forces at play aft or fore of a frame. Since frames do not always coincide with the starting points of tanks, locations or buoyancy stations, it is necessary to know, given a frame $f \in F$, the fraction of weight that needs to be taken into account from a location l, tank u or station s. For this purpose the constant $p^{\mathit{Aft}}_{lf} \in [0,1]$ is used to denote the fraction of cargo to be considered from location $l \in L$ aft of frame $f \in F$ and p^{Fore}_{lf} for the fraction fore of the frame ($p^{\mathit{Aft}}_{uf}, p^{\mathit{Fore}}_{uf}$ for the tanks and $p^{\mathit{Aft}}_{sf}, p^{\mathit{Fore}}_{sf}$ for the buoyancy). Since the shear forces and bending moments are calculated per frame, errors in the linearization accumulate the further away from the calculation frame the weights are. This can become problematic in the case of bending where the forces are multiplied by the arm, substantially increasing the approximation error. Shear and bending calculations can be done for either fore or aft part of a frame. It stands to reason that aft stress calculations are more accurate at the stern of the vessel and fore calculations at the bow. For that reason a more precise modeling of stress forces requires the calculation at both the aft and fore part of a frame where the two resulting stresses are blended such that aft calculations are weighted more at stern and less at bow. Constraint (5.42) calculates the shear forces both aft and fore of each frame $f \in F$ and defines the shear variable $v^{S}_{p_0 f}$, and where $W\alpha_f$ is the constant weight aft or fore of frame f. The final shear calculation where the aft and fore shear are mixed using a scaling factor $w_f \in [0,1]$ (such that it is 1 for the first frame at bow and 0 in the first frame at stern) is kept within the limits $\mathcal{L}^{\mathit{Shear}+}_{p_0 f}$ and $\mathcal{L}^{\mathit{Shear}-}_{p_0 f}$ in constraint (5.44). The same calculation is made for the bending. The bending variable $v^{B\alpha}_{p_0 f}$ is defined in constraint (5.43) where $a^{\alpha}_{lf}, a^{\alpha}_{uf}, a^{\alpha}_{sf}$ are the arm to frame $f \in F$ of location $l \in L$, tank $u \in U$ and buoyancy section $s \in S$ for both the aft and fore calculation. The constant moment of the vessel is given by the constant M^{o}_{f} and bending is kept within the limits $\mathcal{L}^{\mathit{Bending}+}_{p_0 f}$ and $\mathcal{L}^{\mathit{Bending}-}_{p_0 f}$ by constraint (5.45). Constraints (5.46) and (5.47) define the cost variable y_u quantifying the changes in tank configuration from the initial estimate. The approximation error increases with the amount of difference from the estimated amount of water in each tank, thus objective 5.21 minimizes this difference.

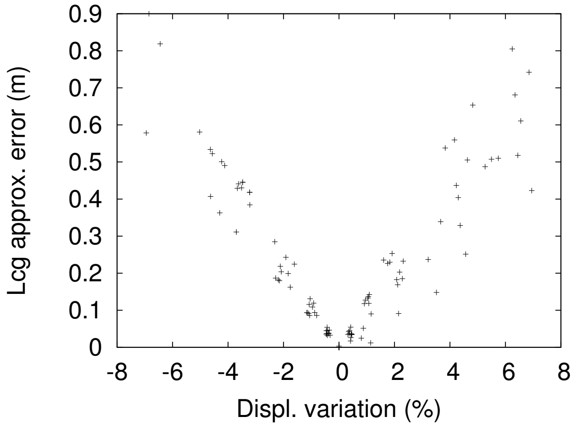
5.3.3 Analysis of Model Accuracy

The improved model for stability and stress calculations, is evaluate experimentally on a large container vessel with a capacity of about 15000 TEUs. This vessel is used as a case study where a set of 10 stowage plans are analysed. The stowage plans are taken directly from plans that have sailed and are thus representative of the loads that can be expected on such vessels at any given port. Table 5.5 gives an overview of the test instance characteristics. In order to evaluate the solutions, we compare with a manual calculation. The manual calculation uses the actual displacement and center of gravity values, and can thus extract accurate information from the hydrostatic data tables.

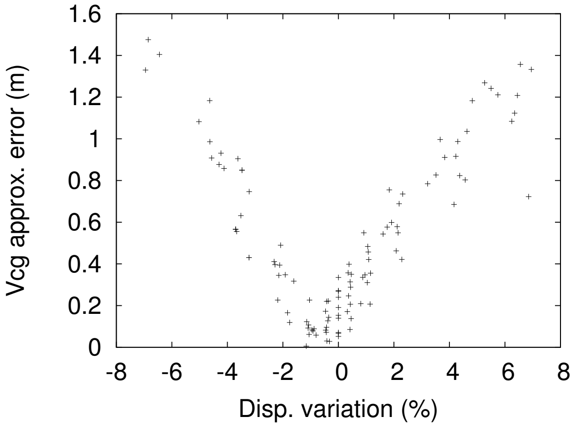
Instances Characteristics								
ID	TEU (%)			Weight (%)			Displacement (tons)	Tanks (tons)
	Total	Release	Load	Total	Release	Load		
1	91.79	39.05	52.74	32.05	13.12	18.93	148520	5160
2	73.83	36.66	37.17	45.43	23.14	22.29	176008	8161
3	59.74	18.45	41.29	42.23	12.07	30.16	168696	6952
4	80.98	28.93	52.06	53.01	19.91	33.10	192286	6962
5	65.65	13.38	52.28	25.76	8.00	17.76	135396	10578
6	48.61	18.01	30.60	24.89	8.83	16.06	133489	10578
7	69.27	28.29	40.98	40.86	17.40	23.46	161492	5199
8	45.81	13.16	32.64	28.84	7.77	21.07	143815	9976
9	58.56	24.89	33.67	30.14	13.78	16.36	141487	5239
10	59.04	20.47	38.57	31.97	9.96	22.01	146120	7341

Table 5.5: *Characteristics of the test instances.* Starting from the left the columns indicate: the ID of the instance, the total utility percentages in terms of TEU capacity used, thereof the percentage of containers in the release and in the loadlist. The next three columns indicate percentages of utilization in terms of weight, in total, for the containers in the release and in the loadlist. The initial displacement and the estimated ballast water are given by the last two columns.

The presented model has been slightly modified to make it was possible to force changes on the amount of ballast water loaded in the tanks. We thus performed experiments allowing different changes in displacement and observed how the accuracy of the model changes. First we consider the linear approximation about the center of gravity of the vessel. We focus on the longitudinal and vertical center of gravity, since our assumption of even transversal weight distribution still holds.



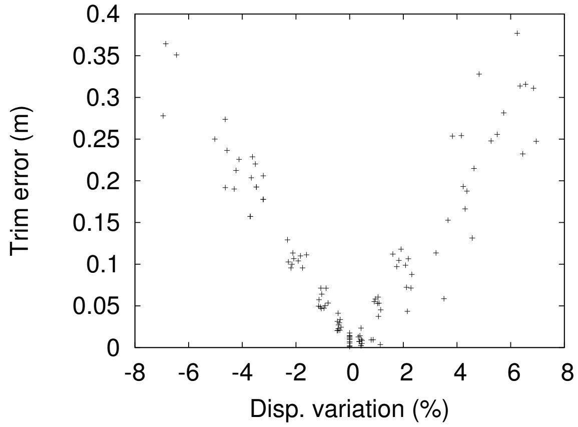
(a)



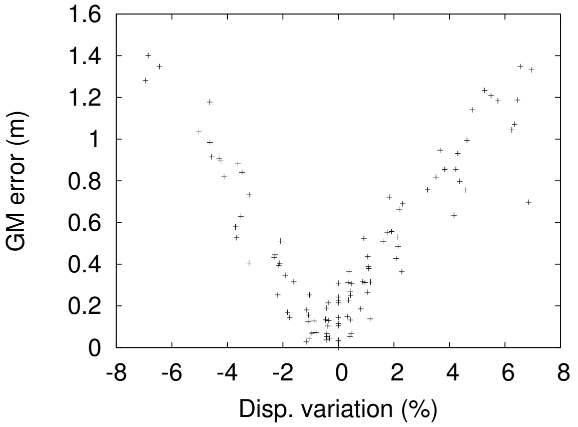
(b)

Figure 5.6: (a) Error in the longitudinal center of gravity (lcg) (b) Error in the vertical center of gravity (vcg).

Figure 5.6 shows two graphs describing how the approximation of the longitudinal (a) and vertical (b) center of gravity behave as the displacement changes. For both graphs, the horizontal axis represents the percentage change of displacement, while the vertical axis represents the error in meters. Each point in the graph is generated by forcing changes in the ballast water of the 10 test instances. In Figure 5.6a it is possible to see, as expected, that when the displacement is unchanged, the value of the longitudinal center of gravity is accurate and the more the displacement moves away from its true value the less accurate the approximation becomes. Note that for a displacement range of 5 percent, the calculation inaccuracy is at most 0.3 meters and thus still very accurate for practical usage. The calculations for the vertical center of gravity are not as accurate (Figure 5.6b). Within the 5 percent range, the linearized value is, however, at most 0.8 meters from the correct one. This was an expected result, as it is not possible to precisely estimate the vertical center of gravity of, for example, locations since we do not know where the containers will be stowed. In order to keep the center of gravity constant, its initial position must be estimated, which in this case is the center of a location. This is not the case for the longitudinal center of gravity since the size of the locations does not allow it to change much. The accuracy of the vertical center of gravity is, however, still acceptable.



(a)



(b)

Figure 5.7: (a) Error for the trim (b) Error for the GM.

Now that we have shown that linearizations for the center of gravity are accurate, we focus on analyzing the accuracy of the hydrostatic data linearization. In Figure 5.7 we use the same graph as before with the horizontal axis describing the percentage displacement changes and the vertical axis the calculation error. Figure 5.7a represents the error for the trim which, as can be seen, is very small. Within a 5 percent displacement range, the error is at most 15 centimeters. Figure 5.7b shows the same analysis for the calculation of GM. Notice that both for the trim and GM calculations, an error is present even when the displacement is unchanged. The error we see in this area of the graph is due to the linearization of the hydrostatic functions. For GM it also includes the approximation error of the vertical center of gravity.

We now move our focus to the linearization of the bonjean areas which we expect to be the most inaccurate. Figure 5.8a shows the same analysis we have done so far for the bonjean areas. The graph shows the maximum error over all bonjean areas as a function of the displacement changes. As can be seen, the variation in displacement is not the main source of error. Most of the inaccuracy is due to the linearization of the hydrostatic data. Figure 5.8b shows how the bonjean error is concentrated at the extremities of the vessel where the hull changes most. The horizontal axis represents the position of the station on the vessel (where 0 is at bow) and the vertical axis is the bonjean error. Worthy of note is the fact that the largest errors are found for stations at the bow. This can be explained by the fact that the range of drafts in our test data forces the linearization of these bonjean areas to be right by the inflection point of the hydrostatic curve (see figure 5.5a) where the linearization is most inaccurate. Better approximations can then be expected for larger drafts, as it is the case for the bonjean of the stations at the stern. The inaccuracy of the bonjean is, however, still quite small if we consider that in the worst case, there is an error of only 4 square meters over an area of over 111 square meters.

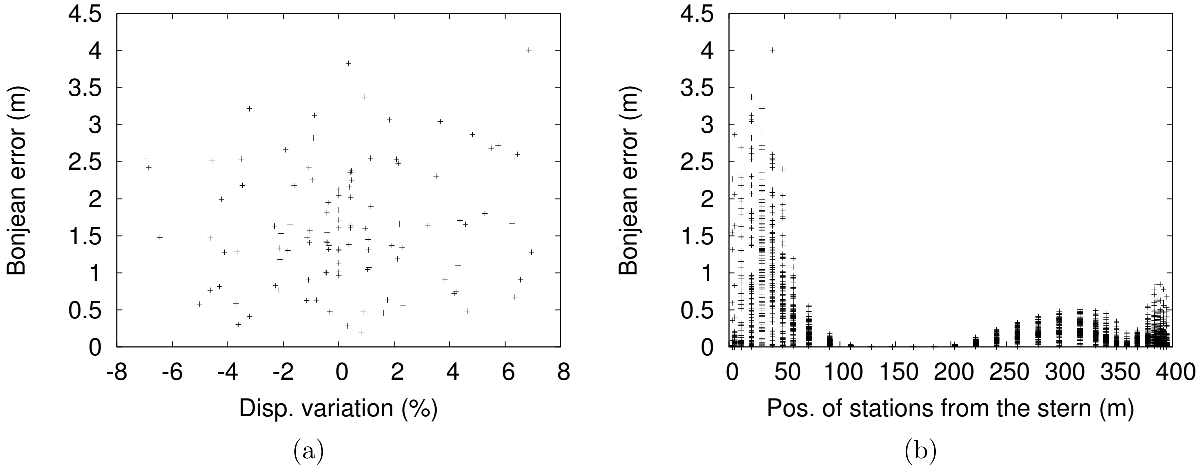


Figure 5.8: (a) Approximation error of the total bonjean as a function of displacement change (b) Approximation error of the bonjean areas per station.

Shear forces and bending moment calculations depend on the buoyancy of the vessel, which is in turn calculated using the bonjean approximations. Figure 5.9a shows, in the same way as the other graphs, how the percentage error in the shear calculation (the vertical axis) behaves as a function of the variation of the displacement (the horizontal axis). As expected, the dominant error is not the approximation of the center of gravity of the vessel since the inaccuracy is more or less the same, independently of how much the displacement changes. A more tight relation can be seen when the shear calculation is related to the error in the bonjean linearization. Figure 5.9b shows the percentage error of the shear force calculation as a function of the total error in bonjean area from the hydrostatic linearization. As depicted, the error in the shear calculation increases with the error in the bonjean linearization. The graph also groups the data points according to their displacement range, and for the data points with no displacement change we can see that the tendency remains the same. One must also take into account that the linearization error of the bonjean is amplified in the shear foces calculation by the fact that is becomes accumulated in the summation of the forces. This particular information is very important when analyzing the error in the bending moment calculation, since this accumulated error is multiplied by the arm of the moment and thus multiplying its impact.

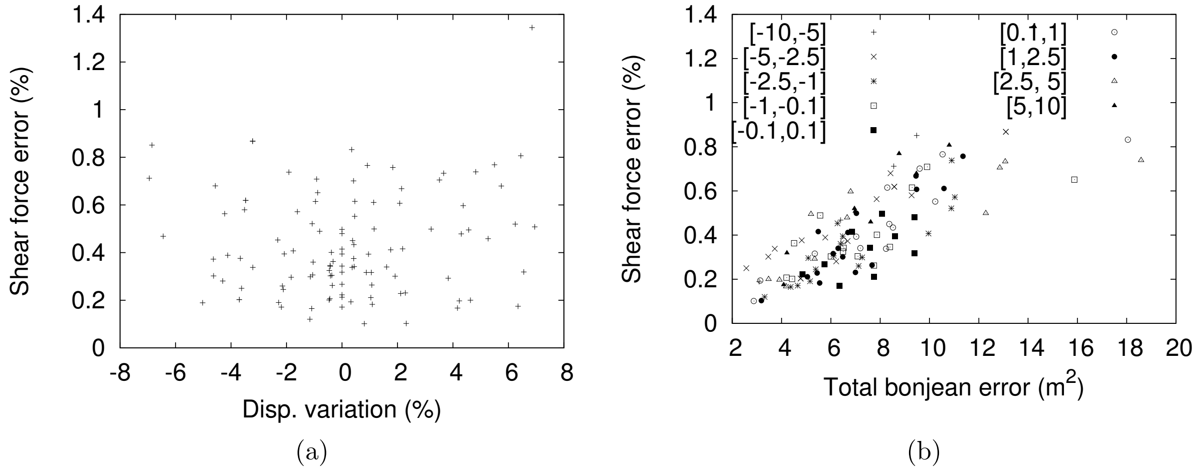
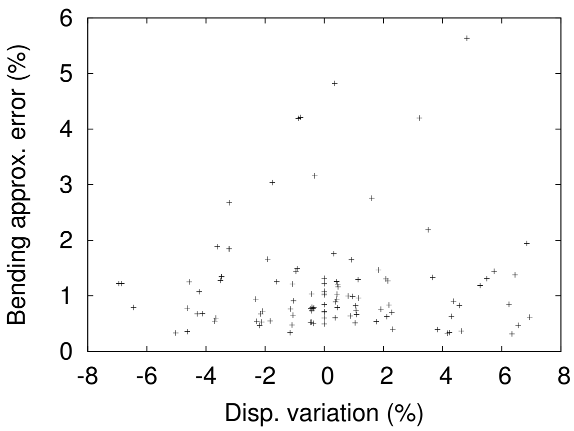
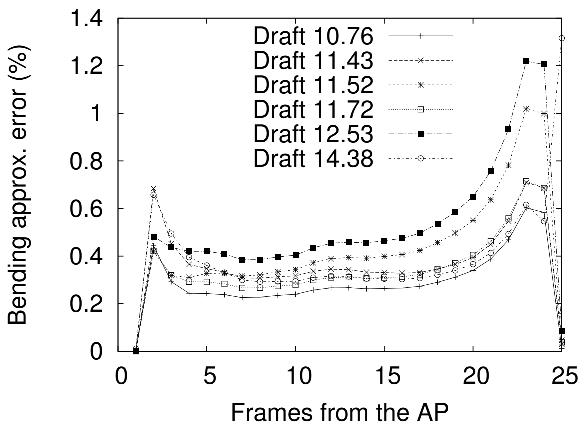


Figure 5.9: (a) Approximation error of the shear forces as a function of displacement change (b) Approximation error of the shear forces as a function of the total bonjean error.

Figure 5.10a shows the error in the bending calculation as a function of the changes in displacement. As expected, like for the shear force calculation, the main source of error is not the approximation of the longitudinal center of gravity, but rather the error in the linearization of the bonjean areas. Due to the fact that the bonjean error is amplified by the multiplication of the arm, it is necessary to look at the error at each calculation frame in order to see how the error of the bending moment changes. We do this by forcing the displacement to remain constant, thus removing the error of the longitudinal center of gravity approximation, and analyzing how the bending error changes at each calculation frame as the draft of each of the 10 test instances changes. The result is shown in Figure 5.10b, where the horizontal axis represents the frames of the vessel and the vertical axis is the percentage of error in the bending moment calculation. Each of the lines plotted in the graph represent one of the 10 test instances each of which has a different draft. As expected, the bending moment is less accurate at the bow and stern due to the imprecision in the bonjean calculations. One more thing worth of notice is that there is no direct relation between the bending error and



(a)



(b)

Figure 5.10: (a) Approximation error of the bending moment as a function of displacement change (b) Approximation error of the bending moment at each bonjean station.

the draft of the vessel. This is due to the non-linear shape of the hull. The bending error for constant displacement does not exceed 1.4 percent. However, when variable displacement is considered an error of up to 3.5 percent might be reached within a 5 percent displacement variation. We consider these approximations acceptable. Higher accuracy can be achieved by reducing the linearization error of the hydrostatic functions for bonjean.

Chapter 6

Slot Planning

The slot planning phase refines the plan generated by the master planning phase by assigning the containers associated with each location to a specific slot in the location. It is during this phase that container stacking constraints are handled. The abstracted decisions taken at the master planning level might, however, not always be feasible. Slot planning must be able to handle these situations. The objectives in master planning, minimization of overstowage and minimization of crane makespan, produce master plans where the average number of discharge ports per location is close to one. This becomes a feature of the input of the slot planning instances, which in comparison to the general problem complexity are much simpler, as far as overstowage is concerned. We thus expect heuristic solutions to achieve high quality solutions fast. As previously mentioned (chapter 5), such efficiency in runtime allows us to spend most of the time in the harder master planning phase.

The slot planning phase uses the following simple algorithm:

1. The container to slot assignment for a specific location is solved using the constraint programming model of Delgado et al. (2012).
2. If no solution is found within a given time limit, the problem is solved using a CBLS algorithm.
3. If no feasible solution is found, a heuristic procedure drops containers until a feasible solution is found.
   (a) The feasible solution is optimized using the same CBLS algorithm.

The CBLS algorithm and the procedure to handle infeasible instances are part of the research behind this thesis and are described in detail in the following sections.

6.1 Slot Planning Problem

The slot planing problem solved in this thesis includes both 20' and 40' containers. As an example of containers with special requirements, it considers reefer and high-cube containers. The problem requires proper handling of stacking rules between the containers and the fulfillment of stack weight and height limits. Taking into consideration the containers already onboard the vessel and stowed in the location, the objective is to minimize overstowage, cluster containers with the same discharge port in the same stack, free as many stacks as possible and minimize the number of non-reefer containers stowed in reefer slots. Compared to the representative problem described in chapter 4, wind-stacking rules are the only missing requirement if it is assumed that stability and stresses are handled by the master planning phase.

Formally, let $\mathcal{S}$ be a set of stacks in a location, $\mathcal{T}_s$ be a set of tiers for each stack $s \in \mathcal{S}$ and let $\mathcal{P} = \{1, 2\}$ be the set describing the aft=1 and fore=2 position of a slot. Given the set $\mathcal{C}$ of all containers to be stowed in the location, we define the decision variable $x_{stp} \in \mathcal{C} \cup \{\perp\}$ to be the assignment of container $c \in \mathcal{C}$ to the slot in stack $s \in \mathcal{S}$, tier $t \in \mathcal{T}_s$ and position $p \in \mathcal{P}$, or the empty assignment $\perp$. An overview of the sets, constants and functions used in this section is given in Table 6.1.

Sets	
$\mathcal{S} \in \{1, ..., N^S\}$	The index set of stacks in the location, where N^S is the number of stacks.
$\mathcal{T}_s \in \{1, ..., N^T_s\}$	The index set of tiers for stack s, where N^T_s is maximum the number of tiers in stack $s \in \mathcal{S}$.
$\mathcal{P} \in \{1, 2\}$	The index set representation of the aft ($p = 1$) and fore ($p = 2$) position of a cell.
$\mathcal{C} \in \{1, ..., N^C\}$	The index set of containers to stow in the location, where N^C is the number of containers.
$\mathcal{C}^P \subset \mathcal{C}$	The subset of containers in the release .
Constants	
$\perp$	The empty assignment.
$A^{20}_{stp} \in \mathbb{B}$	True iff the slot in stack $s \in \mathcal{S}$, tier $t \in \mathcal{T}_s$ and position $p \in \mathcal{P}$ can hold a 20' container.
$A^{40}_{st} \in \mathbb{B}$	True iff the cell in stack $s \in \mathcal{S}$ and tier $t \in \mathcal{T}_s$ can hold a 40' container.
$W_s \in \mathbb{R}_+$	The maximum weight of the stack $s \in \mathcal{S}$.
$H_s \in \mathbb{R}_+$	The maximum height of the stack $s \in \mathcal{S}$.
Attribute functions	
$w(c) : \mathcal{C} \cup \{\perp\} \mapsto \mathbb{R}_+$	The weight of the container $c \in \mathcal{C}$ or 0 if $c = \perp$.
$h(c) : \mathcal{C} \cup \{\perp\} \mapsto \mathbb{R}_+$	The height of the container $c \in \mathcal{C}$ or 0 if $c = \perp$.
$r(c) : \mathcal{C} \cup \{\perp\} \mapsto \mathbb{B}$	True iff the container $c \in \mathcal{C}$ is a reefer.
$\perp(c) : \mathcal{C} \cup \{\perp\} \mapsto \mathbb{B}$	True iff the container $c = \perp$.
$d(c) : \mathcal{C} \mapsto \mathbb{N}$	The discharge port of the container $c \in \mathcal{C}$.
$f(c) : \mathcal{C} \cup \{\perp\} \mapsto \mathbb{B}$	True iff the container $c \in \mathcal{C}$ is a 40' container.
$t(c) : \mathcal{C} \cup \{\perp\} \mapsto \mathbb{B}$	True iff the container $c \in \mathcal{C}$ is a 20' container.

Table 6.1: Constants, attribute functions, and variables of the slot planning model.

We propose a logic based model. A feasible solution to the proposed slot planning problem fulfills the following constraints:

$$\forall s \in \mathcal{S}, t \in \mathcal{T}_s \,.\, \neg f(x_{st1}) \wedge (f(x_{st2}) \Rightarrow \bot(x_{st1})) \tag{6.1}$$

$$\forall s \in \mathcal{S}, t \in \mathcal{T}_s \setminus \{1\}, p \in \mathcal{P} \,.\, \neg\bot(x_{stp}) \Rightarrow \big(t(x_{s(t-1)1}) \wedge t(x_{s(t-1)2})\big) \vee f(x_{s(t-1)1}) \tag{6.2}$$

$$\forall s \in \mathcal{S}, t \in \mathcal{T}_s, p \in \mathcal{P} \,.\, t(x_{stp}) \Rightarrow A^{20}_{stp} \tag{6.3}$$

$$\forall s \in \mathcal{S}, t \in \mathcal{T}_s \,.\, f(x_{st1}) \Rightarrow A^{40}_{st} \tag{6.4}$$

$$\forall s \in \mathcal{S}, t \in \mathcal{T}_s \setminus \{N^T_s\}, p \in \mathcal{P} \,.\, f(x_{st1}) \Rightarrow \neg t(x_{s(t+1)p}) \tag{6.5}$$

$$\forall c \in \mathcal{C} \,.\, |\{x_{stp} = c \,|\, s \in \mathcal{S}, t \in \mathcal{T}_s, p \in \mathcal{P}\}| = 1 \tag{6.6}$$

$$\forall c \in \mathcal{C}^P \,.\, x_{s_c t_c p_c} = c \tag{6.7}$$

$$\forall s \in \mathcal{S}, t \in \mathcal{T}_s, p \in \mathcal{P} \,.\, r(x_{stp}) \wedge t(x_{stp}) \Rightarrow A^R_{stp} \tag{6.8}$$

$$\forall s \in \mathcal{S}, t \in \mathcal{T}_s \,.\, r(x_{st1}) \wedge f(x_{st1}) \Rightarrow A^R_{st1} \vee A^R_{st2} \tag{6.9}$$

$$\forall s \in \mathcal{S} \,.\, \sum_{t \in \mathcal{T}_s} (w(x_{st1}) + w(x_{st2})) \leq W_s \tag{6.10}$$

$$\forall s \in \mathcal{S} \,.\, \sum_{t \in \mathcal{T}_s} \max(h(x_{st1}), h(x_{st2})) \leq H_s \tag{6.11}$$

The assignment convention for the 40’ containers is maintained by constraint (6.1), where $f(c)$ indicates if container $c \in \mathcal{C} \cup \{\bot\}$ is a 40’ container and $\bot(c)$ indicates it is an empty assignment ($c = \bot$). Physical support from below is guaranteed to all containers via constraint (6.2), where $t(c)$ is true if container $c \in \mathcal{C} \cup \{\bot\}$ is a 20’ container. Cell capacities for 20’ and 40’ containers are fulfilled by constraint (6.3) and (6.4), where the constant A^{20}_{stp} is the 20’capacity of a slot and A^{40}_{st} is the 40’ capacity of a cell. Stacking rules between 20’ and 40’ containers are modelled by constraint (6.5), where N^T_s is the index of the top tier in stack $s \in \mathcal{S}$. Each container is assigned to exactly one slot (6.6). Given a set of release containers $\mathcal{C}^P$ constraint (6.7) ensures the original assignment (s_c, t_c, p_c) is maintained.

Reefer slots are indicated with the constant $A^R_{stp} = \mathit{true}$, and 20’ reefer containers are forced to be stowed in those slots by constraint (6.8), while (6.9) ensures that 40’ reefer containers are stowed in cells where either one of the two slots has a power-plug. Reefer containers are identified by the function $r(c)$. Stacks weights and heights are maintained within limits by constraints (6.10) and (6.11), where $w(c)$ and $h(c)$ are respectively the weight and height of container $c \in \mathcal{C} \cup \{\bot\}$. Stack weight and height limits are defined by the constants W_s and H_s

Two of the objective modelled in the slot planning phase are known from the literature, namely overstowage minimization and the clustering of containers with the same discharge port on one stack. The other objectives have not been modelled before and are aimed to generating a robust plan. The first of these attempts at leaving as many stacks free of containers as possible, thus generating space in the location for the next ports. The second, reduces the number of non-reefer containers stowed in reefer slots since those are scarce. Following is the modeling of the slot planning objective components:

$$o_{os} = \sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}_s}\sum_{p\in\mathcal{P}} o_{stp} \tag{6.12}$$

$$o_{ur} = \sum_{s\in\mathcal{S}}\sum_{t\in\mathcal{T}_s}\sum_{p\in\mathcal{P}} ur_{stp} \tag{6.13}$$

$$o_{ps} = \sum_{s\in\mathcal{S}} |\, \{d(x_{stp}) \,|\, t\in\mathcal{T}_s, p\in\mathcal{P}, \neg\bot(x_{stp})\}\,| \tag{6.14}$$

$$o_{us} = \sum_{s\in\mathcal{S}} us_s \tag{6.15}$$

In (6.12) one unit cost is counted for each container that is overstowing another one below in the stack, where o_{stp} defines whether a container $c \in \mathcal{C}$ in stack $s \in \mathcal{S}$, tier $t \in \mathcal{T}_s$ and slot position $p \in \mathcal{P}$ overstows another container in the stack. Thus, $o_{stp} = 1$ if $\neg\bot(x_{stp})$ and there exists a tier $t' \in \{t_s, ..., t-1\}$ below t with an overstowed container $d(x_{st'p}) < d(x_{stp})$ ($d(c)$ return the discharge port of container c). Otherwise $o_{stp} = 0$. Objective (6.13) counts one unit cost for each misused reefer slot, where $ur_{stp} = 1$ if A^R_{stp} and $(f(x_{st1}) \wedge \neg r(x_{st1}) \vee t(x_{stp}) \wedge \neg r(x_{stp}))$, and 0 otherwise. Notice that a 40' non-reefer container will add a unit cost for each reefer slot it covers. In order to favor stacks stowing containers with the same port of destination, one unit cost is counted for each discharge port present in the stack (6.14). In order to minimize the number of used stacks, one unit cost for each stack used is counted by (6.15) where $us_s = 1$ if there exists a $t \in \mathcal{T}_s$ and $p \in \mathcal{P}$ such that $\neg\bot(x_{stp})$, and 0 otherwise.

An optimal solution to a slot planning problem optimizes the following weighted sum:

$$\min \quad C^O o_{os} + C^C o_{ps} + C^S o_{us} + C^R o_{ur} \tag{6.16}$$

where the cost weights C^O, C^C, C^S and C^R represent the priority given to each objective by the stowage coordinators.

6.2 Constraint-Based Local Search (CBLS)

CBLS is an architecture for combinatorial local search algorithms based on the concept of constraints and objectives. The constraint satisfaction part of a combinatorial optimization problem is transformed into an optimization problem where the objective is to minimize constraint violations.

At the core of the architecture we find the concept of *invariants* or one-way constraints. Invariants are represented by *incremental variables* which express a relationship that must be maintained once a new assignment is made to the decision variables. Consider the following example:

$$v = \sum_{i=1}^{n} x_i \tag{6.17}$$

where x_i is a decision variable and v is an incremental variable. Expresion (6.17) is an invariant, and each time a variable x_i assumes a new value, the incremental variable v must be updated accordingly.

Once invariants are available it is natural to create compound objects that maintain properties incrementally. Those objects are called *differentiable objects*. An examples of those are constraints and objectives. Differentiable objects maintain properties such as satisfiability, violations and the contribution of each variable to the violation. Complex functions and evaluations to differentiate similar solutions can also be maintained.

In order to incrementally maintain the system, it is enough to implement incremental algorithm for the basic invariants based on the neighborhood operator of choice.

It is now possible to define formal models of combinatorial problems that make use of the invariant and differentiable objects. Such a model then provides the means to define search algorithms that use the available information, such as constraints violations, for heuristic decisions. This way it is effectively possible to divide the modeling and search parts of local search algorithms.

We refer the reader to Hentenryck and Michel (2009) for a in-depth description of the architecture and the concepts behind CBLS.

6.3 A CBLS for Slot Planning

The CBLS algorithm proposed for slot planning is a classical hill-climbing search using the *min-conflict* heuristic Minton et al. (1992). The initial solution is given by a placement heuristic that relaxes the weight and height constraint to find a initial container arrangement which minimises the objective function. After the initial configuration is found, the algorithm diverges from its classical definition by first focusing on finding a feasible solution with respect to the constraints. Once such a solution is found, it is possible to constrain the neighborhood function to remain within the feasible region while optimizing the objectives.

The aim is to gradually reduce the complexity of the problem in such a way that a feasible solution is found fast allowing the algorithm to use more time searching for an optimal solution. In order for the algorithms to pursue different search paths, randomization is used whenever possible and a race beween parallel searches is performed.

The CBLS algorithm uses a neighborhood generated by swapping containers within the location. A swap is an exchange of some containers between a pair of cells. Formally, a swap γ is a pair of tuples $\gamma = (\langle s, t, c\rangle, \langle s', t', c'\rangle)$ where the containers c in the cell at stack s and tier t exchange position with the containers c' in the cell at stack s' and tier t'. The sets c and c' can contain at most two containers. Swaps are implemented with two functions $swap^{20}$ for exchanging position of 20' containers, and $swap^{40}$ for exchanging position of 40' containers.

- $swap^{20}(x_{stp}, x_{s't'p'})$ where $p \neq p'$ if $(s,t) = (s',t')$, swaps the value of x_{stp} and $x_{s't'p'}$ and covers the following swap types 1) $20' \leftrightarrow 20'$ and 2) $\perp \leftrightarrow 20'$.

- *swap*^{40} $((x_{st1}, x_{st2}), (x_{s't'1}, x_{s't'2}))$ where $(s,t) \neq (s',t')$, pairwise swaps the values of the cell variables (x_{st1}, x_{st2}) and $(x_{s't'1}, x_{s't'2})$ and covers the following swap types 1) $(40', \perp) \leftrightarrow (40', \perp)$, 2) $(\perp, 20') \leftrightarrow (40', \perp)$, 3) $(20', \perp) \leftrightarrow (40', \perp)$, 4) $(20', 20') \leftrightarrow (40', \perp)$ and, 5) $(\perp, \perp) \leftrightarrow (40', \perp)$.

Proposition 2. *The swap neighborhood Γ defined by the union of swap^{20} and swap^{40} swaps for a slot planning problem Π is complete.*

Proof in Appendix C.

6.3.1 Constraint Violations

In CBLS it is common to evaluate and represent a constraint in terms of the set of variables that violate it. When the violation of a constraint is 0 the constraint is satisfied. When a variable violates a constraint, its violation degree is either fixed (e.g, 1) or it reflects to which extent the variable breaks the constraint. This representation is ideal when using the min-conflict heuristic, since it makes it possible to identify which variable violates most constraints and to what degree.

Let π_{stp} denote the value of variable x_{stp} in assignment π. Each constraint and objective is then defined in terms of a function $\sigma(\pi)$ on the assignment π of variables over the violations of each slot variable defined by $\nu(\pi, s, t, p)$, where s is the stack, t is the tier, and p is the position of the slot. To ease the readability, we often interpret the Boolean values *false* and *true* as the numerical values 0 and 1, respectively.

The 20' and 40' capacity constraint (6.3) and (6.4) are represented by

$$\nu_1(\pi, s, t, p) = \neg(t(\pi_{stp}) \Rightarrow A^{20}_{stp})$$
$$\sigma_1(\pi) = \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}_s} \sum_{p \in \mathcal{P}} \nu_1(\pi, s, t, p)$$

and

$$\nu_2(\pi, s, t, p) = \neg(f(\pi_{stp}) \Rightarrow A^{40}_{st})$$
$$\sigma_2(\pi) = \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}_s} \sum_{p \in \mathcal{P}} \nu_1(\pi, s, t, p).$$

The "no hanging containers" constraint for 20' and 40' containers (6.2) is represented by

$$\nu_3(\pi, s, t, p) = \sum_{t'=t_s}^{t-1} \neg\left(\neg\perp(\pi_{stp}) \Rightarrow (t(\pi_{st'1}) \wedge t(\pi_{st'2})) \vee f(\pi_{st'1})\right)$$
$$\sigma_3(\pi) = \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}_s \setminus \{t_s\}} \sum_{p \in \mathcal{P}} \nu_3(\pi, s, t, p).$$

Thus, for some stowed containers, the degree of violation is defined as the number of slots in cells below with insufficient support. The reefer constraints (6.8) and (6.9) are represented by

$$\begin{aligned}\nu_4(\pi, s, t, p) &= \neg \left(r(\pi_{xtp}) \Rightarrow A^R_{stp} \vee \left(f(\pi_{stp}) \wedge (A^R_{st1} \vee A^R_{st2}) \right) \right) \\ \sigma_4(\pi) &= \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}_s} \sum_{p \in \mathcal{P}} \nu_4(\pi, s, t, p).\end{aligned}$$

The maximum stack height constraint (6.11) and the maximum stack weight constraint (6.10) are represented by

$$\begin{aligned}\nu_5(\pi, s, t, p) &= \begin{cases} max\left(0, \frac{\vartheta^{H\pi}_s - H_s}{|\vartheta^{C\pi}_s|}\right) & : \neg\bot(\pi_{stp}) \\ 0 & : \text{ otherwise} \end{cases} \\ \sigma_5(\pi) &= \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}_s} max\left(\nu_5(\pi, s, t, 1), \nu_5(\pi, s, t, 2)\right)\end{aligned}$$

$$\begin{aligned}\nu_6(\pi, s, t, p) &= \begin{cases} max\left(0, \frac{\vartheta^{W\pi}_s - W_s}{|\vartheta^{C\pi}_s|}\right) & : \neg\bot(\pi_{stp}) \\ 0 & : \text{ otherwise} \end{cases} \\ \sigma_6 &= \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}_s} \sum_{p \in \mathcal{P}} \nu_6(\pi, s, t, p).\end{aligned}$$

where $\vartheta^{W\pi}_s = \sum_{t \in \mathcal{T}_s} \sum_{p \in \mathcal{P}} w\left(\pi_{stp}\right)$ is the weight, $\vartheta^{H\pi}_s = \sum_{t \in \mathcal{T}_S} max\left(h\left(\pi_{st1}\right), h\left(\pi_{st2}\right)\right)$ is the height of stack s and $\vartheta^{C\pi}_s = \sum_{t \in \mathcal{T}_s} \sum_{p \in \mathcal{P}} \neg\bot(\pi_{stp})$ is the current number of containers stowed in stack s for assignment π. For these constraints, the violation degree is distributed equally over variables holding containers since it is not possible to identify which one of them will have the largest influence. Finally, the no 20' over 40' container constraint (6.5) is represented by

$$\nu_7(\pi, s, t, p) = \sum_{t'=t_s}^{t-1} \neg \left(t(\pi_{stp}) \Rightarrow \neg f(\pi_{st'1}) \right)$$

$$\sigma_7 = \sum_{s \in \mathcal{S}} \sum_{t=t_s+1}^{N^T} \sum_{p \in \mathcal{P}} \nu_7(\pi, s, t, p).$$

Similar to ν_3, the violation degree of a 20' container is the number of 40' containers stored below.

The remaining constraints are implicitly satisfied by the algorithm by first assigning release containers to their given position, and by heuristically placing the remainder of the containers to variables at the beginning of the algorithm according to the convention of assigning 40' containers, constraint (6.1), the release constraint (6.7), and the all-loaded constraint (6.6) are also satisfied.

6.3.2 Objective Violations

The objectives have been defined in the form of soft-constraints, where the number of violations is the actual objective value. The overstowage objective (6.12) is represented by

$$\begin{aligned}\nu_8(\pi, s, t, p) &= \exists t' \in \{t-s, \dots, t-1\}, p' \in \mathcal{P} \,.\, \neg\bot(\pi_{st'p'}) \wedge d(\pi_{stp}) > d(\pi_{st'p'}) \\ \sigma_8(\pi) &= \sum_{s\in\mathcal{S}} \sum_{t\in\mathcal{T}_s} \sum_{p\in\mathcal{P}} \nu_8(\pi, s, t, p).\end{aligned}$$

The free reefer slots objective (6.13) is represented by

$$\begin{aligned}\nu_9(\pi, s, t, p) &= A^R_{stp} \wedge (f(\pi_{st1}) \wedge \neg r(\pi_{stp}) \vee t(\pi_{stp}) \wedge \neg r(\pi_{stp})) \\ \sigma_9 &= \sum_{s\in\mathcal{S}} \sum_{t\in\mathcal{T}_s} \sum_{p\in\mathcal{P}} \nu_9(\pi, s, t, p).\end{aligned}$$

The free stack objective (6.15) is represented by

$$\begin{aligned}\nu_{10}(\pi, s, t, p) &= \begin{cases} \frac{1}{\vartheta^C_s} & : \neg\bot(\pi_{stp}) \\ 0 & : \text{ otherwise} \end{cases} \\ \sigma_{10} &= \sum_{s\in\mathcal{S}} \sum_{t\in\mathcal{T}_s} \sum_{p\in\mathcal{P}} \nu_{10}(\pi, s, t, p).\end{aligned}$$

The last objective, pure stacks (6.14) is represented by

$$\begin{aligned}\nu_{11}(\pi, s, t, p) &= \begin{cases} \frac{|\{d(\pi_{s'tp}) \,|\, t' \in \mathcal{T}_s, p' \in \mathcal{P}, \neg\bot(\pi_{st'p'})\}|}{\vartheta^{C\pi}_s} & : \quad \neg\bot(\pi_{stp}) \\ 0 & : \quad \text{otherwise} \end{cases} \\ \sigma_{11} &= \sum_{s\in\mathcal{S}} \sum_{t\in\mathcal{T}_s} \sum_{p\in\mathcal{P}} \nu_{11}(\pi, s, t, p).\end{aligned}$$

As for ν_5 and ν_6, we distribute the violation degree equally over variables holding containers since it is not possible to identify which one of them will have the largest influence.

6.3.3 Incremental evaluation

Candidate swap variables are selected based on their current violation degree; once the first variable is chosen, the algorithm evaluates the possibility of swapping the container related to this variable with the container of any other variable. The incremental evaluation of swap moves is performed using specialized algorithms for each constraint and objective. Evaluation of a move with those operations only involves a partial re-computation of the constraints or objectives that are affected by the change. Formally, let $swap(\pi, \gamma)$ be the assignment π' resulting from performing swap $\gamma \in \Gamma$ on the current assignment π. For a constraint or objective $\sigma \in \{\sigma_1, ..., \sigma_{11}\}$, we then define the delta change $\delta_\pi(\sigma, \gamma) = \sigma(\pi') - \sigma(\pi)$.

Taking the weight constraint (σ_6) as an example, the violation based on the swap can be recalculated by removing the weight of the swapping containers from the current weight of their respective stacks and then adding it to the current weight of the respective stacks of destination. The difference in violation between the new weight violation and the original one would be the result of the delta evaluation. The delta evaluations for constraints σ_1, σ_2, σ_4, and σ_5 are similar to the weight constraint and will not be described further. The σ_3 and σ_7 constraints on the other hand are more complex, since the swap may affect many containers in the involved stacks. Here we show how to define one of these constraints for a specific swap case. The remaining constraints can be implemented using similar ideas.

We consider the no 20' containers on top of 40' constraint (σ_7) for a swap $\gamma = (\langle s, t, c\rangle, \langle s, t', c'\rangle)$ within the same stack of a 40' container with two null containers ($swap^{40}$ type 5). The other cases can be defined in a similar fashion. If the 40' container is above the null containers, the value of the delta evaluation is equal to the number of 20' containers between the null containers and the 40' container. Otherwise it is simply the negative of this. Thus, if $tb_\pi(c, c')$ denote the number of 20' containers stored between container c and c', the delta evaluation for this swap case is defined by

$$\delta_\pi(\sigma_7, \gamma) = \begin{cases} tb_\pi(c, c') & : t > t' \\ -tb_\pi(c', c) & : \text{ otherwise} \end{cases}, \text{ where } tb_\pi(c, c') = \sum_{m=t}^{t'} \sum_{p \in \mathcal{P}} t(\pi_{stp}).$$

Similar to the constraints, the delta evaluation of the objectives can be divided into simple (σ_9 and σ_{10}) and complex (σ_8 and σ_{11}) cases. The delta evaluation for objectives σ_9 and σ_{10} are similar to the weight constraint (σ_6) and are therefore not described further. Here we show how to define the delta evaluation of the overstowage objective (σ_8). The delta evaluation of the pure stack objective (σ_{11}) can be computed in a similar way.

Consider a single 20' to 20' swap $\gamma = (\langle s, t, c\rangle, \langle s', t', c'\rangle)$ ($swap^{20}$ type 1) between two distinct stacks. The delta evaluation is computed using the following schema. For each container, we remove the violation it contributes, and we add the violation that it creates when swapped to its new position. A graphical representation of the algorithm is shown in Figure 6.1. Consider for the moment the container c, following the evaluation schema. Its violation $\nu_8(\pi, s, t, p)$ must first be removed, but also the violations of the containers overstowing c. For this purpose we have defined the function $o_\pi(s, t, c)$, which given a cell (s, t) and a container c, counts the number of containers stowed under the specific cell which are overstowed by c. Formally

$$o_\pi(s, t, c) = \sum_{t'=1}^{t-1} \sum_{p' \in \mathcal{P}} ov_\pi(c, \pi_{st'p'}), \text{ where } ov_\pi(c, c') = \begin{cases} 0 : & \perp(c) \vee \perp(c') \\ 1 : & d(c) > d(\pi_{st'p'}) \\ 0 : & \text{otherwise} \end{cases}.$$

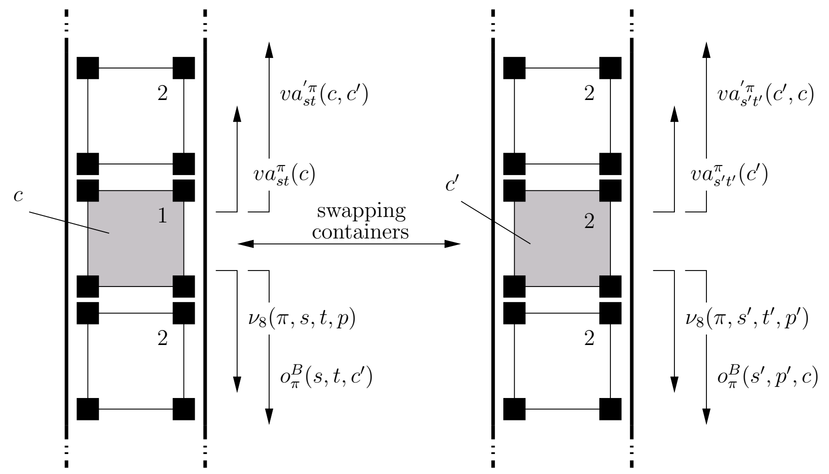


Figure 6.1: Graphical representation of the delta evaluation of the overstowage objective for a $20' \leftrightarrow 20'$ swap between two distinct stacks.

One violation can now be subtracted for each container c^a above c which *only* overstow c (i.e., $d(c^a) > d(c) \wedge o_\pi(s,t,c^a) = 1$), since only in this case will the violation of c^a change to zero when c is removed. This is done using the function $va^{\pi}_{st}(c) = \sum_{m=t+1}^{N^T} \sum_{n\in\mathcal{P}} \big(\neg\bot(\pi_{smn}) \wedge o_\pi(s,m,\pi_{smn}) = 1 \wedge d(\pi_{smn}) > d(c)\big)$. The next step is adding the violations created by replacing c' with c in stack (s',t'). Here we can use $o_\pi(s',t',c)$ and add one violation if $o_\pi(s',t',c) > 0$. Counting the number of violations caused by c to the containers above c' is a similar to the $va^{\pi}_{st}(c)$ computation, but we must remember that the current violation is based on c' being present in stack (s',t'). The previous function is thus modified to be

$$\begin{aligned} va'^{\pi}_{s't'}(c,c') = \sum_{m=t'+1}^{N^T} \sum_{n\in\mathcal{P}} & \neg\bot(\pi_{s'mn}) \wedge \\ & (o_\pi(s',m,\pi_{s'mn}) = 0 \wedge d(\pi_{s'mn}) > d(c)) \vee \\ & (o_\pi(s',m,\pi_{s'mn}) = 1 \wedge d(\pi_{s'mn}) > d(c) \geq d(c'))\,. \end{aligned}$$

Performing the same operations for c' the complete delta evaluation is given by

$$\begin{aligned} \delta_\pi(\sigma_8,\gamma) = & -\nu_8(\pi,s,t,p) - va^{\pi}_{st}(c) + o^B_\pi(s',t',c) + va'^{\pi}_{s't'}(c,c') \\ & -\nu_8(\pi,s',t',p') - va^{\pi}_{s't'}(c') + o^B_\pi(s,t,c') + va'^{\pi}_{st}(c',c), \end{aligned}$$

where $o^B_\pi(s,t,c) = 1$ if $o_\pi(s,t,c) > 0$ and 0 otherwise.

The delta evaluations must be highly efficient operations due to their extensive use in the search procedures. For most of the constraints and objectives (σ_1, σ_2, σ_4, σ_5, σ_6, σ_9, and σ_{10}), the delta swap function run in constant time. This is not the case for the more complex ones (σ_3, σ_7, σ_8, and σ_{11}), but it is easy to show that the complexity in these cases is at most linear in the number of tiers.

Incremental evaluation is then also applied to the computation of the current assignment if a beneficial swap is found. The model is not recalculated entirely, only those parts that are influenced by the change are re-evaluated. Such partial re-evaluation is referred to as *incremental update*. Incrementally maintained variables in the model are the violation variables ($\nu_1 ... \nu_{11}$ and $\sigma_1 ... \sigma_{11}$) and the auxiliary variables (ϑ_s^W, ϑ_s^H and ϑ_s^C). Incremental updates are implemented using very similar principles to the incremental evaluation of swaps and are therefore not discussed any further.

6.3.4 Placement heuristic

The aim of the placement heuristic is to find an initial container assignment. The procedure tries to minimize the objectives and satisfy as many of the constraints as possible. The heuristic is based on simple rules suggested by the nature of the constraints and objectives. Pre-placed containers are assigned to their original position in a pre-processing step, thus implicitly fulfilling constraint (6.7). Stacks are then filled bottom-up one container at a time, making sure that all containers have support (σ_3). Before the placement, all the containers to be loaded $\mathcal{C} \backslash \mathcal{C}^P$ are sorted using the ordering $\preceq_c$ defined by

$$\begin{aligned} c \preceq_c c' \Leftrightarrow\ & d(c) > d(c') \\ & \vee\ d(c) = d(c') \wedge t(c) \wedge \neg t(c') \\ & \vee\ d(c) = d(c') \wedge (t(c) \Leftrightarrow t(c')) \wedge r(c) \wedge \neg r(c') \\ & \vee\ d(c) = d(c') \wedge (t(c) \Leftrightarrow t(c')) \wedge (r(c) \Leftrightarrow r(c')) \end{aligned}$$

This ordering reduces overstowage by placing first containers with a later discharge port. 20' containers are placed before 40' containers in an attempt to avoid the placement of 20' containers on top of 40' containers (σ_7), and reefer containers are assigned before non-reefer since usually reefer slots are at the bottom of a bay (σ_4).

The placement procedure is shown in Algorithm 1. A critical point is the assignment of 20' containers, since it is possible to have two 20' containers with different discharge ports being assigned to two different stacks and generating odd cells (cells that only contain one 20' container). In order to avoid this behavior, the placement heuristic uses the *oddSlot* flag, which is raised when a 20' container is assigned to a slot in an empty cell. When the flag is raised, the heuristic is forced to place the next 20' container in the empty slot of the odd cell (line 3-4 and 15-16). This check ensures that as long as the number of containers to load in the location is consistent with the location capacity, the placement heuristic will always be able to assign all containers to a slot. The procedure places the containers one at a time (line 2), first trying to find space on a stack that contains containers with the same discharge port (line 5-6). If none is found, it tries to find a stack with containers that have a greater discharge port minimizing overstowage (line 7-8). If it is not yet possible to find a placement for the container, then an empty stack is chosen (line 9-10). Should none of those cases find a suitable assignment, the container is pushed to the *WAIT_STACK* (line 11-12). Once a placement has been tried for each container, the *WAIT_STACK* is emptied and each container is placed sequentially on the first available stack (line 13-18). The selection of stacks (lines 5-10) is done in an order $\preceq_s$ based on the number of tiers: $s \preceq_s s' \Leftrightarrow |\mathcal{T}_s| \leq |\mathcal{T}_{s'}|$. This ordering helps the heuristic identify an assignment which uses as few stacks as possible.

Algorithm 1: Placement heuristic

```
1  Require: all pre-placed containers assigned their given slot, $\mathcal{C} \setminus \mathcal{C}^P = \{$all containers to load$\}$ are sorted according to the ordering $\preceq_c$
2  forall $c \in \mathcal{C} \setminus \mathcal{C}^P$ do
3      if $t(c) \wedge oddSlot$ then
4          place c in the odd slot
5      else if $\exists s \in \mathcal{S}, t \in \mathcal{T}_s, p \in \mathcal{P} \,.\, \neg\bot(\pi_{stp}) \wedge d(\pi_{stp}) = d(c)$ then
6          assign $c$ to stack $s$
7      else if $\exists s \in \mathcal{S}, t \in \mathcal{T}_s, p \in \mathcal{P} \,.\, \neg\bot(\pi_{stp}) \wedge d(\pi_{stp}) > d(c)$ then
8          assign $c$ to stack $s$
9      else if $\exists s \in \mathcal{S} \, \forall t \in \mathcal{T}_s, p \in \mathcal{P} \,.\, \bot(\pi_{stp})$ then
10         assign $c$ to stack $s$
11     else
12         push(c, WAIT_STACK)
13 while NOT empty(WAIT_STACK) do
14     $c = pop(WAIT\_STACK)$
15     if $t(c) \wedge oddSlot$ then
16         place c in the odd slot
17     else
18         place c in the first available stack
```

6.3.5 The Search Strategy

Once an initial assignment has been generated, the actual CBLS procedure refines this assignment by first reaching a feasible solution and subsequently optimizing its objective value. Briefly this algorithm can be seen as two sequential hill-climbing phases based on the min-conflict heuristic. To simplify the description of the algorithm, we define the following functions $\nu^\sigma(\pi, s, t, p) = \sum_{i=1}^{7} \nu_i(\pi, s, t, p)$ and $\delta^\sigma_\pi(\gamma) = \sum_{i=1}^{7} \delta_\pi(\sigma_i, \gamma)$ representing respectively the violations attributed to a variable and the incremental evaluation of a possible swap. And similarly for the objectives $\nu^o(\pi, s, t, p) = \sum_{i=8}^{11} \nu_i(\pi, s, t, p)$ and $\delta^o_\pi(\gamma) = \sum_{i=8}^{11} \delta_\pi(\sigma_i, \gamma)$.

The general search procedure is outlined in Algorithm 2. It is important to notice that the swap selection is done through the selection of two slots, where a slot is defined by the triple $\tau = \langle s, t, p \rangle$, where s is the stack, t is the tier, and p is the position of the slot. Let T denote the set of all slot triples. The function $\Gamma(\tau, \tau') : T \times T \mapsto \Gamma$ then defines the swap based on the containers found in the two slots defined by τ and τ'.

Algorithm 2: CBLS algorithm

```
1  $sideMove \leftarrow$ false
2  $\pi = placementHeuristic()$
3  while $\sum_{i=1}^{7} \sigma_i > 0$ do
4      $\pi' \leftarrow \pi$
5      selectMax $s \in \mathcal{S}, t \in \mathcal{T}_s, p \in \mathcal{P}$  on $\nu^\sigma(\pi, s, t, p)$ do
6          selectMin $s' \in \mathcal{S}, t' \in \mathcal{T}_s, p' \in \mathcal{P}$  on $\delta^\sigma_\pi(\gamma = \Gamma(\langle s, t, p\rangle, \langle s', t', p'\rangle))$ do
7              $\pi \leftarrow swap(\pi, \gamma)$
8          $sideMove \leftarrow \sigma(\pi') = \sigma(\pi)$
9  $changed = true$
10 $sideMove = 0$
11 while $changed \wedge sideMoveCount < MAX\_SIDEMOVES$ do
12     $changed = false$
13     selectMax $s \in \mathcal{S}, t \in \mathcal{T}_s, p \in \mathcal{P}$  on $\nu^o(\pi, s, t, p)$ do
14         select $s' \in \mathcal{S}, t' \in \mathcal{T}_s, p' \in \mathcal{P}$ do
15             $changed = true$ on $\delta^o_\pi(\gamma = \Gamma(\langle s, t, p\rangle, \langle s', t', p'\rangle))$
16             $\pi = swap(\pi, \gamma)$
17             $sideMove = 0$
18         if $\neg changed$ then
19             select $s' \in \mathcal{S}, t' \in \mathcal{T}_s, p' \in \mathcal{P}$ on $eval^o_\pi(\gamma = \Gamma(\langle s, t, p\rangle, \langle s', t', p'\rangle))$ do
20                 $sideMove \leftarrow sideMove + 1$
21                 $changed = true$
22                 $\pi = swap(\pi, \gamma)$
23 return $\pi$;
```

The algorithm starts with the initial candidate assignment returned by the placement heuristic on line 2, and then performs the first hill-climbing CBLS over the neighborhood until all the constraints are satisfied (line 3-8). The step function, which selects a new assignment, makes the selection of the slots that will define the swap in two phases. It begins (line 5) by selecting a slot triple $\tau = \langle s, t, p\rangle$ preventing, however, the selection of slots holding pre-placed containers or null containers, and selects only those that actually violate some constraints. Between all the possible slots the one that contains the container with the maximum degree of violation is selected. The algorithm proceeds (line 6) by selecting another slot triple $\tau' = \langle s', t', p'\rangle$ and prioritizes swaps that improve the objective value. The selection in this case filters out all the slots holding pre-placed containers and make sure that the second slot is not the same as the first one. Moreover it selects only slots, which resulting swap actually minimize the violations the most. The *sideMove* flag has been included in the filter in such a way that if raised, it will allow the selection of swaps which will result in an assignment that does not improve the solution, in an attempt to escape the local minima. The *sideMove* flag is initiated in line 1 and is raised in line 8 if no selection of swap is performed. Should a swap selection have been made, the swap is then actually performed in line 7 using the function $swap(\pi, \gamma)$.

Once a feasible solution is reached, the second hill-climbing phase begins (line 9-23). Here the objectives start having a central role in the search process.

The swap selection in this phase needs, however, to be smarter since for some objectives, in particular overstowage (σ_8), free stack (σ_{10}) and pure stack (σ_{11}), a single swap often does not lead to a change in the objective value. To address this problem, we have defined a tie-breaking function that can evaluate a swap that has no objective value improvement, but still causes a more desirable assignment. The function is called $eval_\pi(o, \gamma)$ and equals the evaluation value of the swap γ for the objective $o \in \{\sigma_8, \sigma_9, \sigma_{10}, \sigma_{11}\}$.

For the overstowage objective (σ_8) the evaluation function is defined as the number of containers overstowed by each container, which will make the algorithm choose a container that overstows many containers over one that only overstows a single container. For the free stack objective (σ_{10}) the swap is evaluated by the number of containers in the stack, so that the search rather swaps containers that are in almost empty stacks. The pure stack objective (σ_{11}) calculates the evaluation by summing the quadratic product of the number of containers with the same discharge port, which will favor those stacks that have the most containers with the same discharge port. The optimality phase uses the function $eval^o_\pi(\gamma) = \sum_{i=8}^{11} eval_\pi(\sigma_i, \gamma)$ as a tie-breaking rule, and similar to the feasibility phase the functions $\nu^o(\pi, s, t, p) = \sum_{i=8}^{11} \nu_i(\pi, s, t, p)$ and $\delta^o_\pi(\gamma) = \sum_{i=8}^{11} \delta_\pi(\sigma_i, \gamma)$ represent the total number of objective violations and the sum of all the delta evaluations of swaps, respectively.

The selection of candidare slots from which a swap is generated is performed similarly to the first phase. The first slot is selected between all the non-empty slots holding no pre-placed containers; the one with the maximal objective violation is chosen (line 13). The second slot is selected randomly among all the slots that generate swaps leading to feasible solutions with improved objective value (line 14). Should a swap not be selected, the tie-breaking rule comes into action at line 19, where a new second slot is selected using the tie-breaking rule as defined by the evaluating function on non-improving swaps. The counter *sideMove* is used to limit the number of side moves that the algorithm may perform. The local search terminates once the maximum number of side moves is reached (line 11).

In general, using a neighborhood operator that checks all possible swaps given an initial selection, is clearly more expensive in terms of runtime than a more stochastic one where only a limited number of swaps is evaluated. However this selection has shown to perform moves that are highly valuable in terms of solution improvement, allowing the search to converge quickly.

Class	40’	20’	Reefer	HC	DSP>1	Inst.
1	√					6
2		√				18
3	√	√				4
4	√			√		42
5	√	√		√		27
6	√		√	√		8
7	√	√	√	√		7
8	√			√	√	7
9	√	√		√	√	10
10	√		√	√	√	2
11	√	√	√	√	√	2

Table 6.2: *Test Set Characteristics.* The first column is an instance class ID. Column 2, 3, 4, and 5 indicate whether 40’, 20’, reefer, and high-cube containers are present. Column 6 indicates whether more than one discharge port is present. Finally, column 7 is the number of instances of the class.

6.4 Experimental Evaluation

The algorithm has been implemented in C++ and all experiments have been conducted on a Linux system with 8 Gb RAM and 2 Opteron Quad Core CPUs, each running on 1.7 GHz and having 2Mb of cache. We use a test data set of 133 instances, including locations between 6 TEUs and 220 TEUs. Table 6.2 gives a summarized overview of these instances.

The experimental results on the test data set, which can be seen in Table 6.3(a), show the percentage of solutions solved within a specific optimality gap (optimal solutions have been generated using the constraint programming algorithm described in Delgado et al. (2012)). It is easy to see that only few instances diverge from near optimality and that in 86% of the cases the algorithm actually reached the optimal solution. Studying the algorithm performance closer, we were able to gain some insight on the quality of the different phases. The heuristic placement does not take the height and weight constraint into account, leaving space for improvements. However Table 6.3 (c) and (d) show that the solution found by the heuristic placement is not far from being feasible in most of the cases. This is supported by the fact that often feasibility is reached within 20 iterations and that 61% of the time, the objective value is not compromised. The quality of the objective value of the first feasible solution is also optimal in 74% of the cases (Table 6.3(b)), suggesting that the heuristic placement procedure is performing well. The results also support our initial hypothesis suggesting that the problem is under-constrained and that as such it is possible to heuristically find high quality solutions in short time. The results also point to the fact that the optimality phase improves only a limited number of instances, which however is important especially in the cases where the optimality gap after the feasibility phase is more than 20%.

The limited improvement of solutions in the optimality phase probably happens

Opt. Gap	
Gap.	Freq.
0%	86%
1%	2%
2%	2%
3%	2%
4%	1%
10%	1%
15%	4%
20%	1%
30%	2%

(a)

Opt. Gap (Feas.)	
Gap.	Freq.
0%	74%
5%	6%
10%	2%
15%	5%
20%	3%
25%	3%
35%	3%
40%	1%
> 100%	3%

(b)

Feas. Iter.	
Iter. ≤	%
0	29%
5	23%
10	18%
15	8%
20	6%
25	3%
30	6%
35	1%
40	2%
45	2%
50	1%
65	2%

(c)

Feas. Worse	
Worse %	%
-20%	2%
-10%	2%
-5%	4%
0%	61%
5%	8%
10%	2%
20%	7%
> 20%	15%

(d)

Table 6.3: *Algorithm Analysis* (a) Cost gap between returned solution and optimal solution. (b) Cost gap between first feasible solution and optimal solution. (c) Number of iterations needed to find the first feasible solution. (d) Worsening of the cost of the heuristic placement when searching for the first feasible solution.

because the search does not allow for a large degree of diversification. Preliminary tests using tabu search have shown that local diversification often was unsuccessful to escape a local minimum due to large structural differences between the local minimum and an optimal solution. A possible approach to solve this problem could be to change the initial placement and the search procedures to follow the heuristic less closely. This method, however, would probably be more expensive in terms of runtime performance.

The average runtime of the instances is 0.18 seconds, with a worst case of 0.65 seconds. Figure 6.2 shows the runtime of the algorithm as a function of the size of the instance measured in TEUs. As depicted, the execution time scales well with the instance size. Figure 6.3 compares the execution time between our algorithm and the complete constraint programming approach used for generating optimal solutions. When generating the instance set for investigating the optimality gap shown in Table 6.3, we excluded instances that were not solvable by the CP approach within 160 seconds. However, in a comparison between the two approaches these instances are particularly interesting.

As depicted, the CP approach is highly competitive within the set of instances that it can solve in 160 seconds. However our approach can solve the problematic instances for CP fast as well. This result indicates that the under-constrained nature of the problem might force exact methods to spend an excessive amount of time proving optimality.

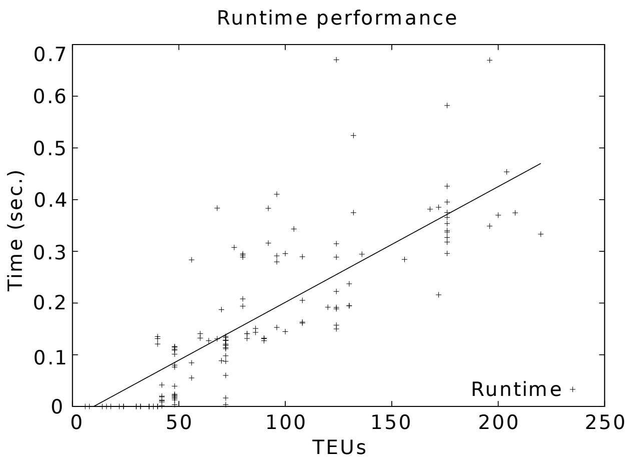


Figure 6.2: Execution time as a function of instance size measured in TEUs.

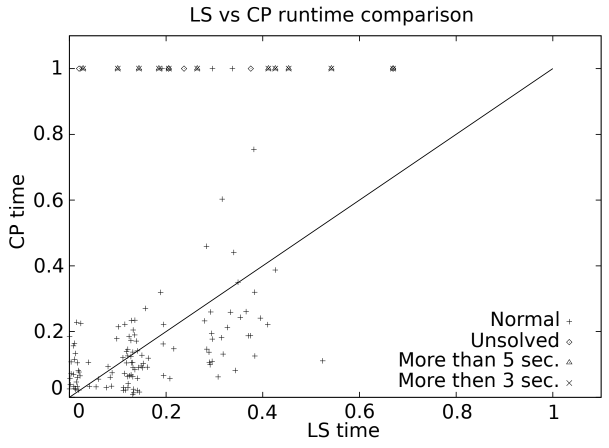


Figure 6.3: Execution time comparison between our algorithm and the complete constraint programming approach used to generate optimal solutions.

6.4.1 Dealing with Fractional and Infeasible Instances

As mentioned before, the abstracted decisions taken during master planning might result in infeasible slot planning instances. CBLS is of great help in this situation, since, thanks to the concept of constraint violations, it is possible to analyse an infeasible instance. Once the feasibility phase fails, the resulting solution is one where the violations have been minimized. This means that the solution is the one closest to feasibility that the algorithm could find. We then find a feasible solution using algorithm 3.

The algorithm selects the variable with the highest degree of violation (line 2) and rolls out the stowed container (line 3). Since the removed container might be in the middle of a stack it might be necessary to lower down all the container in higher tiers (line 6). Should the removed container be a 20' container (line 4) it is necessary to remove the other 20' container as well (line 5). The process is iterated until no more violations exists and thus the solution is feasible (line 1). Note that this heuristic

Algorithm 3: Feasibility heuristic

```
1 while $\sum_{i=1}^{7} \sigma_i > 0$ do
2     selectMax $s \in \mathcal{S}, t \in \mathcal{T}_s, p \in \mathcal{P}$  on $\nu^\sigma(\pi, s, t, p)$ do
3         $x_{stp} = \perp$
4         if $t(x_{stp})$ then
5             $x_{stp^{-1}} = \perp$
6         adjustStack(s)
```

procedure aims at reaching a feasible solution fast. The procedure could, each time a container is dropped, run a search for a new feasible solution thus minimizing the number of dropped containers. We do not believe, however, this to be necessary as we do not expect a large number of dropped containers.

The slot planning experiments discussed in this section are based on master plans obtained from the IP and MIP multi-port master planning experiment with 5% optimality gap previously presented in chapter 5.

So far we have talked about single instances of slot planning, however, when complete stowage plans are generated, master plans based on the MIP model may have fractional numbers of containers to stow in locations which is physically impossible. It is thus necessary to deal with fractionality looking at the vessel as a whole.

We attempt to stow a fractioned container in one of the locations where a fraction of it has been assigned by the master plan. If none of these locations have capacity left, the container is rolled out.

The quality of each slot plan is evaluated, again, by comparing it with the best slot plan generated by CP for the same location within twenty minutes. In the cases where the number of containers of one or more types were reduced by CBLS, the slot plan is evaluated against the best plan generated by CP within twenty minutes, stowing the same containers as CBLS. Table 6.4 summarizes the results of our experiments.

Vessels are slot planned fast by our approach. For the instances with a master plan available, slot plans are generated within an average of 5.9 seconds in total. There is no time-wise dominance of slot plans generated from MIP and IP master plans, indicating that integrality constraints do not affect the complexity of slot planning. When slot planning IP master plans, a reduction in the number of containers rolled out due to fractionality and odd number of 20' containers in locations is observed in most of the instances. This is, however, a very small fraction (0.88% max). In the instances where IP and MIP master plans are provided, the number of containers rolled out by CBLS differs in average only in 0.85 containers (4.65 for the IP and 5.5 for the MIP master plans). These facts indicate no considerable impact of using MIP master plans.

Instance number four has the maximum roll-out for IP and MIP slot plans. This instance contains locations where entire stacks are limited to a specific container length. Such particularity constrains the possible combinations of containers that fulfill stack weight limits, forcing CBLS to drop containers. Even considering this outlier, the aver-

Slot Planning Current Port														
ID	Conts.	Time(s)		Locs.		Frac.+Odd		CBLS rolled		Rolled out(%)		Gap(%)	
		Cont.	Int.	Cont.	Int.	Cont.	Int.	Cont.	Int.	Cont.	Int.	Cont.	Int.
1	500	1.83	1.82	28	24	5	7	7	0	2.4	1.4	0.91	0.0
2	2086	11.22	11.91	76	76	14	10	15	7	1.39	0.81	0.56	0.12
3	2086	11.15	11.91	76	76	14	10	15	8	1.39	0.86	0.5	0.1
4	407	2.27	3.39	21	24	8	8	28	39	8.85	11.55	0.05	0.25
5	288	1.23	1.14	12	10	1	1	0	0	0.35	0.35	0.0	0.0
6	2227	14.87	13.82	74	76	28	11	15	20	1.93	1.39	0.92	0.64
7	125	2.0	2.76	14	10	3	3	0	0	2.4	2.4	0.0	0.0
8	78	1.22	1.67	9	11	3	1	0	0	3.85	1.28	0.0	0.0
9	278	4.04	5.29	15	12	5	1	0	0	1.8	0.36	0.0	0.31
10	172	3.76	3.36	17	17	4	6	0	0	2.33	3.49	0.23	0.68
11	-	-	-	-	-	-	-	-	-	-	-	-	-
12	571	12.19	11.52	41	41	0	0	0	0	0.0	0.0	0.0	0.0
13	2190	17.02	18.84	66	73	7	5	0	3	0.32	0.37	0.15	0.0
14	641	2.1	-	35	-	6	-	0	-	0.94	-	0.0	-
15	787	9.44	8.83	38	41	13	4	0	0	1.65	0.51	1.1	0.23
16	1075	7.84	-	36	-	5	-	0	-	0.47	-	0.0	-
17	401	2.06	2.52	19	19	2	2	0	0	0.5	0.5	0.0	0.0
18	569	5.02	4.17	39	34	2	2	0	0	0.35	0.35	0.15	0.67
19	869	5.95	5.75	30	29	3	7	0	0	0.35	0.81	0.0	0.0
20	631	4.04	3.91	27	26	3	11	0	0	0.48	1.74	0.0	0.04
21	989	9.75	10.03	38	33	5	1	0	9	0.51	1.01	0.63	0.86
22	764	5.1	8.28	18	20	5	3	27	0	4.19	0.39	1.03	0.6
23	608	5.7	5.94	31	31	1	1	0	0	0.16	0.16	0.0	0.0
24	474	5.64	-	14	-	2	-	29	-	6.54	-	0.0	-
25	104	1.86	1.72	11	12	3	3	0	0	2.88	2.88	0.0	0.0
26	253	1.71	2.6	19	17	4	6	0	0	1.58	2.37	0.0	0.0
27	280	4.0	2.12	17	18	4	8	0	0	1.43	2.86	0.37	0.0
28	1116	6.78	5.39	31	34	6	4	0	0	0.54	0.36	0.66	0.0
29	421	2.96	2.11	22	24	1	3	0	0	0.24	0.71	0.02	0.0
30	193	2.34	2.83	16	12	3	3	0	0	1.55	1.55	3.45	0.0

Table 6.4: *Slot Planning MIP (Cont.) and IP (Int.) master plans with 5% gap.* The first and second columns are the id of the instance and the number of containers to stow in the first port. The next columns show grouped results of slot planning based on MIP and IP master plans. The third and fourth columns show the runtime for the slot plans, fifth and sixth columns is the number of locations to which slot planning was to be performed. The seventh and eighth columns totalize the number of rolled out containers by fractionality and odd number of 20' containers, the ninth and tenth columns are the containers rolled out by CBLS, and eleventh and twelfth columns are the percentage of total containers rolled out. The last two columns show the average gap of the slot plans. A dash indicates that no master plan was provided.

age percentage of roll-out containers is 1.7% and 1.5% when slot planning IP and MIP master plans, respectively. These numbers are reasonable given the amount of containers typically rolled from a loadlist by stowage coordinators. Note that the improved weight constraints in the master planning model with ballast tanks (see Section 5.3) would solve this particular issue since specific weight constraints are considered for each container length.

Of the slot plans generated for the IP and MIP master plans 2.24% have an optimality gap over 5%, and only 1.24% a gap over 10%. The maximum gap, however, is 55%. CP generated optimal slot plans within one second for 96.2% of the locations of the MIP master plans and 96.0% of the locations of the IP ones. Moreover, CP was able to prove optimality 87.8% and 86.5% of the slot plans generated for the locations for the MIP and IP master plans, respectively.

Chapter 7

Conclusions

The aim of this thesis has been to investigate the possibility of applying automated stowage planning in practice. The work presented does not aim at showing that stowage coordinators can be substituted by automated planners, but rather to explore the possibility of using automatically generated stowage plans in decision support tools for stowage coordinators. In order to answer the research question, two main aspects of automated stowage planning were to be taken into account. First, the quality of the generated solution and second, the performance of the system in terms of runtime. To ensure the generated solutions are of high quality, we have proposed a representative model in collaboration with the industry. This representative problem includes all the relevant computational components of container stowage planning and all the details that make the solutions of this problem possible to evaluate by stowage coordinators. The representative problem requires the handling of vessel stability and stress forces, the consideration of 20' and 40' containers and all their stacking rules, and the optimization of overstowage and crane utilization.

The algorithmic approach presented in this thesis hierarchically decomposes the problem into a master planning and a slot planning phase. During the master planning phase, groups of containers are assigned to sub-sections of the vessel. During this phase, stability and stress forces are solved, hatch-overstowage and crane utilization is optimized. The slot planning phase, solves the stowage problem of each location, handling stacking constraints and optimizing overstowage. This decomposition, and the runtime performance of the two phases, is able to solve large scale container stowage problems within 10 minutes, which is the time limit given by the industry for a tool to be useful within the day to day planning routine of stowage coordinators. Further improvements of the master planning model were presented in this thesis that analysed the possibility of including ballast tanks in the optimization model. We also presented an extension of the computational complexity results for the container stowage planning problem, proving that the open problem of the complexity of the capacitated k-shift problem is in P, and that the hatch overstow problem is NP-Complete.

The experimental results collected in this thesis show that automated stowage plans can be generated for the representative problem within the 10 minutes time frame set by the industry, thus giving a positive answer to the research question central to this thesis. Moreover, it has been shown that ballast water can be successfully modelled using standard mathematical programming methods via the use of linearization of hydrostatic data. The possibility of modeling such problems with standard modeling tools provides the industry with the flexibility of easily adding side-constraints and changing the objectives.

7.1 Outlook and Future Directions

This thesis has shown a proof of concept that container stowage planning can be automated, resulting in solutions of high quality within a short time. The integration of such system within the planning process of stowage coordinators is an interesting direction of future work. In order for the current state-of-the-art to be part of the daily work routine, a number of extensions are needed, such as the handling of lashing forces and support for special requirement container types. It is debatable whether those extensions should be a part of the optimization model or if they should be included into an external decision process. The modality in which optimization models integrate into daily work routines can be a fascinating challenge as they will have to be able to handle the heuristic choices and quality criterions that might change between coordinators. In this direction, the work of Jensen et al. (2012) is the first to apply configuration ideas to the container stowage problem, where binary decision diagrams are used to guide the user.

Besides extensions to the current approach, some of the results obtained indicate other interesting research directions in terms of the general approach taken. Consider, for example, the runtime performance of the two phases. It might be possible to exchange information between the two phases such that a true optimal solution can be found. Now consider the master planning phase which includes the planning of down stream ports. Cargo flow information for such ports is based on forecast data, which is more imprecise the further down the stream the plan goes. An interesting future analysis would be to use a rolling horizon optimization and study how it impacts the solutions. Such an approach could greatly improve the runtime of the master planning phase.

Stowage Planning in Artificial Intelligence

A very different research direction can be the analysis of the knowledge of stowage coordinators. Successful approaches to stowage planning, such as the one used in this thesis, are based on the work process of stowage coordinators. It is not always simple, however, to extract this knowledge, as different heuristic rules might be used by different coordinators. Systematic analysis of historical data might allow learning algorithms to infer heuristic rules which can be applied to improve the current models. An example could be to estimate the distribution of containers with the same discharge port on a vessel given a loadlist and a release. Such information could be used as branching heuristics, thus boosting the solution process.

Stowage Planning in Classical Computer Science

The computational complexity study from Appendix A also opens future research directions. The capacitated k-shift problem was proven to be in P for fixed size vessels, and since the size of nowadays vessels is not likely to change drastically in the years to come, the search for an efficient polynomial algorithm for this problem might bring some insight about the structure of the container stowage problem as a whole.

Stowage Planning in Related Problems

Container stowage planning can be seen as a central element of many decisions in the liner shipping industry. The fact that it can be solved within acceptable time limits opens the opportunity of solving other problems that depend on stowage planning. Cargo flow analysis over the strings of a liner shipping network could benefit from the integration of stowage planning, as it would be possible to analyse the feasibility of the cargo flows based on the vessels deployed in more detail than the raw TEU capacity of a vessel. Heuristically generated stowage plans could then also be implemented into simulation models for the analysis of business decisions, for example in network design or repositioning.

Appendix A

Complexity Analysis

The main reference, within the published academic works that analyses the container stowage problem with respect to computational complexity is the work of Avriel et al. (2000). The authors analyse the most basic components of the problem and prove that the decision problem of container stowage is NP-complete. This chapter briefly introduces this work and shows additional complexity result for the container stowage problem.

A.1 The k-Shift Problem

The k-shift problem (kSP), first introduced by Avriel et al. (2000), is the problem that asks if a set of containers that must be loaded and discharged at different discrete time points can be stowed with less than k shifts (restows) in a set of stacks with a given capacity. Formally, an instance of kSP is a tuple $\langle n, C, in, out, S, m, k \rangle$, where n is the number of time points, C is the set of containers, $in(c) \in \{1, ..., n\}$ ($out(c) \in \{1, ..., n\}$) is the time points where container c must be loaded in (discharged from) one of the stacks, S is a finite set of stacks, $m \in \mathbb{N}$ is the maximum number of containers a stack can hold, and k is the maximum number of shifts allowed. The question is whether the containers can be assigned to the stacks such that at most k shifts are required to retrieve them. Formally, is there an assignment $A : C \to S$ that is within the stack capacity (i.e., $\forall t \in \mathbb{N}, s \in S \,.\, |\{\, c \mid A(c) = s, in(c) \leq t < out(c)\}\,| \leq m$) that requires at most k shifts (i.e., $|\{w \in C \mid \exists\, v \in C \,.\, A(v) = A(w) \,\wedge\, in(v) < in(w) < out(v) \,\wedge\, in(w) < out(v) < out(w)\}| \leq k$)?

Example 1. Consider the k-shift problem depicted in Figure A.1, where $C = \{c_1 \ldots c_{13}\}$, $in(c_1, \ldots, c_4) = 1$, $in(c_5, c_6) = 2$, $in(c_7, \ldots, c_{10}) = 3$, $in(c_{11}) = 4$, $in(c_{12}, c_{13}) = 5$, $out(c_1, \ldots, c_4) = 3$, $out(c_5) = 4$, $out(c_6) = 6$, $out(c_7, \ldots, c_{10}) = 5$, $out(c_{11}, c_{12}, c_{13}) = 6$, $S = \{s_1, s_2\}$, $m = 3$. The answer to this kSP is "yes" for all $k \geq 3$.

Avriel et al. (2000) distinguish between the *Capacitated* k-Shift Problem (CkSP) and the *Uncapacitated* k-Shift Problem (UkSP). There is no difference between the CkSP and the kSP while the UkSP is a kSP where the capacity of the stack $m = \infty$.

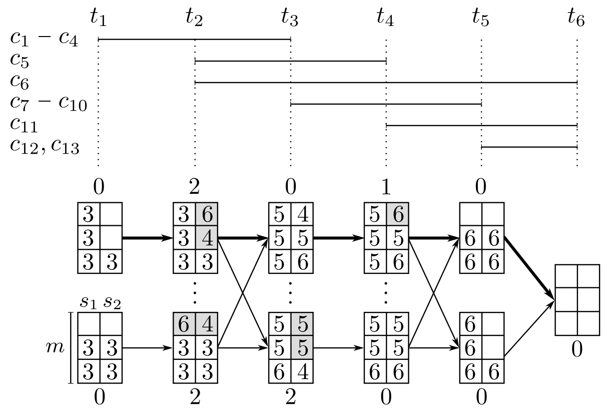


Figure A.1: A kSP instance with $m = 3$. The instance is a "yes"-instance for any $k \geq 3$. Possible configurations (disregarding symmetric ones) are shown at each time step with the discharge time of the container in each slot. Gray slots contain containers that must be shifted, and each configuration is labeled with the number of containers that will be shifted at that time-step. A path through the configurations represents a complete stowage plan, and the optimal plan is shown with bold arcs.

A.2 UkSP is NP-Complete

The NP-completeness of the UkSP is proven by Avriel et al. (2000) by a reduction from the *circle-graph coloring problem.* Avriel et al. (2000) shows that a special case of the UkSP, where $k = 0$, is equivalent to the circle-graph coloring problem and thus known complexity results can be easily applied to the U(zero)SP. Avriel et al. (2000) show how a circle-graph can be constructed from an instance of a U(zero)SP where the number of colors on the circle-graph coloring problem corresponds to the number of stacks in the U(zero)SP. The authors prove the following theorem:

Theorem 1. *Let C be the number of columns in an uncapacitated bay. Then, the uncapacitated shift problem is NP-complete for $C \geq 4$.*

In this theorem C corresponds to the number of stacks $|S|$ and an uncapacitated bay corresponds to the set of stacks in S. Even though the authors base the proof on the U(zero)SP, the result carries over to the UkSP since the U(zero)SP is a special case of the UkSP. Due to the equivalence with the circle-graph coloring problem, it is also known that the U(zero)SP is in P for any uncapacitated bays with less than 4 stacks. Beside Aslidis (1990) O(n^3) algorithm for the UkSP with one stack, the complexity of UkSP for two and three stacks is still an open problem.

A short coming, in terms of practical use of the result, of theorem 1 is that, as the authors hint at, if $|S| \geq |P|$ the problem becomes trivially solvable in polynomial time since stacks can be dedicated to a single discharge port. A more precise theorem would then be:

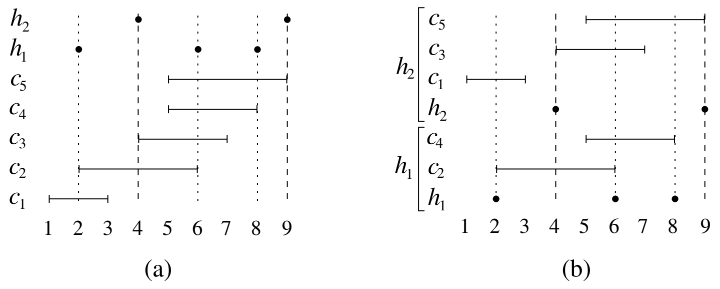


Figure A.2: (a) An HOP instance with $m = 2$ and $k = 1$. (b) an assignment of containers to hatches with only one hatch overstow (for h_1) showing that it is a "yes"-instance.

Theorem 2. *Let C be the number of columns in an uncapacitated bay and P be the number of ports visited. Then, the uncapacitated shift problem is NP-complete for $4 \leq C < |P|$.*

A.3 Complexity of CkSP

Avriel et al. (2000) extend the complexity result of the UkSP of to the CkSP, since a polynomial reduction from UkSP is trivial (it is enough to set $m = |C|$ to simulate an uncapacitated stack). The authors thus show that the CkSP is NP-complete.

Theorem 3. *Let C be the number of stacks of capacity m in a bay. The capacitated k-shifts problem is NP-complete for all $m > 1$ and $C \geq 4$.*

A.4 The Hatch Overstow Problem

In Chapter 2, the concept of hatch-overstowage was introduced. As this problem significantly impacts the total number of re-handles, it is of interest to analyze it's computational complexity. We perform this analysis defining the Hatch Overstow Problem (HOP). HOP is a decision problem that asks whether a set of containers that must be loaded and discharged in a set of numbered ports visited in the order $1, 2, \dots$ can be stowed on a set of hatch covers without causing more than k hatch overstows when the hatches may have to be removed in some ports to access containers stowed below them. Formally, an instance of the HOP is a tuple $\langle C, in, out, H, r, m, k\rangle$, where C is a finite set of containers, $in(c) \in \mathbb{N}$ ($out(c) \in \mathbb{N}$) is the port that container c must be loaded to (unloaded from) one of the hatches, H is a finite set of hatches, $r(h) \in 2^{\mathbb{N}}$ is the set of ports where hatch h must be removed, $m \in \mathbb{N}$ is the maximum number of containers that each hatch can hold at any time, and $k \in \{0, \dots, |C|\}$ is the maximum number of hatch overstows.

The question is whether the containers can be assigned to the hatches without causing more than k hatch overstows. Formally, the question is whether there exists an assignment $A : C \to H$ that is within the hatch capacity (i.e., $\forall t \in \mathbb{N}, h \in H \, . \, \big| \{ c \mid A(c) = h, in(c) \leq t < out(c) \} \big| \leq m$) and has at most k hatch overstows (i.e., $\big| \{ c \mid \exists p \in r(A(c)) \, . \, in(c) < p < out(c) \} \big| \leq k$).

Example 2. Consider an HOP with $C = \{c_1, c_2, c_3, c_4, c_5\}$, $in(c_1) = 1$, $in(c_2) = 2$, $in(c_3) = 4$, $in(c_4) = 5$, $in(c_5) = 5$, $out(c_1) = 3$, $out(c_2) = 6$, $out(c_3) = 7$, $out(c_4) = 8$, $out(c_5) = 9$, $H = \{h_1, h_2\}$, $r(h_1) = \{2, 6, 8\}$, $r(h_2) = \{4, 9\}$, $m = 2$, and $k = 1$. As depicted in Figure A.2, the answer to this HOP is "yes".

We now prove that the HOP is NP-complete by a reduction from the Set Cover Problem (SCP). Recall that an instance of the SCP is a tuple $\langle S, \mathcal{A}, k' \rangle$, where S is a finite set, $\mathcal{A}$ is a collection of non-empty subsets of S, and $k' \in \{1, \ldots, |\mathcal{A}|\}$. The question is whether $\mathcal{A}$ contains a cover for S of size k' or less, i.e. a subset $\mathcal{A}' \subseteq \mathcal{A}$ with $|\mathcal{A}'| \leq k'$ such that every element of S belongs to at least one member of $\mathcal{A}$.[1]

Example 3. Consider an SCP with $\mathcal{A} = \{a_1, a_2, a_3\}$ where $a_1 = \{e_1, e_3\}$, $a_2 = \{e_3, e_4\}$, and $a_3 = \{e_1, e_2, e_4\}$, $S = \{e_1, e_2, e_3, e_4\}$, $k' = 2$. Clearly this is a "yes"-instance since S can be covered by $\mathcal{A}' = \{a_1, a_3\}$.

Theorem 4. The HOP is NP-complete.

Proof We have HOP $\in$ NP since the assignment A can be used as a certificate that can be checked in polynomial time. We next prove that SCP $\leq_p$ HOP which shows that the HOP is NP-hard. The reduction algorithm begins with an instance $\langle \{e_1, \ldots, e_{|S|}\}, \{a_1, \ldots, a_{|\mathcal{A}|}\}, k' \rangle$ of the SCP. We shall now construct a HOP instance $\langle C, in, out, H, r, m, k \rangle$ that has an assignment with no more than k hatch overstows if and only if the SCP instance has a cover with a size no greater than k'. The HOP instance is constructed by reducing the hatch capacity to one and using the containers to represent the elements in S and using the hatches to represent $\mathcal{A}$. The idea is that a container that belongs to a subset can be assigned to a hatch representing it without causing overstowage. To measure the size of these non-overstowing covers, $|\mathcal{A}|$ blocking containers and k' extra hatches are introduced. Formally, we have $C = S \cup \{b_1, \ldots, b_{|\mathcal{A}|}\}$, where $\langle in(e_i), out(e_i) \rangle = \langle 2i{-}1, 2i{+}1 \rangle$ for $1 \leq i \leq |S|$ and $\langle in(b_i), out(b_i) \rangle = \langle 1, 2|S|{+}1 \rangle$ for $1 \leq i \leq |\mathcal{A}|$. Further, $H = \mathcal{A} \cup \{f_1, \ldots, f_k\}$, where $r(a_i) = \{2j \,|\, e_j \in S \setminus a_i\}$ for $1 \leq i \leq |\mathcal{A}|$ and $r(b_i) = \{2j \,|\, e_j \in S\}$ for $1 \leq i \leq k$. Finally, we have $m = 1$ and $k = |\mathcal{A}|$. Clearly, this HOP instance can be constructed in polynomial time. As an example of the construction, Figure A.3 shows the HOP instance of the SCP defined in Example 3.

We must show that this transformation of SCP into HOP is a reduction. First suppose that the SCP has a cover $\mathcal{A}' = \{a'_1, \ldots, a'_n\}$ where $n \leq k'$. For the corresponding HOP, assign each container representing an element in S to a hatch representing a subset in the cover that includes it. This is possible with $m = 1$ because none of these

[1]There are slight variations of the SCP in the literature. The one present here differs from SP5 in Garey and Johnson (1979) by requiring that the subsets of $\mathcal{A}$ are non-empty. It is trivial to show that SCP is NP-complete by a reduction from SP5.

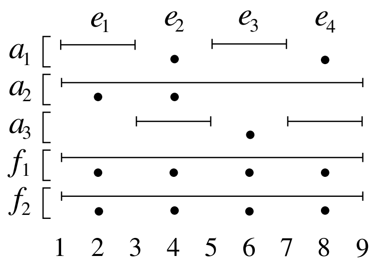


Figure A.3: The HOP instance of the SCP defined in Example 3. As in Figure A.2(b), we show an assignment that proves the HOP to be a "yes"-instance, since there is no more than 3 hatch overstows.

container transports overlap each other. Further, none of these assignments overstow. Then assign $|\mathcal{A}|-n$ blocking containers to the hatches representing subsets that are not included in the cover and assign the remaining n blocking containers to extra hatches. This is possible since no other containers are assigned to these hatches and there are enough extra hatches because $n \leq k'$. Since all the subsets in $\mathcal{A}$ are assumed to be non-empty, we have that all $|\mathcal{A}|$ assignments of blocking containers overstow. The number of overstowing containers, however, is still not greater than k as required.

Conversely, suppose that the HOP has a feasible assignment with $n \leq k$ overstows. Since all blocking containers overstow, we must have $n \geq |\mathcal{A}|$. But since $k = |\mathcal{A}|$, we have $n = k = |\mathcal{A}|$. This means that no element containers overstow, which is only possible if they are assigned to hatches that represent subsets that form a cover $\mathcal{A}'$ of S. The size of this cover can at most be k' because every subset in the cover requires an extra hatch to move a blocking container to and there are only k' of these extra hatches. □

A.5 Remarks

The complexity results presented in this chapter are worth a few remarks. Previous work has shown that even a simple component of the container stowage problem, such as the Capacitated k-Shift Problem, is NP-Complete. We have further confirmed these findings by showing that even when only considering hatch-overstowage the problem is still NP-Complete. It is important to note, however, that for a fixed size vessel it is possible to find solutions in polynomial time for the Capacitated k-Shift Problem. Since vessels' size will unlikely change drastically in the next two decades, even if the algorithm in itself is not efficient, this result opens a new interesting research direction.

Appendix B

Decomposition in Scheduling

This appendix presents the results of a collaboration with Professor Pascal Van Hentenryck. The aim of the collaboration was to explore the application of constraint based decomposition frameworks on scheduling problems, specifically on the flexible jobshop problem. Particular attention is given to the Adaptive Randomized Decomposition (ARD) framework that for the first time is applied to these problems.

B.1 Constraint Based Decompositions for Scheduling Problems

Constraint-based schedulers have been widely successful in modeling and solving complex industrial scheduling applications. They provide a general-purpose approach to applications involving complex resources and constraints. Moreover, in recent years, the combination of Constraint Based Scheduler (CBS) and local search techniques (e.g., large neighborhood search or iterative flattening) has been instrumental in obtaining high-quality solutions quickly and improving best-known solutions in a variety of pure problems such as cumulative scheduling and open-shop problems.

This research considers the application of constraint-based scheduling to flexible jobshop problems, a generalization of the traditional jobshop scheduling where activities have a choice of machines. Flexible jobshops are quite challenging, since they add another level of decisions and reduce the power of filtering algorithms for disjunctive resources. In particular, this paper studies both Large Neighborhood Search (LNS) and ARD inspired by research on large-scale vehicle-routing applications. LNS uses random, temporal, and machine neighborhoods, while ARD exploits temporal and machine decouplings to produce subproblems that can be optimized separately.

Empirical results on some standard benchmarks show that, within 5 minutes, both LNS and ARD produce many new best solutions and are about 0.5% in average from the best-known solutions. Moreover, over longer runtimes, they improve 61% of the best-known solutions and match the remaining ones. The empirical results also show the importance of hybrid neighborhoods and decompositions in LNS and ARD.

These results are obtained with a rather naive CBS search, which requires little development effort on top of a modern constraint-based scheduling and does not use advanced search heuristics or learning techniques. As a consequence, they further demonstrate the versatility of the general-purpose approach of combining CBS with local search and decomposition schemes. The ARD schemes are also shown to be close in quality and efficiency to LNS, indicating that they are likely to provide high-quality solutions to large-scale problems which cannot be handled globally.

B.2 The Flexible Jobshop Problem

A flexible jobshop is specified by a set of jobs, each of which consists of a sequence of activities. Each activity can execute on a set of machines, each with a possibly different duration. No two activities can execute on the same machine at the same time and the goal is to minimize the makespan, i.e., the completion date of all activities. More formally, each activity a has a set $M(a)$ of machines on which it can execute and a duration $d(a,m)$ for each machine $m \in M(a)$. Every job defined by a sequence of activities $\langle a_1, ..., a_n \rangle$ generates a set of precedence constraints (a_{i-1}, a_i) for $i \geq 2$. We use $\mathcal{A}$ to denote the set of activities, $\mathcal{P}$ the set of precedence constraints, and $\mathcal{M}$ the set of machines. The time horizon H for the schedule is given by $\left[0, \sum_{a \in \mathcal{A}} \max_{m \in M(a)} d(a,m)\right]$.

A solution to the flexible jobshop is a pair of assignments (σ, μ), where $\mu : \mathcal{A} \mapsto \mathcal{M}$ assigns a machine $\mu(a) \in M(a)$ to each activity a and $\sigma : \mathcal{A} \mapsto H$ assigns a starting date $\sigma(a)$ to each activity a. A solution is feasible if it satisfies the precedence and capacity constraints, i.e.,

$$\begin{array}{l} \forall (a_i, a_j) \in \mathcal{P} : \ \sigma(a_j) \geq \sigma(a_i) + d(a_i, \mu(a_i)) \\ \forall m \in \mathcal{M}, t \in H : |\mathcal{A}(\sigma, m, t)| \leq 1 \end{array}$$

where $\mathcal{A}(\sigma, m, t)$ is the set of activities assigned to machine m at time t, i.e.,

$$\{a \in \mathcal{A} \ \mid \ \mu(a) = m \ \wedge \ \sigma(a) \leq t \leq \sigma(a) + d(a, \mu(a))\}.$$

An optimal solution is a feasible solution (σ, μ) minimizing

$$\max_{a \in \mathcal{A}} \sigma(a) + d(a, \mu(a)).$$

We also use $\omega(a)$ to denote $\sigma(a) + d(a, \mu(a))$ in this paper.

B.3 Prior Work

Approaches to solve flexible shop problems are often divided between hierarchical and simultaneous searches. Hierarchical heuristics (Akella and Gershwin, 1984; Bona et al., 1990; Brandimarte, 1993; Escudero, 1989) are based on the property that, given a machine assignment, the objective function can be optimized solving a classic jobshop problem. Simultaneous approaches (Barnes and Chambers, 1996; Brucker and Neyer, 1998; Jurisch, 1992; Hurink et al., 1994; Vaessens, 1995; Mastrolilli and Gambardella, 2000; Hmida et al., 2007, 2010; Gao, 2008; Pezzella et al., 2008) tend to identify neighborhoods which either solve the routing and the assignment into different transactions (Barnes and Chambers, 1996; Brucker and Schlie, 1990; Jurisch, 1992; Hurink et al., 1994; Hmida et al., 2007, 2010) or at the same time (Dauzère-Pérès and Paulli, 1997; Vaessens, 1995; Mastrolilli and Gambardella, 2000; Gao, 2008; Pezzella et al., 2008). Specifically to the benchmarks used in this paper, Mastrolilli and Gambardella (2000) proposed a tabu search procedure using two neighborhoods functions defined by the moving of an operation and its feasible or optimal insertions, improving a large number of the best-known upper bounds. An hybrid search combining genetic and variable neighborhood descent algorithms was implemented by Gao (2008), reporting better performance than the tabu search. However, details of new upper bounds have not been reported. A constraint programming approach was developed by Hmida et al. (2010) using discrepancy search but the results are dominated by both Mastrolilli and Gambardella (2000) and Gao (2008).

B.4 A Constraint-Based Scheduling Model

A simple CBS formulation for the flexible shop is shown in figure B.1 (using a Comet-like syntax). Lines 1–5 initialize the constants, i.e., the set of activities and machines, the durations, the set of machines required by each activity, and the set of precedence constraints. Lines 6 and 7 define the decision variables as activity objects, which can be queried for their starting and ending dates. Line 8 defines a pool of unary resources which maintain the machine selection of each activity and the capacity constraints using edge-finder and NotFirst/NotLast propagators. Line 9 declares the array of selected machines for each activity (see also lines 20–21) Lines 11–22 define the objective function and the constraints of the problem. The search procedure is specified in lines 24–30 and is rather naive. It starts by branching over the machine selection variables, considering activities with the fewest machines first (lines 24–26). The machines are then ranked, starting with those with the least slack (lines 27–28), before assigning the earliest starting dates to all activities (lines 29–30).

B.5 Large Neighborhood Search

LNS is a combination of Local Search and CP, which has proved effective in solving large-scale combinatorial optimization problems. The search procedure is based on a destruction/construction principle. Once an initial solution is found, part of the variable assignments are relaxed (*destruction*), while keeping the remaining variables fixed. A new solution is then found by re-optimizing the assignment of the free variables (*construction*). These two steps are then iterated until some termination criterion. When applied to scheduling problems, it is beneficial to impose only precedence constraints

```
range A = ...;
range M = ...;
int[] d[A] = ...;
int[] m[A] = ...;
set{Precedence} P = ...;
Activity act[a in A]();
Activity makespan();
UnaryResourcePool pool(M);
var{int} sm[A];

minimize
  makespan.end()
subject to{
  forall(c in P)
    act[c.before] precedes act[c.after];
  forall(a in A)
    act[a] precedes makespan;
  forall(a in A)
    act[a] pool.requires(m[a],d[a]);
  forall(a in A)
    sm[a] = pool.getSelectedMachine(act[a]);
}
using {
  forall(a in A) by (size(sm[a]))
    tryall(m in m[a])
      sm[a] = m;
  forall(m in M)  by (slack(m))
    rank(m);
  forall(a in A)
    label(activity[a].start());
}
```

Figure B.1: A CP model

between the "fixed" variables, and not actual starting times. In particular, the idea is to extract a partial order from the current best solution and to ensure that the new solution satisfies this ordering (e.g., Carchrae and Beck (2009); Godard et al. (2005); Cesta et al. (2000); Michel and Van Hentenryck (2004)). This is captured, for flexible jobshops, by the following definitions.

Definition 2 (POS-feasible Solution)**.** *Let $\mathcal{R}$ be a set of activities to relax and (σ_o, μ_o) be a feasible solution. A solution (σ, μ) is* POS-feasible *wrt (σ_o, μ_o) and $\mathcal{R}$ if it satisfies $\sigma(a) \geq \sigma(b) + d(b, \mu(b))$ for all $a, b \in \mathcal{A} \setminus \mathcal{R}$ such that $\sigma_o(a) \geq \sigma_o(b) + d(b, \mu_o(b))$ and $\mu_o(a) = \mu_o(b)$.*

It is also desirable to fix the machines of the activities which have not been relaxed.

Definition 3 (Fully POS-feasible Solution)**.** *Let $\mathcal{R}$ be a set of activities to relax and (σ_o, μ_o) be a feasible solution. A POS-feasible (σ, μ) is fully POS-feasible wrt (σ_o, μ_o)*

and $\mathcal{R}$ if it is POS-feasible wrt (σ_o, μ_o) and $\mathcal{R}$ and satisfies $\mu(a) = \mu_o(a) \wedge \mu(b) = \mu_o(b)$ for all $a, b \in \mathcal{A} \setminus \mathcal{R}$.

The neighborhoods in this paper only consider fully POS-feasible solutions and are generalizations of those used in Carchrae and Beck (2009). They differ on the choice of the set $\mathcal{R}$ of activities to relax and whether or not some additional machine constraints are placed on the relaxed activities. Three main neighborhoods are considered:

1. *The random neighborhood*: The set $\mathcal{R}$ is a random set of activities.
2. *The time-window neighborhood*: A time window $[\alpha, \beta]$ is chosen randomly and $\mathcal{R}$ is the set of all activities in the interval $[\alpha, \beta]$.
3. *The machine neighborhood*: A set of machines is selected randomly and $\mathcal{R}$ is the set of all activities on those machines.

Moreover, three additional neighborhoods are constructed from these by selecting a subset $R \subseteq \mathcal{R}$ and imposing the constraint: $\forall a \in R : \mu(a) = \mu_o(a)$.

B.6 Adaptive Randomized Decompositions

The concept of Adaptive Randomized Decomposition (ARD) was proposed in Bent and Van Hentenryck (2010) to tackle large scale vehicle routing problems. Its aim is to find a sequence of decouplings, i.e., subproblems that can be independently optimized and whose solution can be merged back into an existing solution to produce a better solution. Formally, given an instance P of a flexible shop problem, the idea is to use the current solution π to find a decoupling (P_o, P_s) with projected solution π_o and π_s. The problem P_o is then re-optimized and its solution is merged into π_s to obtain a new solution for P. The ARD thus follows two simple principles:

1. Starting from an initial solution π_0 of P, it produces a sequence of solutions $\pi_1, ..., \pi_n$ such that the objective function $f(\pi_0) \geq f(\pi_1) \geq ... \geq f(\pi_n)$.
2. At step i, the solution π_{i-1} is used to obtain the decoupling (P_o, P_s) of P with solutions π_o and π_s. The problem P_o is then re-optimized and its solution π_o^* is used to obtain the new solution of $\pi_i = \text{MERGE}(\pi_o^*, \pi_{i-1})$.

The choice of algorithms for optimizing the sub-problems is independent from the ARD scheme. Our results were obtained by using CP and LNS algorithms.

B.6.1 Time Decomposition

The time decomposition extracts a subproblem consisting of the activities that lies within a time window $\langle s, e \rangle$.

Definition 4. *A time decomposition* $\langle \mathcal{R}^d, \mathcal{P}^d, \alpha, \beta, \gamma, \phi \rangle$ *of a solution* (σ_o, μ_o) *wrt* $\langle s, e \rangle$ *is a flexible jobshop defined over the activities*

$$\mathcal{R}^d = \{a \in \mathcal{A} | \omega_o(a) > s \wedge \sigma_o(a) < e\},$$

with precedence constraints

$$\mathcal{P}^d = \{(a,b) \in \mathcal{P} | a \in \mathcal{R}^d \wedge b \in \mathcal{R}^d\},$$

with availability constraints on the machines

$$\begin{array}{l} \gamma(m) = \min_{a \in \{a \in \mathcal{A} \setminus \mathcal{R}^d | \sigma_o(a) \geq e \wedge \mu(a) = m\}} \sigma_o(a) \\ \phi(m) = \max_{a \in \{a \in \mathcal{A} \setminus \mathcal{R}^d | \omega_o(a) \leq s\}} \omega_o(a)), \end{array}$$

and with bounds on the activity starting times

$$\alpha(a) = \begin{cases} \omega_o(b) & \text{if } \exists (b,a) \in \mathcal{P} : \; b \notin \mathcal{R}^d \\ 0 & \text{otherwise;} \end{cases}$$

$$\beta(a) = \begin{cases} \sigma_o(b) & \text{if } \exists (a,b) \in \mathcal{P} : \; b \notin \mathcal{R}^d \\ \infty & \text{otherwise.} \end{cases}$$

A feasible solution to the time decomposition satisfies all traditional constraints of the flexible shop, as well as the additional constraints:

$$\begin{array}{l} \forall a \in \mathcal{R}^d : \sigma(a) \geq \phi(\mu(a)) \\ \forall a \in \mathcal{R}^d : \omega(a) \leq \gamma(\mu(a)) \\ \forall a \in \mathcal{R}^d : \sigma(a) \geq \alpha(a) \\ \forall a \in \mathcal{R}^d : \omega(a) \leq \beta(a). \end{array}$$

The time decomposition remains essentially a flexible shop problem, and can be solved using the same algorithms. However, since the problem is now decoupled, knowledge of how the re-optimized schedule will affect the overall solution is missing. Moreover, using the makespan minimization as objective is not flexible enough as we will illustrate shortly. It is more appropriate to use an objective function that maximizes the distance between each activity and their completion time bounds, allowing a better left shift of the entire schedule.

Definition 5 (Time Decomposition Objective)**.** *The objective of a time decomposition* $\langle \mathcal{R}^d, \mathcal{P}^d, \alpha, \beta, \gamma, \phi \rangle$ *is defined by*

$$\textit{maximize} \min_{a \in \mathcal{R}^d} \min(\beta(a) - \omega(a), \gamma(\mu(a)) - \omega(a))$$

Figure B.2 illustrates the benefit of this new objective. Part (a) shows the decomposition (jobs are denoted by colors), Part (b) shows the optimized schedule using the

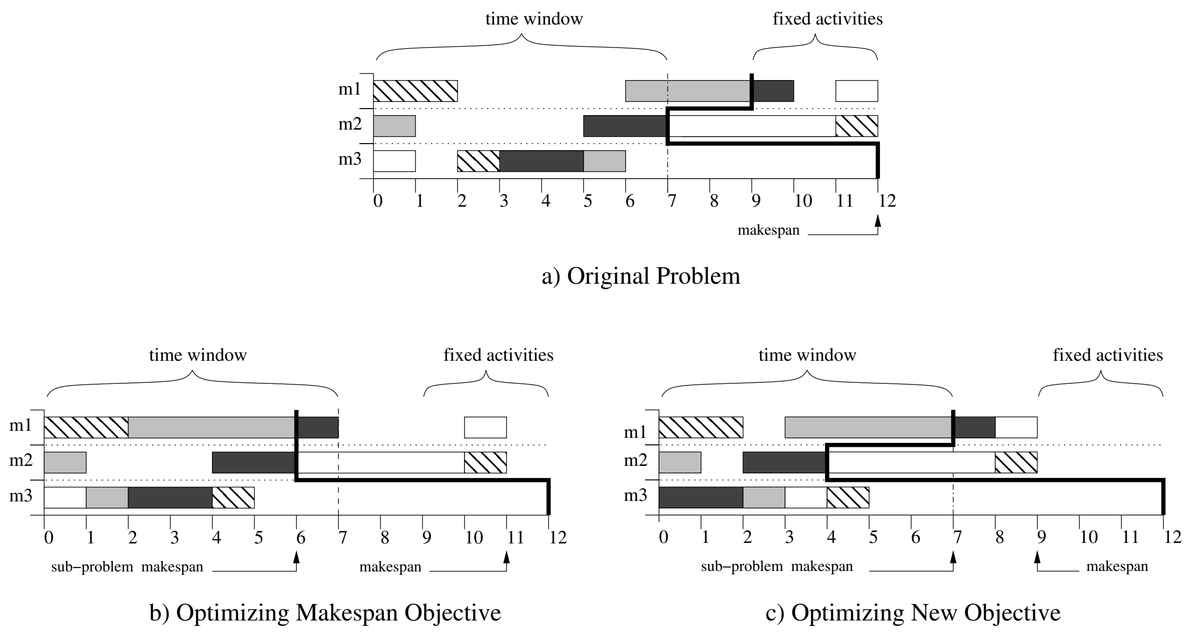


Figure B.2: Illustrating the Objective for Time Decompositions.

makespan objective, and Part (c) the schedule obtained with new objective. The new objective achieves a better makespan overall, although its local makespan is worse.

B.6.2 Machine Decomposition

The idea behind the machine decomposition is to extract a subproblem by selecting activities executing on a subset of the machines $\mathcal{M}^d$.

Definition 6. *A machine decomposition $\langle \mathcal{R}^d, \mathcal{P}^d, m^d, \alpha, \beta \rangle$ of a solution (σ_o, μ_o) wrt a set $\mathcal{M}^d$ of machines is a flexible shop defined over the activities $\mathcal{R}^d = \{a \in \mathcal{A} | \mu_o(a) \in \mathcal{M}^d\}$, with precedence constraints $\mathcal{P}^d = \{(a,b) \in \mathcal{P} | a \in \mathcal{R}^d \wedge b \in \mathcal{R}^d\}$, with bounds on the activity starting times*

$$\alpha(a) = \begin{cases} \omega_o(b) & \textit{if } \exists (b,a) \in \mathcal{P} : \; b \notin \mathcal{R}^d \\ 0 & \textit{otherwise;} \end{cases}$$

$$\beta(a) = \begin{cases} \sigma_o(b) & \textit{if } \exists (a,b) \in \mathcal{P} : \; b \notin \mathcal{R}^d \\ \infty & \textit{otherwise,} \end{cases}$$

and a reduced set of machines $m^d(a)$ for all activities $a \in \mathcal{R}^d$ defined by $m^d(a) = m(a) \cap \mathcal{M}^d$. A feasible solution to a machine decomposition satisfies all constraints of the flexible shop, as well as the additional constraints:

$$\forall a \in \mathcal{R}^d : \sigma(a) \geq \alpha(a)$$
$$\forall a \in \mathcal{R}^d : \omega(a) \leq \beta(a).$$

Since the machine decomposition has full knowledge of the activities within each machine, minimizing the makespan guarantees the generation of non-degrading solutions as long as machines on the critical path are in the set $\mathcal{M}^d$.

B.6.3 Solution Merging

Time and machine decompositions ensure that precedence and machine availability constraints are satisfied with respect to the original solution. As a result, it is possible to combine the machine assignments and the machine precedences to merge the solutions.

Definition 7. *Let (σ_d, μ_d) be a solution from the time decomposition $\langle \mathcal{R}^d, \mathcal{P}^d, \alpha, \beta, \gamma, \phi \rangle$ wrt (σ_o, μ_o) and $\langle s, e \rangle$. The merging of (σ_d, μ_d) and (σ_o, μ_o) is the solution (σ_m, μ_m) obtained by*

$$\begin{aligned} \mu_m(a) &= \mu_d(a) && \text{if } a \in \mathcal{R}^d \\ \mu_m(a) &= \mu_o(a) && \text{otherwise} \end{aligned}$$

and such that σ_m is assigned a start date minimizing the set of precedence constraints

$$\begin{aligned} &\{(a,b) | \mu_o(a) = \mu_d(b) \wedge a \notin \mathcal{R}^d \wedge b \in \mathcal{R}^d \wedge \sigma_d(b) \geq \omega_o(a)\} \cup \\ &\{(a,b) | \mu_d(a) = \mu_o(b) \wedge a \in \mathcal{R}^d \wedge b \notin \mathcal{R}^d \wedge \sigma_o(b) \geq \omega_d(a)\} \cup \\ &\{(a,b) | \mu_d(a) = \mu_d(b) \wedge a \in \mathcal{R}^d \wedge b \in \mathcal{R}^d \wedge \sigma_d(b) \geq \omega_d(a)\} \cup \\ &\{(a,b) | \mu_o(a) = \mu_o(b) \wedge a \notin \mathcal{R}^d \wedge b \notin \mathcal{R}^d \wedge \sigma_o(b) \geq \omega_o(a)\}. \end{aligned}$$

The merging is similar for the machine decomposition.

B.7 Experimental Evaluation

The proposed algorithms were evaluated on the eData set of flexible shop instances by Hurink et al. (1994). Results are reported over the set of instances *la21* to *la40* which are the largest of the set (the remaining instances being relatively easy). The algoritms were implemented on top of the COMET system and run on an Intel 2.8 GHz Xeon processor with 8Gb of RAM. Due to the non-deterministic nature of the searches, average results over 10 runs are reported.

The large neighborhood algorithms LNS and the decomposition algorithm ARD using CP or LNS for subproblems were compared to the two best-performing heuristic algorithms. We use the following notations: TB stands for the tabu search of Mastrolilli and Gambardella (2000), hGA for the hybrid genetic algorithm of Gao (2008), hLNS for the LNS search using an hybrid random selection of the proposed relaxations, ARD(CP) for the ARD procedure using CP as a search algorithm and ARD(LNS) ARD using hLNS. Both ARD(CP) and ARD(LNS) use an adaptive selection for time windows in the following sense: if no improvement is found within 5 iterations, the size of decomposition increases, only to be brought back to the initial size upon finding an improved solution.

Problem	hLNS		ARD(CP)		ARD(LNS)		TB	hGA
	5 min	10 min	5 min	10 min	5 min	10 min		
la21-25	5.48	5.48	5.77	5.57	5.61	5.59	5.62	5.60
	(6.13)	(5.98)	(6.73)	(6.55)	(6.50)	(6.25)	(5.93)	(5.66)
la26-30	3.09	2.95	3.78	3.33	3.86	3.16	3.74	3.28
	(4.41)	(4.05)	(4.89)	(4.48)	(5.13)	(4.39)	(3.76)	(3.32)
la31-35	0.88	0.46	0.45	0.32	0.62	0.32	0.30	0.32
	(1.25)	(0.97)	(1.08)	(0.89)	(1.33)	(1.03)	(0.32)	(3.32)
la36-40	8.95	8.91	10.04	9.48	9.57	9.44	8.99	8.82
	(10.09)	(9.89)	(11.52)	(11.29)	(10.95)	(10.42)	(9.13)	(8.95)

Table B.1: Quality of Solutions Obtained in 5 and 10 Minutes

The ARD(CP) and ARD(LNS) use a time decomposition chosen between 20% and 50% of the horizon, with 5% step increase. The machine decomposition uses $|\mathcal{M}|/2$ machines. The hLNS search selects activities with 50% probability in the random relaxation, time windows randomly chosen between 25% and 50% of horizon, and a number of machines randomly selected in $[2, |\mathcal{M}|/4]$ in the machine relaxation. Three additional relaxations are derived from these relaxations by fixing the machine of a relaxed activity to its current selection with a 33% probability.

Overall Quality of the Results Table B.1 depicts the quality of solutions found in 5 and 10 minutes by using the Mean Relative Error (MRE) computed as $100\%(UB - LB)/LB$. The first column describes the set of instances, while the following columns present the aggregated results for each of the algorithms, giving the best performance and showing the average performance in parenthesis. Algorithms hLNS, ARD(CP), and ARD(LNS) achieve comparable results to those found using the best heuristic methods and the best solutions of hLNS and ARD(LNS) often produce improvements over the dedicated heuristics, in particular on la21-25 and la26-30. Note also that hLNS and ARD produces results that are about 0.5% in average from the best upper bound.

These results are surprisingly good, given that the CBS algorithms use a rather simple search: They thus require very little development effort on top of a modern constraint-based scheduler (e.g., LNS adds another 50 lines of code) and could certainly be improved by using more advanced search techniques, such as texture-based heuristics Beck et al. (1997).

Table B.2 presents the best results for runs of 5 and 60 minutes for all the experiments. Bold face means an improvement over the best known upper bound (last column) while italics means that the best known upper bound has been matched. It is interesting to point out that hLNS improves or matches all the best results in 60 minutes and improves more than 50% of them. In fact, hLNS produces similar results

Problem	ARD(CP)		ARD(LNS)		hLNS		UB
	5 min	60 min	5 min	60 min	5 min	60 min	
la21	1025	**1016**	1017	**1015**	**1014**	**1009**	1017
la22	**881**	**881**	*882*	**880**	**880**	**880**	882
la23	*950*	*950*	*950*	*950*	*950*	*950*	950
la24	*909*	*909*	**908**	**908**	*909*	**908**	909
la25	*941*	*941*	942	**936**	**940**	**936**	941
la26	**1124**	**1111**	1127	**1109**	**1110**	**1107**	1125
la27	*1186*	**1182**	1189	**1182**	**1182**	**1181**	1186
la28	*1149*	**1145**	**1147**	**1144**	**1148**	**1142**	1149
la29	1122	**1117**	1129	**1111**	**1111**	**1111**	1118
la30	1219	1214	1212	**1196**	1211	**1195**	1204
la31	1541	**1533**	1554	1541	1565	*1539*	1539
la32	*1698*	*1698*	*1698*	*1698*	*1698*	*1698*	1698
la33	*1547*	*1547*	*1547*	*1547*	*1547*	*1547*	1547
la34	1609	1599	1609	*1599*	1618	*1599*	1599
la35	*1736*	*1736*	*1736*	*1736*	*1736*	*1736*	1736
la36	1174	*1162*	1167	**1160**	**1160**	**1160**	1162
la37	*1397*	*1397*	*1397*	*1397*	*1397*	*1397*	1397
la38	1156	**1143**	1159	**1143**	1148	**1143**	1144
la39	1198	1186	1187	*1184*	*1184*	*1184*	1184
la40	1167	1161	1157	**1147**	**1146**	**1144**	1150

Table B.2: Best Results For 5 and 60 Minutes Runs.

after 5 minutes, except on two benchmarks. ARD(LNS) produces relatively similar results: After a hour, it improves or matches all the best-known upper bounds except on two instances. The results after 5 minutes are a bit weaker but they still improve or match many of the best-known solutions.

Observe that hLNS explores larger neighborhoods than ARD(LNS) (since the relaxed activities can be inserted anywhere in the schedule), as well as the additional random neighborhood. It is thus not surprising that hLNS dominates ARD(LNS) on these instances. What is interesting is how close their performances are: This provides some preliminary evidence that ARD(LNS) may provide high-quality solutions quickly to large instances for which it would be too costly to reason about the schedule globally.

Impact fo the Neighborhoods We now study the impact of the various neighborhoods and decompositions on runs of 15 minutes. Figure B.3 compares the time and machine decompositions, as well as their hybridization. The results show the clear benefit of the hybridization. Figure B.4 depicts the results for large neighborhood search. For these runs of 15 minutes, they indicate that the random and time neighborhoods are most important: The machine neighborhood does not seem to bring additional benefits. Figure B.5 shows that the machine neighborhood provides improvements for longer runs: It compares LNS with and without the machine neighborhood and ARD(LNS).

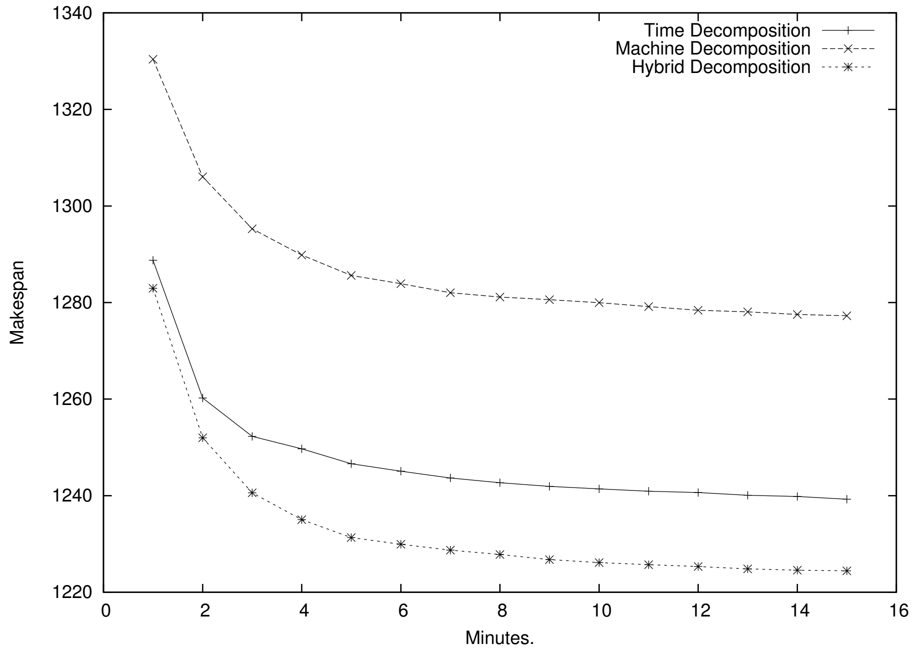


Figure B.3: The Impact of the Decompositions.

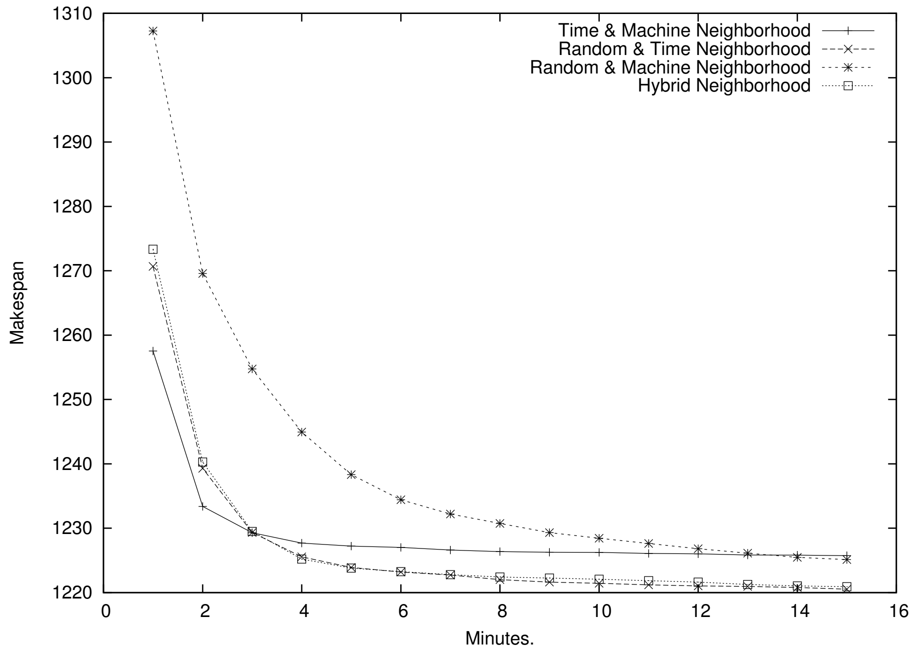


Figure B.4: The Impact of the Neighborhoods.

The results indicate that the machine neighborhood starts improving the results after 15 minutes and is necessary for LNS to dominate ARD(LNS). These results also shed some interesting light on the strength of the decomposition approach, which does not rely on the random neighborhood, giving us some reasonable confidence that it will scale nicely on large-scale instances.

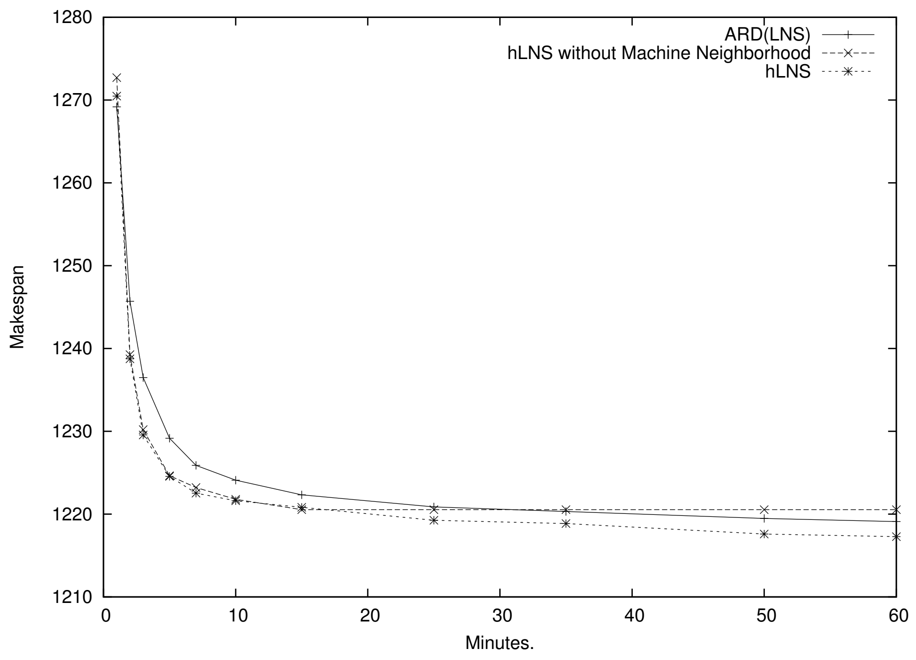


Figure B.5: The Impact of the Neighborhoods over Long Runs.

B.8 Concluding Remarks

This appendix presented large neighborhood and adaptive randomized decomposition approaches to the flexible jobshop problem. It demonstrated that a simple CBS formulation, enhanced with LNS or ARD, provides very high-quality results quickly on some standard classes of benchmarks and requires little development effort on top of a modern CBS systems. Moreover, the approaches improved 60% of the best upper bounds, while matching the remaining ones. The quality of the decomposition approach indicates that it is likely to scale to large-scale problems for which considering the problem in its entirety is not feasible. These results were achieved without advanced heuristics or learning techniques (Carchrae and Beck, 2009), suggesting that there is room for significant improvements. Overall, these results seem to confirm that LNS and ARD over a CBS formulation is an effective and general-purpose approach to many complex scheduling problems.

Appendix C

Completeness of the CBLS neighborhood

Proposition 3. *The swap neighborhood* Γ *defined by the union of* $swap^{20}$ *and* $swap^{40}$ *swaps for a slot planning problem* Π *is complete.*

Proof. We prove the claim by showing that any current assignment π of the variables of Π can be changed to an arbitrary assignment π' via a sequence of swaps. Let x^{π}_{stp} denote the value of variable x_{stp} in assignment π. Assume without loss of generality that the variables of Π are re-assigned from π to π' according to some total ordering of the cells $\prec$. Consider assigning the variables of cell (s, t) at some point in this ordered re-assignment. We have two cases:

1. $f(x^{\pi'}_{st1})$ (i.e., a 40' container must be placed in the cell). If $x^{\pi'}_{st1} = x^{\pi}_{st1}$ the cell assignment is correct. Otherwise find the container $x^{\pi'}_{st1}$ in a cell (s', t') and use $swap^{40}$ of type 1 to 5 to assign it to x_{st1}.

2. $\neg f(x^{\pi'}_{st1})$ (i.e., 20' containers or empties must be placed in the cell). If $x^{\pi'}_{st1} = x^{\pi}_{st1}$ then the aft slot of the cell is assigned correctly. Otherwise find the container (including $\perp$) in the fore slot of the cell or in another cell (s', t') and use $swap^{20}$ type 1 or 2 to assign it to x_{st1}. Do the same for x_{st2}, except only look for the container in another cell (s', t').

Because all variables of cells previous to (s, t) have already been assigned, the cells (s', t') with containers to swap into (s, t) must come after (s, t) (i.e. $(s, t) \prec (s', t')$). Since the cell (s, t) is arbitrarily chosen, we have that all cells can be re-assigned from π to π' using the resulting sequence of swaps. $\square$

Bibliography

(2010). *Storck Guide: Stowage& Segregation to IMDG Code.* Storck Verlag Hamburg.

Akella, R. and S. Gershwin (1984, September). Performance of Hierarchical Production Scheduling Policy. *IEEE Transactions on Components, Hybrids, and Manufacturing Technology 7*(3), 225–240.

Alphaliner (2012, 04). Top 100.

Ambrosino, D., D. Anghinolfi, M. Paolucci, and A. Sciomachen (2010). An experimental comparison of different heuristics for the master bay plan problem. In *Proceedings of the 9th Int. Symposium on Experimental Algorithms*, pp. 314–325.

Ambrosino, D. and A. Sciomachen (1998). A constraint satisfaction approach for master bay plans. *Maritime Engineering and Ports 36*, 175–184.

Ambrosino, D. and A. Sciomachen (2003). Impact of yard organization on the master bay planning problem. *Maritime Economics and Logistics* (5), 285–300.

Ambrosino, D., A. Sciomachen, D. Anghinolfi, and M. Paolucci (2009, March). A new three-step heuristic for the master bay plan problem. *Maritime Economics and Logistics 11*(1), 98–120.

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

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

Aslidis, A. (1990). Minimizing of overstowage in container ship operations. *Operational Research 90*, 457–471.

Aslidis, A. H. (1984). Optimal container loading. Master's thesis, Massachusetts Institute of Technology.

Aslidis, A. H. (1989). *Combinatorial Algorithms for Stacking Problems.* Ph. D. thesis, Massachusetts Institute of Technology.

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

Avriel, M., M. Penn, N. Shpirer, and S. Witteboon (1998). Stowage planning for container ships to reduce the number of shifts. *Annals of Operations Research 76*, 55–71.

Azevedo, A., C. Ribeiro, A. Chaves, G. Sena, L. Salles Neto, and A. Moretti (2012, February). Solving the 3d containership stowage loading planning problem by representation by rules and beam search. In *Proceedings of the 1st International Conference on Operations Research and Enterprise Systems - ICORES 2012*.

Barnes, W. J. and J. B. Chambers (1996). Flexible Job Shop Scheduling by Tabu Search.

Beck, J., A. Davenport, E. Sitarski, and M. Fox (1997). Texture-based heuristics for scheduling revisited. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 241–248. Citeseer.

Bent, R. and P. Van Hentenryck (2010). Spatial, Temporal, and Hybrid Decompositions for Large-Scale Vehicle Routing with Time Windows. In D. Cohen (Ed.), *Principles and Practice of Constraint Programming*, Volume 6308 of *Lecture Notes in Computer Science*, Berlin, Heidelberg, pp. 99–113–113. Springer Berlin Heidelberg.

Bona, B., P. Brandimarte, C. Greco, and G. Menga (1990). Hybrid hierarchical scheduling and control systems in manufacturing. *IEEE Transactions on Robotics and Automation 6*(6), 673–686.

Botter, R. and M. Brinati (1992). Stowage container planning: A model for getting an optimal solution. In *Proceedings of the 7th Int. Conf. on Computer Applications in the Automation of Shipyard Operation and Ship Design*, pp. 217–229.

Brandimarte, P. (1993, September). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research 41*(3), 157–183.

Brucker, P. and J. Neyer (1998, March). Tabu-search for the multi-mode job-shop problem. *OR Spektrum 20*(1), 21–28.

Brucker, P. and R. Schlie (1990, December). Job-shop scheduling with multi-purpose machines. *Computing 45*(4), 369–375.

Carchrae, T. and J. Beck (2009). Principles for the design of large neighborhood search. *Journal of Mathematical Modelling and Algorithms 8*, 245–270. 10.1007/s10852-008-9100-2.

Cesta, A., A. Oddi, and S. Smith (2000). Iterative flattening: A scalable method for solving multi-capacity scheduling problems. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 742–747. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Dauzère-Pérès, S. and J. Paulli (1997). An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search.

Delgado, A., R. M. Jensen, K. Janstrup, T. H. Rose, and K. H. Andersen (2012). A constraint programming model for fast optimal stowage of container vessel bays. *European Journal of Operational Research 220*(1), 251 – 261.

Delgado, A., R. M. Jensen, and C. Schulte (2009). Generating optimal stowage plans for container vessel bays. In *Proceedings of the 15th Int. Conf. on Principles and Practice of Constraint Programming (CP-09)*, Volume 5732 of *LNCS Series*, pp. 6–20.

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

Escudero, L. (1989). A mathematical formulation of a hierarchical approach for production planning in FMS. *Modern Production Management Systems*, 231–245.

Fan, L., M. Low, H. Ying, H. Jing, Z. Min, and W. Aye (2010). Stowage planning of large containership with tradeoff between crane workload balance and ship stability. Volume 2182, pp. 1537–1543.

Gao, J. (2008, September). A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research 35*(9), 2892–2907.

Garey, M. R. and D. S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman.

Giemsch, P. and A. Jellinghaus (2004). Optimization models for the containership stowage problem. In *Operations research proceedings 2003: selected papers of the International Conference on Operations Research (OR 2003), Heidleberg, September 3-5, 2003*, pp. 347. Springer Verlag.

Godard, D., P. Laborie, and W. Nuitjen (2005). Randomized large neighborhood search for cumulative scheduling. In *Proceedings of the International Conference on Automated Planning & Scheduling (ICAPS 2005)*, pp. 81–89.

Gumus, M., P. Kaminsky, E. Tiemroth, and M. Ayik (2008). A multi-stage decomposition heuristic for the container stowage problem. In *Proceedings of the Manufacturing and Service Operations Management (MSOM) conference.*

Hentenryck, P. and L. Michel (2009). *Constraint-based local search.* The MIT Press.

Hmida, A. B., M.-J. Huguet, P. Lopez, and M. Haouari (2007). Climbing depth-bounded discrepancy search for solving hybrid flow shop problems. *European Journal of Industrial Engineering 1*(2), 223–243.

Hmida, B. A., H. Mohamed, H. Marie-José, and P. Lopez (2010). Discrepancy search for solving flexible scheduling problems. In *2th International Workshop devoted to Project Management and Scheduling.*

Hurink, J., B. Jurisch, and M. Thole (1994, December). Tabu search for the job-shop scheduling problem with multi-purpose machines. *OR Spektrum 15*(4), 205–215.

Imai, A., K. Sasaki, E. Nishimura, and S. Papadimitriou (2006). Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. *European Journal of Operational Research* (171), 373–389.

Jensen, R., E. Leknes, and T. Bebbington (2012). Fast interactive decision support for modifying stowage plans using binary decision diagrams. In *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS'12)*, pp. 1555–1561.

Jurisch, B. (1992). *Scheduling jobs in shops with multi-purpose machines.* Ph.d. dissertation, Universitat Osnabruck.

Kaisar, E. (1999). *A Stowage Planning Model for Multiport Container Transportation.* Ph. D. thesis, University of Maryland.

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

Levinson, M. (2006). *The Box: How the Shipping Container Made the World Smaller and the World Economy Bigger.* Princeton University Press.

Li, F., C. Tian, R. Cao, and W. Ding (2008). An integer linear programming for container stowage problem. In M. Bubak, G. van Albada, J. Dongarra, and P. Sloot (Eds.), *Computational Science ICCS 2008*, Volume 5101 of *Lecture Notes in Computer Science*, pp. 853–862. Springer Berlin / Heidelberg.

Liu, F., M. Low, W. Hsu, S. Huang, M. Zeng, and C. Win (2011). Randomized algorithm with tabu search for multi-objective optimization of large containership stowage plans. In J. Bse, H. Hu, C. Jahn, X. Shi, R. Stahlbock, and S. Vo (Eds.), *Computational Logistics*, Volume 6971 of *Lecture Notes in Computer Science*, pp. 256–272. Springer Berlin / Heidelberg.

Low, M. Y.-H., M. Zeng, W.-J. Hsu, S.-Y. Huang, F. Liu, and C. A. Win (2011). Improving safety and stability of large containerships in automated stowage planning. *IEEE Systems Journal 5*(1), 50–60.

MAN Diesel & Turbo (2008, 08). Propulsion trends in container vessels.

Mastrolilli, M. and L. M. Gambardella (2000, January). Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling 3*(1), 3–20.

Michel, L. and P. Van Hentenryck (2004). Iterative relaxations for iterative flattening in cumulative scheduling. In *ICAPS04*.

Minton, S., M. D. Johnston, A. B. Philips, and P. Laird (1992). Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artif. Intell. 58*(1-3), 161–205.

Nugroho, S. (2004). Case-based stowage planning for container ships. In *International Logistics Congress*, Number December 2004, pp. 2–3.

Pacino, D., A. Delgado, and R. Jensen (2012). Modeling ballast water in container stowage planning. Accepted for presentation at the 25th European Conference on Operational Research (EURO 2012), 8-11 July 2012, Vilnius (LT).

Pacino, D., A. Delgado, R. Jensen, and T. Bebbington (2012a). An accurate model for seaworthy stowage planning with ballast tank. Under review for the 3nd International Conference on Computational Logistics, (ICCL12).

Pacino, D., A. Delgado, R. Jensen, and T. Bebbington (2012b). Large-scale container vessel stowage planning using mixed integer and constraint programming. Under review at Transportation Science.

Pacino, D., A. Delgado, R. M. Jensen, and T. Bebbington (2011). Fast generation of near-optimal plans for eco-efficient stowage of large container vessels. In J. Bse, H. Hu, C. Jahn, X. Shi, R. Stahlbock, and S. Vo (Eds.), *Computational Logistics*, Volume 6971 of *Lecture Notes in Computer Science*, pp. 286–301. Springer Berlin / Heidelberg.

Pacino, D. and P. V. Hentenryck (2011). Large neighborhood search and adaptive randomized decompositions for flexible jobshop scheduling. In *IJCAI*, pp. 1997–2002.

Pacino, D. and R. M. Jensen (2009). A local search extended placement heuristic for stowing under deck bays of container vessels. In *The 4th Int. Workshop on Freight Transportation and Logistics (ODYSSEUS-09)*.

Pacino, D. and R. M. Jensen (2010). A 3-phase randomized constraint based local search algorithm for stowing under deck locations of container vessel bays. Technical Report TR-2010-123, IT-University of Copenhagen.

Pacino, D. and R. M. Jensen (2012). Constraint-based local search for container stowage slot planning. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, pp. 1467–1472.

Pacino, D., K. Tierney, and R. M. Jensen (2012). Np-hard components of container vessel stowage planning. To be submitted to OR Letters.

Pacino, D. and P. Van Hentenryck (2011). Adaptive randomized decompositions for jobshop scheduling. *Doctoral Program of The Seventeenth International Conference on Principles and Practice of Constraint Programming (CP 2011)*, 73.

Pezzella, F., G. MORGANTI, and G. CIASCHETTI (2008, October). A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers & Operations Research 35*(10), 3202–3212.

Sciomachen, A. and A. Tanfani (2003). The master bay plan problem: a solution method based on its connection to the three-dimensional bin packing problem. *IMA Journal of Management Mathematics 14*, 251–269.

Scott, D. and D. Chen (1978). A loading model for a container ship. Technical report, Matson Navigation Company, Los Angeles.

Shields, J. J. (1984). Container stowage: A computer aided pre-planning system. *Marine Technology 21*(4).

Tupper, E. C. (2009). *Introdution to Naval Architecture*. Elsevier.

Vaessens, R. R. (1995). *Generalized job shop scheduling : complexity and local search*. doctoral thesis, Eindhoven University of Technology.

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

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

Wilson, I. D. (1997). *The Application of Artificial Intelligence Techniques to the Deep-sea Container Stowage Problem*. Ph. D. thesis, University of Glamorgan.

Wilson, I. D. and P. Roach (2000). Container stowage planning: A methodology for generating computerised solutions. *Journal of the Operational Research Society 51*(11), 248–255.

Yanbin, L., W. Kangping, G. Dongwei, P. Wei, and Z. Chunguang (2008). Multi-agent era model based on belief solves multi-port container stowage problem. In *Proceedings of the 2008 Seventh Mexican International Conference on Artificial Intelligence*, MICAI '08, Washington, DC, USA, pp. 287–292. IEEE Computer Society.

Yoke, M., H. Low, X. Xiao, F. Liu, S. Y. Huang, W. J. Hsu, and Z. Li (2009). An automated stowage planning system for large containerships. In *In Proceedings of the 4th Virtual Int. Conference on Intelligent Production Machines and Systems*.

Zhang, W., Y. Lin, and Z. Ji (2005). Model and algorithm for container ship stowage planning based on bin-packing problem. *Journal of Marine Science and Application 4*(3).