Routing and tank allocation for a chemical tanker

Mari Jevne Arnesen
Magnhild Gjestvang

Industrial Economics and Technology Management

Submission date:	June 2015
Supervisor:	Kjetil Fagerholt, IØT
Co-supervisor:	Kristian Thun, IØT

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Description of the thesis

The purpose of the thesis is to model port operations done by a chemical tanker, with the objective of finding optimal routes in a port. The problem is considering one specific ship delivering and picking up cargoes in a port, complying with restrictions regarding time windows and draft limits. Cargo handling and stowage requirements are of particular relevance for a chemical shipping company, and are also included.

The following topics will be studied:

- A tanker routing problem modeled as a pickup and delivery problem including time windows and draft limits, solved both as a MIP implemented in Xpress Mosel and by using a dynamic programming algorithm implemented in Java.
- A pickup and delivery problem including time windows, draft limits, and tank allocation, solved both exactly and by using a heuristic.
- Testing and discussion of results from generated test instances.
- An evaluation of the models and solution methods developed.

Preface

This master thesis is the final result of the work for our Master in Science in Industrial Economics and Technology Management, with specialization in Managerial Economics and Operations Research. The thesis is a continuation of our work for the project thesis, done in the fall semester 2014.

The purpose of the thesis has been to model port operations done by a chemical tanker, with the objective of finding optimal routes in a port. This master thesis is a part of the GREENSHIPRISK project, which is a collaboration between Odfjell, Western Bulk, MARINTEK, Norwegian School of Economics, and Norwegian University of Science and Technology.

We would like to thank our supervisor Professor Kjetil Fagerholt for his insights and helpful guidance. We would also like to express our gratitude to our co-supervisor, Kristian Thun, for his support and feedback, especially in the phase of developing the Java-code used in this thesis, as we had no knowledge with Java as a programming language. At last, we would like to thank our contact in the case company, Klaus Walderhaug, for his contribution of real case data and valuable input on shipping related issues.

Trondheim, June 5, 2015

Mari Jevne Arnesen   [illegible]

Mari Jevne Arnesen   Magnhild Gjestvang

Abstract

This thesis considers a chemical tanker arriving a port with several terminals where customers are to be served. The customers have both pickup and delivery requirements, and the intention is to find a feasible route for serving the customers such that the total time used in port is minimized. Chemical tankers carry chemicals in compartments, and the routing decision is complicated when these chemicals need to be allocated to specific tanks. Draft limits at the terminals and time windows for when the cargoes can be served will also influence the terminal sequence the ship can follow.

A literature study examining problems with similar characteristics is presented. There has been done relatively little research on ship routing and scheduling compared to vehicle routing, and this thesis contributes to the literature by studying a pickup and delivery problem with time windows and draft limits. This has never previously been explicitly studied. Also including tank allocation to the problem is an additional contribution to literature, as the allocation of cargoes to tanks is an important planning problem for chemical shipping companies.

Two models are developed, one without and one with tank allocation. The models are solved with two different solution methods. The model without tank allocation is solved exactly by direct implementation in a commercial mixed integer programming solver (MIP-solver), and by using a dynamic programming algorithm. The model with tank allocation is solved exactly with a MIP-solver and by using a heuristic. The heuristic consists of a dynamic programming algorithm to find ship routes and a MIP-solver to find feasible tank allocations. The computational study in this thesis includes testing of the two models using both solution methods for instances of different sizes, and analyses of the models and solution methods.

The results from the computational study show that when the problem without tank allocation is solved using a MIP-solver, only small instances are solved in a short time.

The results from the computational study show that the MIP-solver is able to solve the problem without tank allocation in a short time only when instances of small sizes are considered. The run times increase drastically when the number of cargoes increases or tank allocation is included. The dynamic programming algorithm and the heuristic shows promising results when solving the models without and with tank allocation, respectively. Both solution methods are able to solve the problems for instances of realistic sizes. These solution methods are therefore considered suitable to be a part of a tool used for decision support for chemical shipping companies.

Sammendrag

Denne avhandlingen betrakter en kjemikalietanker som ankommer en havn med flere terminaler hvor kunder skal betjenes. Skipet skal enten hente eller levere laster hos de ulike kundene, og hensikten med problemet er å finne en mulig rute slik at den totale tiden som brukes i havnen er minimert. Kjemikalietankere oppbevarer lastene i tanker, og allokeringen som må gjøres når laster skal tilordnes tanker kompliserer ruteplanleggingen. Dypgangsrestriksjoner for terminalene og tidsvinduer for når lastene kan betjenes kan også påvirke ruten skipet kan seile.

I avhandlingen presenteres en litteraturstudie hvor problemer med lignende egenskaper har blitt undersøkt. I forhold til klassisk ruteplanlegging har det blitt gjort relativt lite forskning spesifikt på maritim ruteplanlegging. Denne avhandlingen bidrar til litteraturen ved å studere et pickup and delivery-problem med tidsvinduer og dypgangsrestriksjoner. Dette problemet har ikke blitt studert eksplisitt før. Å inkludere tankallokering i modellen er et ekstra bidrag til litteraturen, da dette er et viktig planleggingsproblem for rederier med kjemikalietankere.

To matematiske modeller har blitt utviklet; en med og en uten tankallokering. Modellene er løst med to ulike løsningsmetoder. Modellen uten tankallokering er løst ved direkte implementering i en kommersiell problemløser for blandede heltallsmodeller (MIP-solver), og ved bruk av en dynamisk programmeringsalgoritme. Begge disse løsningsmetodene er eksakte. Modellen med tankallokering er løst eksakt ved bruk av en MIP-solver og ved bruk av en heuristikk. Heuristikken bruker en dynamisk programmeringsalgoritme til å finne mulige skipsruter for så å bruke en MIP-solver til å løse tankallokeringsproblemet. Beregningsstudiet i denne avhandlingen omfatter testing av de to modellene med de ulike løsningsmetodene. Testingen er gjort på instanser av forskjellige størrelser for å analysere hvordan løsningsmetodene fungerer.

Resultatene i beregningsstudiet viser at løsningsmetoden som innebærer å bruke en MIP-solver kan løse problemet uten tankallokering raskt kun for små instanser. Kjøretidene øker drastisk når antallet laster i instansene øker, eller når tankallokering inkluderes i problemet. Løsningsmetodene som bruker den dynamiske programmeringsalgoritmen og heuristikken for å løse henholdsvis problemene uten og med tankallokering viser derimot gode resultater når instansene er av realistiske størrelser. Disse løsningsmetodene vurderes derfor som lovende til å kunne være en del av et verktøy som kan brukes for beslutningsstøtte for rederier som står ovenfor lignende problemer.

Contents

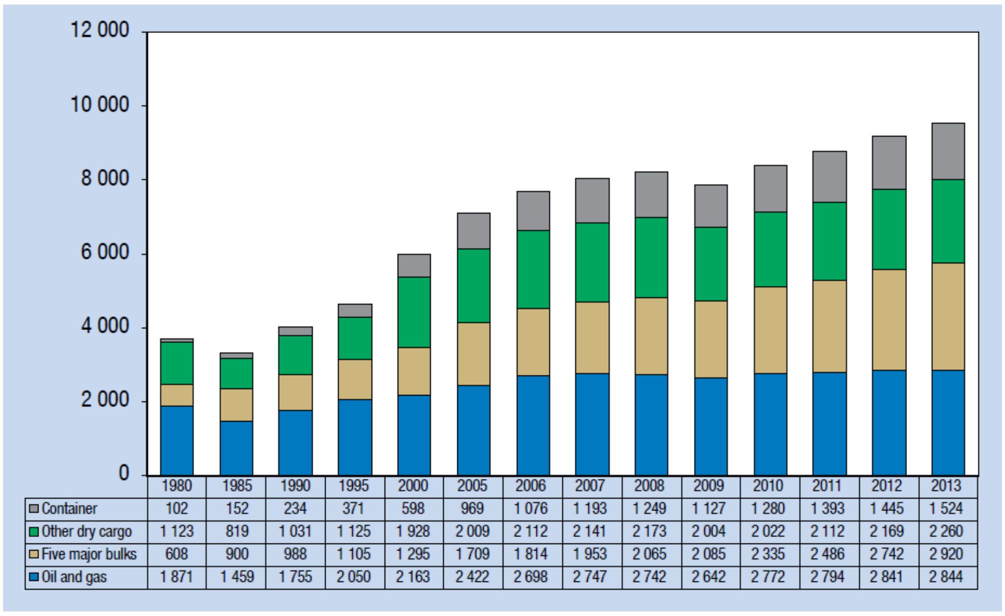
List of Tables

List of Figures

Chapter 1

Introduction

With globalization and increasing international trade and collaboration, the uneven distribution of populations and resources causes a significant need for worldwide transportation. Maritime transportation is the largest contributor to the transportation of goods, where 80% of global trade by volume is carried by sea. In 2013, about 9.6 billion tonnes of goods were carried by sea, which was an increase from 9.2 billion tonnes in 2012 (UNCTAD, 2014). Figure 1.1 shows the development in international seaborne trade. It can be seen that the seaborne trade has been experiencing growth over the last decades, and about doubled since 1995. As a result of this growth, the world fleet is expanding, reaching more than 1.69 billion deadweight tonnage in January 2014. *Deadweight tonnage* (dwt) is a measure of how much weight, in metric tons, a ship is carrying or can safely carry, including cargoes, fuel, supplies, crew and passengers.



	1980	1985	1990	1995	2000	2005	2006	2007	2008	2009	2010	2011	2012	2013
▨ Container	102	152	234	371	598	969	1 076	1 193	1 249	1 127	1 280	1 393	1 445	1 524
🟩 Other dry cargo	1 123	819	1 031	1 125	1 928	2 009	2 112	2 141	2 173	2 004	2 022	2 112	2 169	2 260
🟨 Five major bulks	608	900	988	1 105	1 295	1 709	1 814	1 953	2 065	2 085	2 335	2 486	2 742	2 920
🟦 Oil and gas	1 871	1 459	1 755	2 050	2 163	2 422	2 698	2 747	2 742	2 642	2 772	2 794	2 841	2 844

Figure 1.1: Development in international seaborne trade for selected years. In millions of tonnes loaded (UNCTAD, 2014)

In the following sections we take a closer look at aspects within maritime shipping relevant for this thesis.

Strategic, tactical and operational planning

The objective of a shipping company is to utilize its fleet in the best possible way. The companies are facing complex planning decisions that can be categorized into strategic, tactical and operational decision levels, depending on the planning horizon (Christiansen et al., 2007). Strategic decisions have a planning horizon of several years, including among others fleet size and mix problems, facility location problems and contract evaluations. At the tactical planning level, decisions with a shorter planning horizon (1 week - 1 year), such as ship routing and scheduling, inventory and production management, are assessed. Operational decisions are the decisions made for the next day or week, e.g. the speed for a sailing leg, the allocation of products to compartments, or decisions regarding whether to take spot loads or not. To be competitive, it is important for the companies to make good decisions throughout the entire planning process and recognize the interplay between the decision levels.

Modes

The routes sailed can be regional at short sea (e.g. by ferries) or global at deep sea (e.g. by ships transporting cargoes from South-America to Asia). Lawrence (1972) presented the *modes* liner, tramp and industrial shipping as a way to classify how the shipping companies operate. In liner shipping, ships are following a fixed route according to a public schedule, while trying to maximize profit. Ships in tramp shipping operate as taxis, where routes are decided on the basis of available cargoes. In addition to mandatory contracted cargoes, tramp shipping companies can carry optional cargoes trying to maximize profit. In industrial shipping, the operator both owns the cargoes and controls the ships, trying to carry all the cargoes while minimizing cost. A shipping company can operate its fleet in multiple modes, as well as transferring the ships from one mode to another depending on its strategy.

The shipping market

The shipping business involves freight of everything from oil, LNG and chemicals, to cars, food and people. The transportation of energy makes up around 50% of the weight transported by sea, and the transport of raw materials for refinement and manufacturing constitutes 30% of the weight (Amdahl et al., 2014). The remaining 20% is transportation of cars, people, food, manufactured and semi-manufactured goods, chemicals and more. The shipping market can be categorized on the basis of ship type. The most important types are crude oil tankers, production and shuttle-tankers, LPG- and LNG-ship, chemical and product tankers, bulk ships, container ships and passenger ships. Bulk shipping is defined as transportation of unpacked, large quantity cargoes in tanks. It can be transportation of both wet bulk products (oil, chemicals and oil products) and dry bulk products (iron ore, grain, coal, bauxite/alumina and phosphate), and make up more than 60% of the weight carried by sea (UNCTAD, 2014). Tanker ships can be defined as ships transporting liquids. In 2012, the total fleet comprised of 12,902 tankers totaling 547 million dwt, thereof 10,194 oil tankers (497 million dwt), 1,144 pure chemical tankers (6.2 million dwt) and 1,564 liquid gas tankers (44 million dwt) (Databases, 2012).

The chemical tanker market

The chemical tanker market is defined as the market for transportation of bulk liquid chemicals by sea (Østensjø, 1992). Compared to other tanker types, the chemical tankers are of smaller sizes due to the special cargoes carried and the restrictions of the port terminals where the tankers load and unload. The specific design of chemical tankers enables them to carry a wide range of different products. A chemical tanker is a ship constructed or adapted for carrying, in bulk, any liquid product listed in the International Bulk Chemical Code (International Maritime Organization, 2015). Figure 1.2 illustrates the main cargoes in the chemical tanker market.

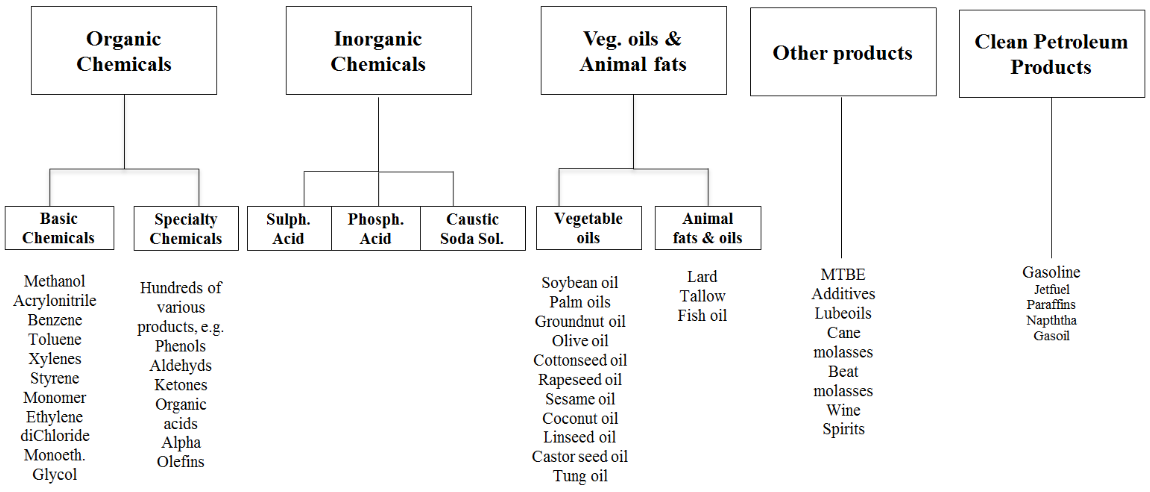

Organic Chemicals
Inorganic Chemicals
Veg. oils & Animal fats
Other products
Clean Petroleum Products
Basic Chemicals
Specialty Chemicals
Sulph. Acid
Phosph. Acid
Caustic Soda Sol.
Vegetable oils
Animal fats & oils
Methanol Acrylonitrile Benzene Toluene Xylenes Styrene Monomer Ethylene diChloride Monoeth. Glycol
Hundreds of various products, e.g. Phenols Aldehyds Ketones Organic acids Alpha Olefins
Soybean oil Palm oils Groundnut oil Olive oil Cottonseed oil Rapeseed oil Sesame oil Coconut oil Linseed oil Castor seed oil Tung oil
Lard Tallow Fish oil
MTBE Additives Lubeoils Cane molasses Beat molasses Wine Spirits
Gasoline Jetfuel Paraffins Napththa Gasoil

Figure 1.2: Main cargoes in the chemical tanker market (Odfjell, 2012)

Chemical tankers have numerous compartments to be capable of carrying the different product types, making them very flexible. Because the shipments are of smaller sizes, the tankers can transport numerous different cargoes simultaneously. The tanks have tank linings consisting of either stainless steel, zinc, epoxy or polymer coating (International Marine, 2015). The coating provides restrictions on which cargoes a tank can carry. The ship can, after tank cleaning, take different cargoes on the same tank on consecutive voyages. Figure 1.3 illustrates an example of the configuration of two different tank ships and how the tanks may be arranged, showing that the tanks can be arranged differently from ship to ship, depending on their capacities and characteristics.

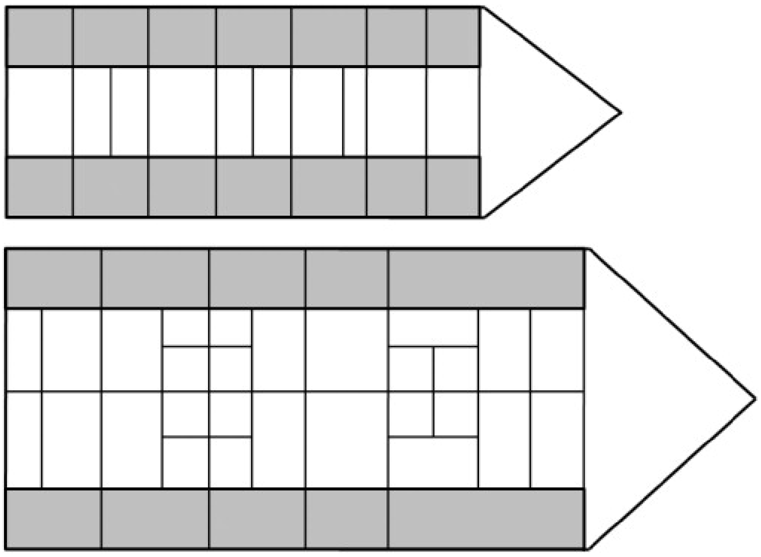


Figure 1.3: The layout of two chemical tankers (Hvattum et al., 2009)

Cargoes

The cargoes carried by sea can be classified as contracted and optional cargoes. Contracted cargoes are specified in long-term agreements, Contracts of Affreightment (COA), between the shipping company and the customer/charterer, with the purpose of commiting a carrier to transport a fixed or determinable quantity of a particular cargo over a specified period of time. Normally, the COA are not restricted to one specific ship, so the carrier have to decide which ship in the operating fleet that is going to transport the cargo. Optional cargoes, or spot loads, are cargoes the shipping company can include in the on-going trading pattern with the intention of maximizing profits. Depending on the characteristics of the spot loads and to where they are going to be transported, the shipping company must decide which ones to include and construct routes that cover the contracted and the optional cargoes.

A shipping company's objective

Chemical tanker companies may increase their profitability with appropriate planning, especially because of an increasing demand for ocean shipping capacity. When measuring commercial performance in the market, the shipping companies may use the key earnings indicator called *time charter equivalent* (TCE). This is a standard measure within the shipping industry. It is used to get a nuanced perspective of the company's earnings and to compare the performance in terms of revenue between different geographical areas, ship types and ship segments. Voyage earnings in terms of TCE are calculated using the following formula:

$$TCE = \frac{Freight\ revenue\ -\ Voyage\ specific\ costs}{Voyage\ duration} \left[\frac{USD\ (\$)}{day}\right]$$

The aim is to maximize the voyage earnings. It can be seen that one of the factors influencing earnings is voyage duration, which includes time at sea, loading and unloading time, and time for waiting and maneuvering. Liquid bulk terminals generally operate on a first-come-first-serve basis, so queuing and waiting for access to berths is a major source of uncertainty. Chemical tankers spend up to 40% of their time in port (Jetlund and Karimi, 2004), despite the fact that the trade routes are all over the world, exemplified in Figure 1.4. The time used in port is of great importance, affecting both the performance of each voyage and the total performance of the shipping company.

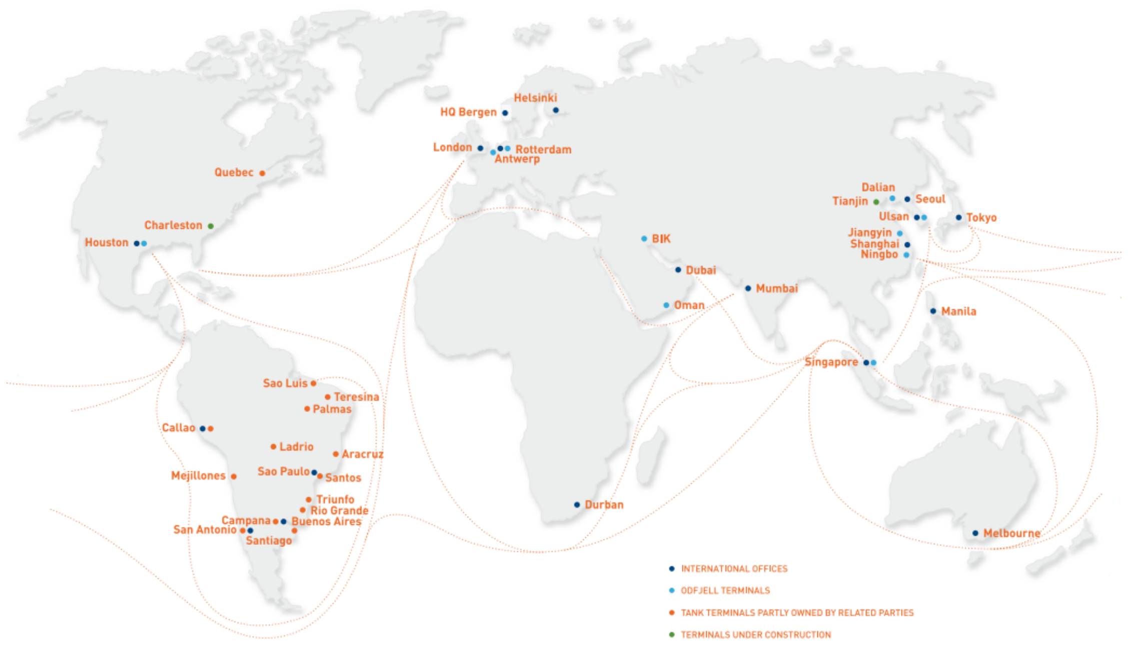


Figure 1.4: The worldwide activities of a chemical shipping company, including the regional and global trade lanes, their terminals and international offices (Odfjell, 2013)

The thesis problem

In a competitive market with increasing demand for ocean shipping capacity, chemical shipping companies are facing complex planning decisions. The companies must perform appropriate planning with regard to strategic decisions such as fleet size and composition, tactical decisions such as cargo routing and scheduling, and operational decisions made from day to day. Aspects of particular relevance to chemical shipping companies are cargo handling and stowage requirements, partly because of the strict regulation of transportation of chemicals.

The purpose of this thesis is to investigate how we can model port operations done by a chemical tanker, with the aim of making the port stay more efficient. The problem handles a ship arriving in a particular port with numerous terminals. The ship visits the terminals to load or unload cargoes. Which cargoes that are to be picked up or delivered are decided through negotiations with customers, which mean that the shipping company operates with a tramp shipping mode. The intention is to find a feasible route for serving the customers in the port, at the same time as the total time used in port is minimized. This objective is motivated by the TCE performance measure, where minimizing time in port strongly influences the voyage earnings obtained. The order in which the terminals should be visited can be seen as both a tactical and an operational decision, depending on how often the shipping company reconsiders which terminal to approach next. The routing decisions are complicated by several factors, e.g. cargo and tank restrictions, stability of the ship and limitations of the port terminals where tankers call to load or unload. This thesis aims to provide new insight in how to combine ship routing with pickups and deliveries and allocation of cargoes to compartments on board the ship.

Upcoming chapters

Chapter 2 describes the problem in more detail in order to enlighten characteristics important for the mathematical models developed. Chapter 3 is a literature review describing relevant problems related to pickup and delivery problems with tank allocation. Chapter 4 presents the two models developed for this thesis, one without tank allocation and one with. Chapters 5 and 6 present the solution methods used for solving the models without and with tank allocation, respectively. In Chapter 7, the results from the computational study are presented and discussed, before practical implications are discussed in Chapter 8. Chapter 9 presents the conclusion and future research.

Chapter 2

Problem description

In this chapter, the thesis' problem is described. First, the main characteristics are explained. Then, the different aspects of the problem are described in detail, followed by a short summary where the objective and limitations of the problem are highlighted.

Problem overview

A chemical tanker assigned to a trade line visits several ports along its way. Outside the ports, the ship moors at an *anchorage* point, waiting to get access to terminals located in the port. Based on contracts between the shipping company and different customers, the ship has a specific commitment regarding which terminals it must visit during the stay for loading and unloading of cargoes. The decision of which cargoes to pick up or deliver is therefore outside the scope of this project. However, the order in which the visits should be done is not specified, and the best routes while complying with given restrictions are to be found. Figure 2.1 illustrates a port with several terminals where the ships must wait at an anchorage point before approaching the terminals.

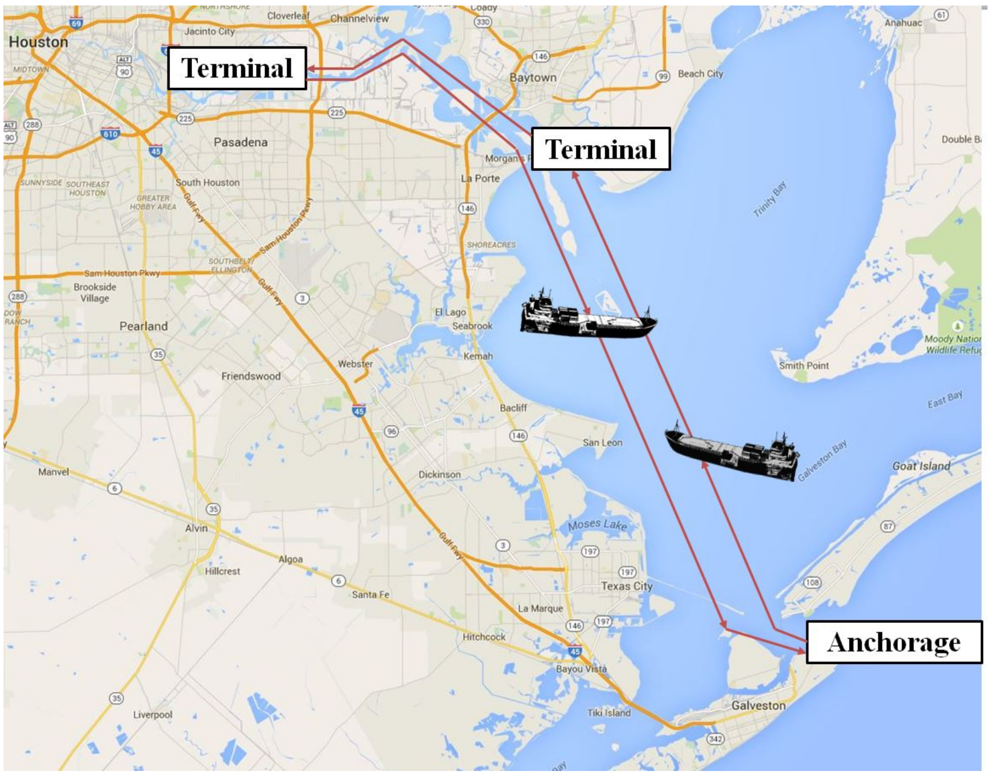


Figure 2.1: Illustration of a port call. The ship moors at an anchorage when waiting to approach the different terminals

The detailed process the ship goes through when arriving the port is central for the addressed problem. The process chart in Figure 2.2 illustrates the port operations that must be conducted. All these operations are considered when performing pickup and delivery of contracted cargoes in the port, and may affect the decision of which route the ship will take. The following sections provide a more comprehensive explanation of the relevant port operations.

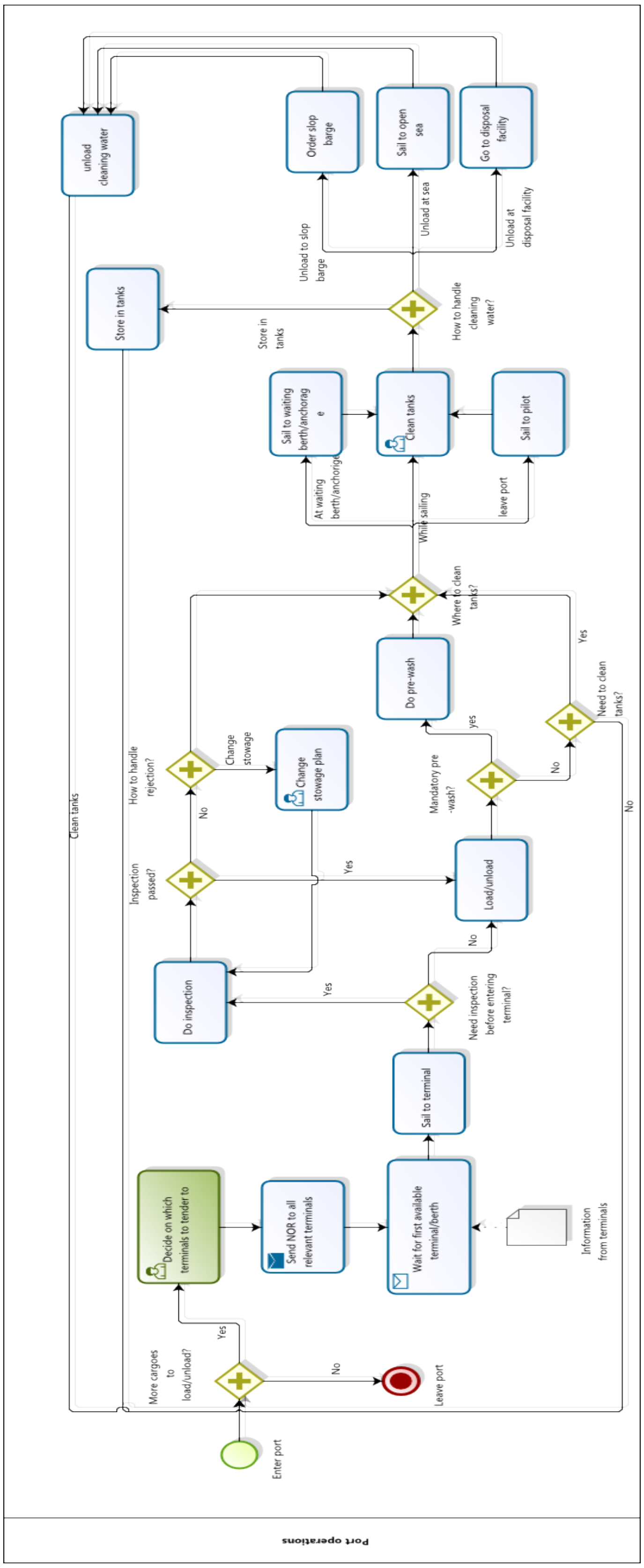


Figure 2.2: Port operations

Tendering

When the ship arrives the port, it must be decided which terminals to tender *notice of readiness (NOR)* to. A NOR can be sent out to relevant terminals if the ship is ready to load/unload the cargoes at those terminals. A ship is ready if it has space in appropriate and satisfactory cleaned tanks, if the ship does not violate cargo compatibility and ship stability restrictions, and if it does not exceed the draft limit when arriving or departing the relevant terminal.

In general, the terminals serve ships on a first-come-first-serve basis according to the time for when the notice of readiness are tendered. After receiving a NOR, the terminals may first respond with a *berth advice*. A berth advice contains assumed waiting time, which is the approximated time the ship must wait in line before it can start approaching the terminal. When the terminal is ready to accommodate the ship, it sends out a message of *berth readiness*, and the ship may start to approach the terminal. The tendering process is illustrated in Figure 2.3.

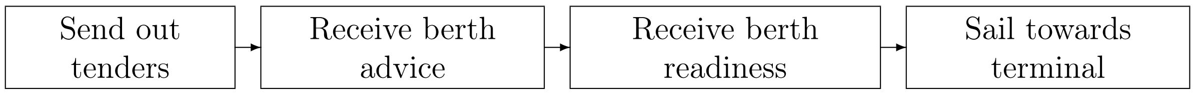


Figure 2.3: The tendering process

When a ship sails to a terminal, it loses its place in line at other terminals. The ship must therefore carefully consider which terminals it chooses to sail to. After serving that terminal, NOR needs to be tendered again. In many occasions, the ship is given access to the next terminal before leaving the current one, and it can therefore sail directly to the next terminal without sailing towards anchorage first. If the ship has not been given access it needs to sail towards anchorage until a message of berth readiness is received. The procedure of sending out tenders and deciding which terminals to sail to continues until all cargoes are loaded/unloaded. Figure 2.4 shows a terminal where a chemical tanker loads and unloads cargoes.

Figure 2.4: A terminal where chemical liquids can be loaded or unloaded (Odfjell, 2015)

Inspection

Before the ship arrives in a terminal, it may have to go through an *inspection*. Before unloading, the cargo(es) may be inspected for contamination.Before a cargo is loaded, previous cargoes, cargo types, customer requirements and tank lining influence the inspection. The inspection is usually performed by having an inspector physically entering the tanks. To check if the ship is properly washed, the inspector will either do a visual inspection of the ship, or he could do a thorough chemical analysis called a *wall wash*. A wall wash is performed by spraying methanol at random areas of the tank(s) and analyzing the residue. If the ship does not pass the inspection, it must sail out from the terminal to get satisfactory ready, i.e. by cleaning the tanks or changing stowage. Then the process of sending out tenders resumes.

Entering the terminal

The time it takes to enter a terminal depends on the ship and the terminal, and is referred to as *entering time*. If the ship is entering a narrow canal or a crowded harbor, it may have to be maneuvered by a *tugboat*, which is pushing or towing the ship. This is considered to be a part of the entering time. The following aspects are important to consider when entering the different terminals.

Time windows and laydays: All cargoes have predetermined *time windows* for when they can be served. Whether the ship complies with the specified time window or not is determined by the time the NOR is tendered. If a NOR is tendered outside the time window, the ship owner may lose the contracted cargo and the charterer may choose to cancel the freight contract and let another operator acquire the quest. The charter party foresees a number of days it takes to load or unload a particular cargo, called *laydays*. The number of laydays is described in the contract. If the charterer needs more time to load or to unload than foreseen, i.e. the laydays are exceeded, the ship is in days/hours of *demurrage*. The charterer will have to pay a certain compensation to the ship owner for this, called demurrage rate (Deseck, 2012).

Draft limits: Before approaching a terminal, the ship must assure that it is physically possible. *Draft* is the depth or height of the submerged ship, from the bottom of the ship to the waterline, measured in meters. *Draft limit* is the height from the seabed to the waterline, as seen in Figure 2.5. The ship's draft must be taken into consideration such that the terminal draft limit is not exceeded. This problem is considering one specific ship at a time. Based on the ship's weight and other characteristics, the draft limits can be expressed as a function of the weight of the cargoes on board the ship. The draft limits are thus expressed as tonnes of load on board the ship. The described problem has both pickups and deliveries, making it uncertain whether the ship's draft is largest when arriving or departing a terminal. This implies that the ship's draft must be evaluated both when arriving and departing the terminals.

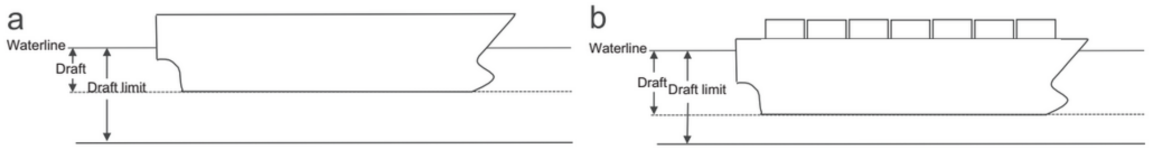


Figure 2.5: Draft and draft limit at a) an unladen ship, and b) a laden ship (Rakke et al., 2012)

Loading and tank allocation

The time it takes to load or unload a cargo is referred to as *loading time*. During loading and unloading of different products, the ship faces a planning problem regarding which tanks that should be used for which cargoes. Hvattum et al. (2009) consider the problem of allocating bulk cargoes to tanks in maritime shipping. They describe the *Tank Allocation Problem* to include aspects regarding capacity, stability and hazardous materials.

Each cargo is allocated to one or more tank(s) depending on the tank type, compatible coating, and capacity. It is not allowed to have different cargoes in the same tank, even if the cargoes are of the same product. Some of the products the tanker carries are categorized as *hazardous materials*. Several of these materials have characteristics such that they cannot be stored in adjacent tanks, and storing of different products sequentially may require cleaning in between.

One must also consider the ship's stability when allocating cargoes to tanks, because this may affect which loading sequences that are permitted. In addition to the sideways stability, which is of critical importance, the ship must consider its *trim*. This is the longitudinal stability, measured as the difference between the draft forward and aft on the ship.

Washing of tanks and noxious liquids

Some of the products the tanker carries may be *noxious liquids*. MARPOL 73/78, Annex II specifies regulations for the control of pollution by noxious liquids in bulk (International Maritime Organization, 2015). After unloading a cargo, the tank must be cleaned and the disposal water must be taken care of, in strict compliance with the requirements of MARPOL and procedures described in the approved Procedure and Arrangements Manual (P&A Manual). The time it takes to clean a tank is called the tank's *washing time*, depending on which type of cargo that was unloaded.

Substances posing a threat to the marine environment are divided into three categories, depending on the level of threat they pose. Some products require a *prewash*, and this shall be conducted before the ship leaves the terminal. The resulting residue shall be unloaded to a reception facility until the concentration of the substance in the effluent is at or below 0.1% by weight (International Maritime Organization, 2015). After a potential prewash, the tank cleaning can be conducted while sailing, or at anchorage.

The disposal water can be stored in tanks, unloaded to slop barges, unloaded at disposal facilities, or unloaded at sea. If the disposal water from different cargoes is stored on the same tanks, they must be compatible. Unload of residues to sea is prohibited unless such unloads comply with the applicable operational requirements described by MARPOL 72/78, Annex II (International Maritime Organization, 2015). If the provisions allow unload of residues it must be performed under a certain set of conditions. The conditions describe the maximum quantity of substances per tank that can be unloaded, the speed of the ship during unload, and the minimum distance from the nearest land where the unloading can happen. The conditions also describe the minimum depth of water during discharge.

Other aspects

There are a few other aspects that the ship must consider during route planning in the port. The quantity of a cargo may vary, specified in the contract. In contrast to an exact specified quantity, it could, within specified upper and lower limits, be up to either the customer or the ship owner to decide the cargo quantity. Another aspect is that several cargoes in the same terminal can be handled at the same time. This requires that there are necessary resources available in the terminal. The cleaning can also, if the necessary resources are available, be done simultaneously.

Summary

The decision of which cargoes to pick up or deliver is outside the scope of this project, as the information about which terminals to serve is given. The objective is to serve the given cargoes and to do the port operations in a sequence such that the time spent in port is minimized, leaving the routing of the ship as the main decision. Which route the ship takes is influenced by the uncertain waiting times of the terminals, inspection and possible washing of tanks on board the ship, as well as draft limits, time windows, and the tank allocation that have to be performed each time a cargo is picked up. Based on the conditions of the port operations, models are developed to provide a ship's route in a port and a plan for how to allocate cargoes to tanks.

Having a pickup and delivery problem with time windows, draft limits and tank allocation, the problem can be classified as a single-vessel PDPTW-DLTA. This is more thoroughly described in the upcoming chapter.

Chapter 3

Literature review

This chapter presents literature relevant to the problem addressed in this thesis. The literature is presented in four categories; routing and scheduling, ship routing and scheduling, stowage, and draft. It should be mentioned that the referred literature does not cover all relevant theory on these topics. Due to the many considerations presented in the problem description in Chapter 2, the problem becomes relatively complex for a small number of cargoes, and may be difficult to solve with an exact method. The literature review has been done to get a more thorough understanding of the previous research done of the aspects relevant for our problem, and the solution methods used for solving the different problems. Similarities and differences between our problem and classical routing and scheduling problems are also examined.

3.1 Routing and scheduling

The routing problems are central to distribution management and logistics (Laporte, 1992). According to Desrochers et al. (1990), vehicle routing problems include one or more vehicles, stationed at one or more depots, which have to serve a collection of customers in such a way that optimal routes are found. There are many variations of the problem, implying that different constraints are included or modified. One example is the traveling salesman problem, which only includes one vehicle.

A ship sailing a given route in a port has the same role as a single vehicle, with anchorage as a single depot where all journeys start and end. The cargoes in the terminals represent different customers that are to be served on a route. In our problem, there are both pickup and delivery cargoes, time windows for when the cargoes can be served and capacity constraints for the ship. The cargoes that are to be delivered are transported from the anchorage, and the cargoes picked up during the same route are transported back to anchorage. Each order is for one specific commodity and must be served for the particular customer requesting it, and the problem can therefore be seen as a multi-commodity problem. Given the constraints, the order in which to visit the terminals has to be decided, trying to minimize the total time used. Our problem has many similarities with the traveling salesman problem, thus it seems reasonable to start the literature review with this.

3.1.1 The Traveling Salesman Problem (TSP)

TSP includes in general a vehicle leaving from a depot, delivering goods to a set of customers exactly once before returning to the depot, trying to find a route of minimal cost. The TSP is a classical discrete or combinatorial optimization problem and belongs to the NP-complete classes, whose computational complexity rises exponentially when the number of cities increases (Geng et al., 2011). Heuristics that obtain near optimal results in a shorter processing time are therefore generally used. There are many variations of this problem, such as including multiple vehicles, time windows, capacity constraints and pickup and delivery.

One modification of the TSP is to include time windows, which applies for the problem presented in this thesis. For business organizations that work with fixed time schedules, time windows will affect the whole planning process. The traveling salesman problem with time windows (TSPTW) imposes restrictions on the permitted time interval for the start of service of the customers. Such a problem is described by Baker (1983), who presents an exact algorithm for finding the minimum time for completing the tour. The algorithm uses a branch-and-bound approach where lower bounds are obtained from the dual of a relaxation of the proposed model. Dumas et al. (1995) have developed an optimal dynamic programming algorithm for finding the minimum total traveling cost, which enhance the already existing approaches (e.g. Baker's algorithm). Here, the time window constraints are used to significantly reduce the state space and the number of state transitions.

Another modification of the problem is to incorporate capacity constraints on the vehicle, which also applies in our problem. In practice, the customers often have specifications on the amount of the goods that are to be delivered or picked up, and thus the traveling salesman will drive a vehicle with a finite capacity. This is considered by Anily and Bramel (1999), who describe a problem that includes both a vehicle with limited capacity and pickup and delivery points along the route. The problem is to determine a tour for the vehicle such that the total distance traveled is minimized and the vehicle capacity is not violated, and the authors present an approximation algorithm where the worst-case bounds are evaluated.

The literature on the traveling salesman problem generally includes determining the route, with or without constraints that make the problem more complex. A TSP may include both pickup and delivery requirements, as in the problem considered by Anily and Bramel (1999). The combination of requirements during the same tour is complicating the problem. Our problem includes both such requirements, and as pickup and delivery problems can be categorized separately, more literature about pickup and delivery problems is included.

3.1.2 Pickup and Delivery Problems (PDP)

Berbeglia et al. (2007) classify PDPs according to a simple three-field scheme consisting of structure, visits, and vehicles. This classification can be used for the problem in this thesis with the intention to place it in context with other PDPs. The structure is *one-to-many-to-one*, where some commodities are available at the depot and are destined to customers, while other commodities are to be picked up at the customers with the depot as destination. The second field provides information of whether pickup and/or delivery operations are performed. Our problem is a *P/D*, meaning that each node either has a pickup or a delivery request, but not both. This is because each node corresponds to a particular customer request. The third field specifies the number of vehicles used in the solution, and the problem described in this thesis is only considering one ship.

A similar one-to-many-to-one single vehicle pickup and delivery problem is presented by Gribkovskaia and Laporte (2008). Figure 3.1 illustrates a typical route sailed for such a problem. Customers 1 and 3 have delivery requests for cargoes A and C, respectively, while customer 2 have a pickup request for cargo B. There are several possible routes the vehicle can take, and the intention is to find the least costly route given the constraints imposed. The paper describes two variations of the problem, one having customers with either a pickup or a delivery request (P/D), similar to our problem.

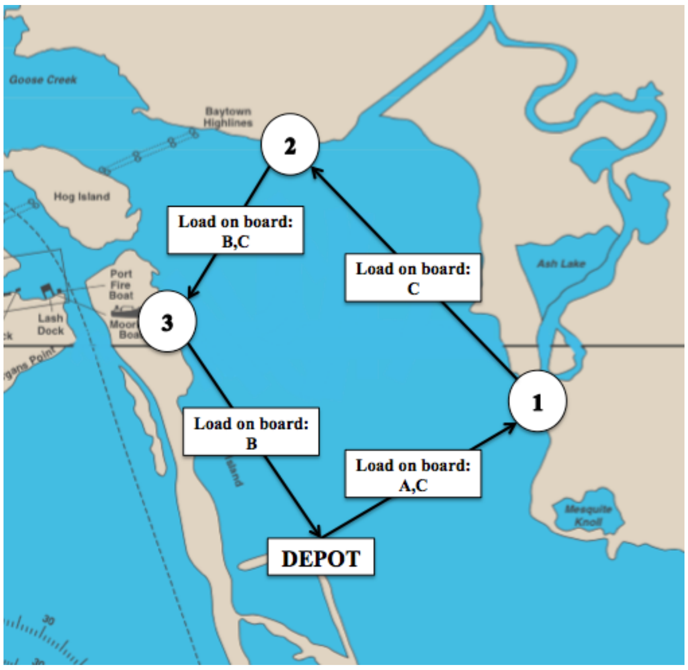


Figure 3.1: One-to-many-to-one pickup and delivery problem

Adding pickup and delivery constraints to the standard TSP may change the shortest feasible tour significantly (Mosheiov, 1994). Mladenović et al. (2012) describe a pickup and delivery traveling salesman problem (PDTSP) with one commodity. The customers either supplies or demands a given quantity of a single product, and the vehicle must visit the customers exactly once, trying to minimize the total distance travelled while not violating the capacity constraints. Our problem has several different commodities, but the other aspects are similar to the problem described above. Hernández-Pérez and Salazar-González (2009) study a problem with several commodities, where the customers are providing or requiring given amounts of different commodities. This problem may be described as a capacitated version of the TSP with precedence constraints. Even though the problem considers several commodities, it differs from our problem by having constraints associated with precedence. Psaraftis (2011) gives a generalization of the mentioned multi-commodity problem, relaxing the constraints saying that the customers can be visited only once.

Dumas et al. (1991) study the pickup and delivery problem with time windows (PDPTW). The problem considers precedence and capacity constraints, in addition to time windows on the pickup and delivery nodes. The aim of the paper is to present an exact algorithm to solve some instances of the PDPTW, and the algorithm uses a column generation scheme with a constrained shortest path as a subproblem. Another paper considering the PDPTW are the one written by Sexton and Choi (1986), but this problem differs from the one above by having soft time windows and only one single vehicle. The authors present a heuristic Benders decomposition procedure to solve the problem. Our problem has some similarities with the mentioned papers, with only one ship and time windows at each terminal. However, the problem differs by not allowing soft time windows and excluding precedence constraints.

3.2 Ship routing and scheduling

This section will provide literature about ship routing and scheduling, trying to relate the theory closer to the problem in this thesis.

Shipping refers to moving of cargoes by ships, where the ships are mostly operating in international trades. Christiansen et al. (2007) present a comprehensive overview of operations research within maritime transportation. The study is organized around the traditional planning levels strategic, tactical and operational planning, which was briefly described in Chapter 1. Tactical problems include ship scheduling and routing, where routing is the assignment of a sequence of ports to a vessel and scheduling is assigning times to the various events on a given route (Christiansen et al., 2007). Because our problem includes both routing and cargo allocation, both tactical and operational aspects are relevant.

In contrast to vehicle routing, there has been done less work on ship routing and scheduling, and Ronen (1983) discusses why ship routing and scheduling problems have been experiencing less attention. The ocean shipping industry has experienced high margins and has a long tradition for manually maneuvering the fleet. The industry is seen as conservative and not open to new ideas compared to the onshore transportation industry. Other reasons for the low attention are the variety in problem structures and operating environments, and the relatively high degree of uncertainty in the operation of ships due to e.g. differing weather conditions. However, it has been done more research on ship routing and scheduling during the last decade, because of the increasing world trade (Christiansen et al., 2013). Operating a fleet of ships is costly and significant savings can be achieved with appropriate planning, emphasizing the relevance of this thesis.

Gribkovskaia et al. (2007) present a pickup and delivery problem considering the service of offshore oil and gas platforms in the Norwegian Sea. One vessel is performing pickups and deliveries at several platforms, where all pickup demands are destined to the base and all delivery demands originate at the same base. The vessel has a capacity restriction, and the platforms have restrictions regarding space availability for loading and unloading. This is similar to our problem, where the anchorage functions as a base and the draft limits at the terminals impose restrictions of whether the ship can dock at the terminal. Another relevant paper by Fagerholt and Christiansen (2000b) presents a bulk ship scheduling problem that combines pickup and delivery with time windows and allocation of cargoes (SSAP). These aspects are of great relevance to our problem and are discussed further in Section 3.3.

Most studies on ship routing and scheduling problems discuss exact methods (Fagerholt et al., 2013). Both Fagerholt (2001) and Brønmo et al. (2006) are generating a set of feasible single ship schedules before solving a set partitioning problem to find an optimal combination, which is a general algorithm to use. For larger and more complex problem instances, this algorithm may require so much computational time that it is not usable in practical contexts. Because of this, local search-based heuristics are introduced. Korsvik et al. (2010) present tramp shipping and scheduling problems that are solved by a tabu search heuristic, while Brønmo et al. (2007) study a multi-start local search heuristic for a similar problem.

There are several models and solution techniques available for problems associated with pickup and delivery, time and capacity. Most of the literature presented includes routing and scheduling problems with different constraints, applications and solution methods, that provide a general and underlying understanding of the literature relevant for the model in this thesis. In addition, the aspects of stowage and draft are of importance, and are examined in the next sections.

3.3 Stowage

The tankers in maritime bulk shipping can be very flexible. They may carry different types of cargoes, and may combine cargoes from different customers on the same vessel. For shipping companies with these flexibility options, an important planning problem is to decide the allocation of cargoes on board the ship. This applies for chemical tankers, which are considered in this thesis.

Hvattum et al. (2009) consider the Tank Allocation Problem (TAP) faced by a maritime bulk shipping company. The problem is including constraints that most shipping companies must handle, e.g. capacity of the tanks, stability of the ship, and regulations due to hazardous materials. They present a model that finds a feasible solution to the TAP, which turns out to be computationally intractable. Øvstebø et al. (2011b) introduced the RORO(Roll-On/Roll-Off) ship storage problem, a problem deciding which cargoes to carry, how much of each cargo to include, and how to stow each cargo on board a RoRo ship, for a given route with pickup and deliveries. A standard MIP-solver and a specially designed heuristic method were presented as solutions methods. The RORO ship differs from a chemical tanker by not consisting of tanks.

Both mentioned papers consider the stowage problem for given ship routes, as opposed to when routing and stowage are done simultaneously. Some research has been done on problems combining stowage and routing, that can incorporate more elements from real planning situations. As mentioned, Fagerholt and Christiansen (2000b) consider a SSAP that combines a multi-ship pickup and delivery problem with time windows and a multi-allocation problem. In a complementary contribution, Fagerholt and Christiansen (2000a) consider a TSP with allocation, time-windows and precedence constraints. In general, the combined ship scheduling and allocation problem cannot be solved directly by commercial optimization software of problems of real size due to its complexity (Fagerholt and Christiansen, 2000a). Fagerholt and Christiansen (2000b) state that because of the relatively small and well constrained problems, ship scheduling are often approached as a set partitioning problem, and they formulate the two problems with the approach of generating feasible schedules in the subproblem.

The pickup and delivery problem with time windows (PDPTW) is NP-hard and can therefore only solve relatively small sized problems (Lu and Dessouky, 2006). Including decisions regarding a stowage plan complicates the problem further. To reduce computational time, the exact method of generating feasible schedules may be replaced with a heuristic approach. Fagerholt et al. (2013) present a tabu search heuristic for a routing and scheduling problem with stowage. The stowage problem was solved as a subproblem for a fixed route. In this thesis, a similar solution method is suggested. A heuristic is presented, performing the routing by using a dynamic programming algorithm before the stowage problem for a given route is solved with a MIP-solver. Øvstebø et al. (2011a) model a problem where routing and scheduling are considered simultaneously with stowage decisions. Here, a ship sails between two geographical regions where decisions must be made regarding the route and schedule of the ship as well as the stowage of cargo on board. It is modeled as a MIP, and additionally a tailor made heuristic was built.

3.4 Draft limits

Rakke et al. (2012) introduce the Traveling Salesman Problem with Draft Limits (TSPDL), which is a problem of determining a feasible and optimal sequence of deliveries of cargoes under draft limit constraints. A ship's draft is influenced by the amount of cargoes on board the ship, and may therefore influence which sequences that are permitted. The draft limit constraints apply for our problem, but unlike the mentioned TSPDL, customers also have pickup requirements. As described by Rakke et al. (2012), the TSPDL has never been previously studied, and related problems are rather limited. To our knowledge a pickup and delivery problem with draft limits has not been explicitly studied.

However, draft limits have been considered in many papers where other aspects are the main focus. Song and Furman (2013) present a maritime inventory routing problem (MIRP) including draft limit considerations. The draft limit will restrict the weight of cargoes on board when a ship enters and leaves the ports. Other examples of papers considering draft limits are Hennig (2010), that considers a crude oil tanker routing and scheduling problem, and Christiansen et al. (2011), that present a MIRP from the cement industry.

3.5 Implications to literature

From this literature study it has come to our attention that there has been done relatively little research on ship routing and scheduling compared to vehicle routing, and that most studies on the subject discuss exact methods. As far as we know, a pickup and delivery problem with time windows and draft limits has never been explicitly studied. A problem with pickup and delivery and time windows (PDPTW) is generally computationally hard to solve. In this thesis, the problem also includes aspects regarding draft limits and stowage, and this increases the complexity. The problem has been formulated both excluding and including tank allocation. Different solution methods have been developed based on this literature study. Both models are solved exactly aiming to examine the effect of including tank allocation and for comparison purposes. With the hope of reducing computational time, a heuristic that divides the problem in one part solving the routing problem and one solving the tank allocation problem is developed.

Chapter 4

Mathematical formulation

In this chapter, we describe two mathematical formulations. Modeling assumptions for the models are discussed in Section 4.1. In Section 4.2 a Pickup and Delivery Problem with Time Windows and Draft Limits (PDPTW-DL) is presented. A problem also considering tank allocation, a Pickup and Delivery Problem with Time Windows, Draft Limits and Tank Allocation (PDPTW-DLTA), is presented in Section 4.3.

4.1 Modeling assumptions

The waiting times are a major source of uncertainty, and it could make sense to operate with different waiting times for different days and for different parts of the day. Probability functions for the waiting times for different terminals at different points in time, approximated on the basis of historical data, could be used to model the waiting times stochastically. Historical data may include incidents not representative for the daily operation which should be considered when developing the relevant probability functions, e.g. complications appearing for another ship during loading/unloading increasing the waiting times. Finding suitable probability functions is complicated, and including stochastic parameters increases the complexity of the model. It is therefore considered reasonable to present the waiting times as given parameters, leaving us with a deterministic model.

With constant waiting times, the process of sending out tenders and receiving berth advices and berth readinesses is simplified. To illustrate the effect of receiving a berth readiness, which is sent out when the terminal is ready to accommodate the ship, it is assumed that the ship cannot sail towards a terminal before the waiting time in that terminal is over. As described in Chapter 2, the ship has either been given access to the next terminal and can sail directly towards it, or it has to sail towards anchorage until a message of berth readiness is received, illustrated in Figure 4.1. If the ship reaches anchorage before a berth readiness is received, it must wait at anchorage until it is received. The sailing time between two terminals used in this thesis is simplified to be the sum of the sailing time between the terminals and the waiting time in the terminal the ship is approaching.

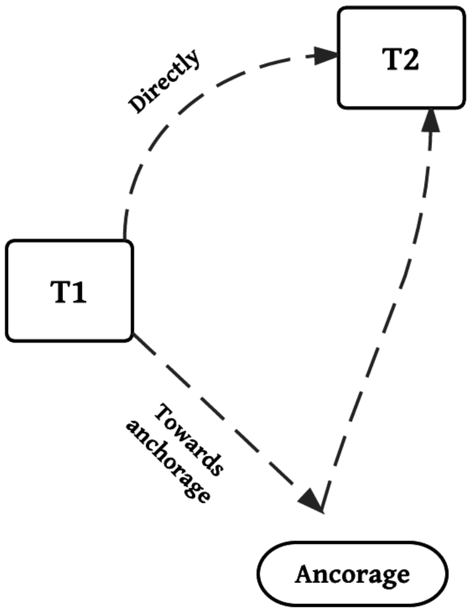


Figure 4.1: The ship can either sail directly between $T1$ and $T2$, or it has to sail towards anchorage until a message of berth readiness is received

The terminals are modeled to operate with hard time windows, meaning that there is no possibility for the ship to arrive outside the set time window. This assumes that the ship needs to wait at a terminal, or outside the terminal, if arriving too early. The ship cannot physically wait at the terminal if arriving to early, because of the presence of many other ships in the harbor. If the previous cargo the ship visited is located in the same terminal as the upcoming, it is assumed that the ship is able to wait at the terminal, not having to enter the same terminal again. There is neither a possibility for the customer to pay a demurrage rate - the charterer is always ready to accommodate the ship, at the specified number of laydays, as long as the ship arrives within the set time window.

At each terminal, the ship can load and unload cargoes. The time it takes for the ship to move from a customer in one terminal to a customer in another terminal is represented as the sum of the loading time, the washing time, the sailing time between the terminals, and the entering time in the new terminal. These components are illustrated in Figure 4.2, and is assumed to be the only time aspects needed to plan a ship route. If the ship serves two cargoes in the same terminal sequentially, the sailing time between them is zero, and the entering time is not added twice.

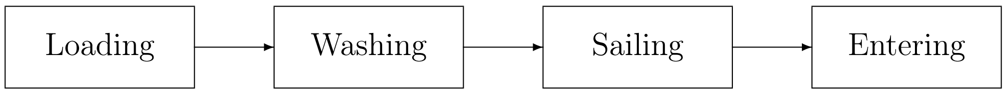


Figure 4.2: The different components of the time between two terminals

The models do not take inspection into account - the ship is always considered ready to pick up or unload cargoes. This means that the tanks on board the ship always are considered to be of satisfactory standard. The process of cleaning and disposing residues is simplified by representing it as a given number of hours for each cargo. This incurs after each unload. If one cargo is allocated to multiple tanks, as may be the case for the problem including tank allocation, it is assumed that those tanks can be washed simultaneously. The washing time for a particular cargo is added only once independently of how many tanks that is used. This also applies for the loading time. If a cargo is loaded to multiple tanks, it is assumed that it can be done simultaneously and the loading time is added only once.

It is assumed that there is no possibility to serve several cargoes in the same terminal at the same time, i.e. they must be served sequentially. The same applies for cleaning of tanks - tanks used for *different* cargoes cannot be cleaned simultaneously. Another assumption made is that each cargo quantity is fixed, i.e. that the weight of the cargoes are precisely specified in the contracts.

We choose to disregard cargoes that are not handled in the case port, meaning that the ship does not contain any cargoes other than those that are to be picked up or delivered in the port. Hence, the ship contains the sum of all delivery cargoes when entering the port, and the sum of all pickup cargoes when leaving the port, nothing more.

When cargoes are loaded and unloaded, the ship stability and trim is an important priority. This was studied by Hvattum et al. (2009) in a similar problem. Within a harbor there are fewer waves than out on the sea and the conditions are such that stability is an issue easier to handle. It is assumed that the ship has access to ballast tanks that can compensate for potentially unstable operations. Because of this, we made the assumption that the ship at all times is stable and has a reasonable trim.

4.2 A pickup and delivery problem with time windows and draft limits

The first model presented does not include the tank allocation problem. Loading and unloading of cargoes are restricted by the total capacity of the ship. The full formulation can also be found in Appendix A.1.

4.2.1 Model formulation

The problem is defined on a graph $G = (N, A)$, where N is the set of all nodes and A is the set of all arcs in the network. Cargoes are modeled as nodes where the loading/unloading happen, indexed by i and j. The number of cargoes is n. Let $N = \{0, ..., n + 1\}$ be the set of all nodes, and let $N^C = \{1, ..., n\}$ be the set of all cargoes. The anchorage is modeled as both an origin node, $i = 0$, and a destination node, $i = n + 1$, at the same geographical place at sea outside the port. A cargo is either a pickup node or a delivery node. Let $N^+ \subset N^C$ be the set of pickup nodes, and let $N^- \subset N^C$ be the set of delivery nodes.

Let the parameter T_{ij} be the time the ship uses from the start of loading/unloading cargo i to the start of loading/unloading cargo j:

$$T_{ij} = L_i + W_i + S_{ij} + E_j,$$

where L_i is the time it takes to load/unload cargo i, W_i is the washing time of cargo i if the cargo has been delivered, S_{ij} is the sailing time from node i to j, and E_i is the entering time at node j's associated terminal. The components included in the parameter T_{ij} was shown in Figure 4.2.

The sailing time S_{ij} between two nodes includes the waiting time in the next terminal. If nodes i and j are in the same terminal, S_{ij} only contains a negative entering time equal to E_j, such that the entering time for arriving a terminal is accounted for only once.

Let Q^+ be the total load in tonnes that is going to be picked up, and Q^- be the total load that is going to be delivered. Let Q_i be the weight of cargo i. The weights of the pickup cargoes are given as positive numbers, and the weights of the delivery cargoes as negative numbers. The total weight capacity of the ship is denoted by K, while D_i is the draft limit in the terminal associated with cargo i. The draft limits are represented in terms of weight, and all weights are given in tonnes.

The cargoes can be loaded/unloaded within a given time interval. Let $\underline{T_i}$ be the earliest start time for loading/unloading of cargo i, and let $\overline{T_i}$ be the latest.

The time variable t_i is the time at which the ship starts to load/unload cargo i. Arc flow variable x_{ij} is equal to 1 if the ship sails directly from node i to j, and 0 otherwise. Variable y_{ij} denotes the total weight on board the ship when sailing from i to j.

Objective function

$$minimize \quad t_{n+1}$$

The objective function minimizes the ship's completion time, which is the time the ship arrives the destination point, $n + 1$, at anchorage. When the ship arrives at anchorage, all port operations have been completed. The start time t_0 is set equal to zero, so t_{n+1} corresponds to the total time used in port.

Flow constraints

$$\sum_{j \in N} x_{0j} = 1, \tag{4.1}$$

$$\sum_{j \in N|(i,j) \in A} x_{ij} = 1, \qquad i \in N^C, \tag{4.2}$$

$$\sum_{i \in N|(i,j) \in A} x_{ij} = 1, \qquad j \in N^C, \tag{4.3}$$

$$\sum_{i \in N} x_{i,n+1} = 1, \tag{4.4}$$

Constraints (4.1)-(4.4) describe the flow along the sailing route, forcing every node to be visited.

Cargo constraints

$$\sum_{j \in N^C} y_{0j} = Q^-, \tag{4.5}$$

Constraint (4.5) ensures that the ship leaving from anchorage contains the total load that is to be delivered at all terminals.

$$\sum_{(i,j) \in A} y_{ij} - \sum_{(j,i) \in A} y_{ji} = -Q_j, \qquad j \in N^C, \tag{4.6}$$

Constraints (4.6) ensure that the difference between ingoing and outgoing shipload in each node equals the weight of the cargo that has been loaded or unloaded in the node.

Capacity constraints

$$\begin{aligned}
0 \leq y_{0j} &\leq min\{D_0, K\}x_{0j}, && (0,j) \in A \mid j \in N \setminus \{0\}, && (4.7)\\
0 \leq y_{i,n+1} &\leq min\{D_{n+1}, K\}x_{i,n+1}, && (i,n+1) \in A \mid i \in N \setminus \{n+1\}, && (4.8)\\
Q_i x_{ij} \leq y_{ij} &\leq (K - Q_j)x_{ij}, && (i,j) \in A \mid i,j \in N^+, && (4.9)\\
(Q_i - Q_j)x_{ij} \leq y_{ij} &\leq K x_{ij}, && (i,j) \in A \mid i \in N^+, j \in N^-, && (4.10)\\
-Q_j x_{ij} \leq y_{ij} &\leq (K + Q_i)x_{ij}, && (i,j) \in A \mid i,j \in N^-, && (4.11)\\
0 \leq y_{ij} &\leq (K - Q_j)x_{ij}, && (i,j) \in A \mid i \in N^-, j \in N^+, && (4.12)
\end{aligned}$$

Constraints (4.7) and (4.8) connect the x- and y-variables out from and in to anchorage, forcing the load on the ship to never exceed neither the total capacity of the ship or the draft limits at anchorage. Constraints (4.9)-(4.12) secure that the load on the ship never exceeds the total capacity of the ship.

Draft limit constraints

$$0 \leq y_{ij} \leq D_j\, x_{ij}, \qquad (i,j) \in A \mid j \in N^-, \tag{4.13}$$

$$0 \leq y_{ij} \leq D_i\, x_{ij}, \qquad (i,j) \in A \mid i \in N^+, \tag{4.14}$$

Constraints (4.13) and (4.14) impose the draft limits when arriving a delivery node and departing a pickup node, respectively.

Time constraints

$$t_i + T_{ij} - t_j - M_{ij}(1 - x_{ij}) \leq 0, \qquad (i,j) \in A, \tag{4.15}$$

Constraints (4.15) ensure that the time at which the ship starts to serve node j must be greater than or equal to the start of serving the previous node i, plus the time it takes to move from i to j. The M coefficient can be calculated as $M_{ij} = max(0, \overline{T_i} + T_{ij} - \underline{T_j})$.

$$t_i - t_0 \geq 0, \qquad i \in N \setminus \{0\}, \tag{4.16}$$

$$t_{n+1} - t_i \geq 0, \qquad i \in N \setminus \{n+1\}, \tag{4.17}$$

Constraints (4.16) ensure that node 0 is the first to be visited. Constraints (4.17) ensure that node $n+1$ is the last node to be visited.

$$\underline{T_i} \leq t_i \leq \overline{T_i}, \qquad i \in N, \tag{4.18}$$

Constraints (4.18) restrict the start time of loading/unloading at each node to be between the given time window. The time window for the origin is such that non-negativity for the time variable is imposed.

Integer constraints

$$x_{ij} \in \{0,1\}, \qquad (i,j) \in A. \tag{4.19}$$

Constraints (4.19) impose the binary requirements on the flow variables.

Subtour elimination constraints

New and additional constraints are included in the above formulation to strengthen the formulation, with the intention to reduce computational time and more effectively find a feasible solution. In the problem studied by Rakke et al. (2012), it was shown that introducing subtour elimination constraints, even though it was not necessary to the mathematical formulation, gave a significant advantage. Two-node subtour elimination constraints, shown in (4.20), are not necessary to the model, but showed reduction in the run times and are therefore added as valid inequalities.

$$x_{ij} + x_{ji} \leq 1, \qquad (i,j) \in A. \tag{4.20}$$

4.2.2 Preprocessing

The arcs in the network are defined by the set A. On the basis of problem characteristics and input data, some arcs can be considered impossible, and hence be excluded from A. This is done for all instances on the basis of the following aspects.

Anchorage is modeled as two nodes, an origin node and a destination node. The ship is suppose to visit a predefined set of nodes in the port, and thus, there are no feasible arcs between the origin and destination. There is neither a possibility to sail in to the origin node nor out from the destination node. These arcs are removed from A. Arcs between nodes of the same number are also removed, because an arc cannot go out from a node and in to the same node.

Some arcs can be removed on the basis of time windows. If the lower limit of the time window at node i, plus the time it takes to sail from that node to another node j, is larger than the upper limit of the time window for node j, the arc between them can be removed.

The draft limits also provide possibilities to remove arcs. The arc from node i to node j can be removed if node i is a pickup cargo and the draft limit in the terminal where j is located is less than the weight of cargo i. Or, if both i and j are pickup cargoes, the arc between i and j can be removed if the draft limit at j is less than the sum of the weights of cargoes i and j. Neither of these aspects applies in the test cases presented in this thesis, due to the chosen cargo quantities and draft limits. Knowing that the ship contains all delivery cargoes when leaving anchorage and all pickup cargoes when returning to anchorage, some arcs may be removed out from and in to anchorage. There are no feasible arcs between the origin and terminals with draft limits less than the total weight of the deliveries, nor between terminals and the destination in cases where the terminals have draft limits less than the total weight of the pickups.

Some arcs can be removed in order to avoid symmetrical solutions. If a pickup node, i, and a delivery node, j, are located in the same terminal, and both the lower and the upper time window of i are greater than or equal to the corresponding time windows of the delivery node j, respectively, it is always better to visit the delivery node directly before the pickup node than visiting the pickup node directly before the delivery node. This is because it could be wise to get rid of one cargo before loading another, considering ship capacity and draft limits. Hence, given that both arcs between the nodes exist, the arc from the pickup node to the delivery node can be removed, see Figure 4.3.

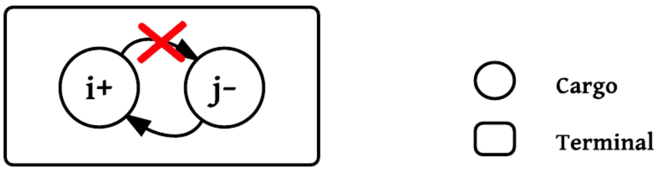


Figure 4.3: If specific requirements regarding time windows apply, the arc from the pickup node to the delivery node can be removed

If a terminal only consists of delivery nodes, more arcs can be removed in order to avoid symmetrical solutions. Consider the delivery cargoes i and j in a terminal with only delivery cargoes. If both arc (i,j) and arc (j,i) exist, and both the lower and upper limits of the time window of node i are lower than those of node j, arc (j,i) can be removed. This is shown in Figure 4.4. If the nodes are visited subsequently, it it always better or equally good to visit node i before j. If the time windows of node i and j are equal, the node with the lowest loading time is favored. If node i has a lower loading time than j, the arc (j,i) would thus be removed, given that both arc (j,i) and (i,j) exists. If the loading times also are equal, then the biggest cargo is favored. If the ship unloads the two cargoes subsequently, the biggest cargo is unloaded first.

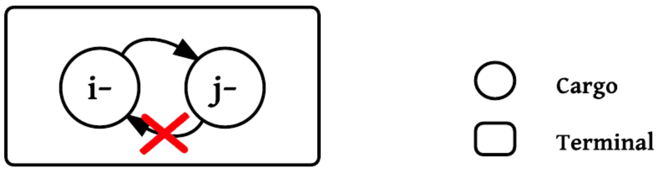


Figure 4.4: If specific restrictions regarding time windows, loading time and cargo sizes apply, one out of two arcs between two delivery nodes can be removed

If a terminal consists of only pickup nodes, symmetrical solutions can be avoided in a similar way as described in the latter paragraph. Consider the pickup cargoes i and j in a terminal with only pickup cargoes. If both arc (i,j) and arc (j,i) exist, and both the lower and upper limits of the time window of node i are lower than those of node j, arc (j,i) can be removed. This is shown in Figure 4.5. If the nodes are visited subsequently, it is always better or equally good to visit node i before j. If the time windows of node i and j are equal, the node with the lowest loading time is favored. If node i has a lower loading time than j, the arc (j,i) would thus be removed, given that both arc (j,i) and (i,j) exists. If the loading times also are equal, one can differentiate on the weights of the cargoes, and choose to load the smallest cargo first.

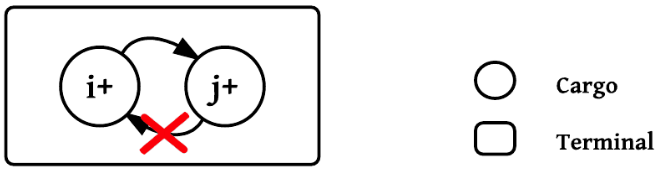


Figure 4.5: If specific restrictions regarding time windows, loading time and cargo sizes apply, one out of two arcs between two pickup nodes can be removed

4.3 A pickup and delivery problem with time windows, draft limits and tank allocation

The first formulation simplifies the fact that on a chemical tanker, the different cargoes are stored in numerous different compartments. Without considering tank allocation, it is only the total ship capacity that is assessed. In the model presented in this section, tank allocation is introduced, leading to a more precise and realistic formulation. Now, the total ship capacity is replaced with tank capacities. It is now possible to describe incompatibilities between tanks and cargoes and between the different cargoes. The tank allocation is considered simultaneously as the route planning. To provide a tank allocation plan specifying which cargoes that are allocated to which tanks in every node on the route, sequencing is included in the model.

Only the sets, parameters, variables and constraints that are in addition to those described in Section 4.2 are presented. The full formulation can be found in Appendix A.1.

4.3.1 Model formulation

The sequence number describing the number in the sequence a node is visited is given by $k \in N$. The set of tanks on board the ship is given by the set F, indexed f. Let N_i^N be the set of cargoes in conflict with cargo i. Let N_f^F be the set of cargoes compatible with tank f, and let F_i^N be the set of tanks compatible with cargo i, while F_{ijf} is the set of tanks that cannot be used for cargo j if cargo i is present in tank f. Let $\overline{K_f^F}$ and $\underline{K_f^F}$ be the maximum and minimum volume capacity of tank f, respectively.

Let V_i be the volume of cargo i. The parameter W_{if} is equal to 1 if tank f initially contains a cargo $i \in N^C$ when leaving anchorage, in $k = 0$, and equal to 0 otherwise. The parameter V_{if}^I gives the initial volume of cargo i at tank f when leaving anchorage, in $k = 0$.

The sequencing variable z_{ik} is equal to 1 if node i is visited as number k in the sequence of nodes visited, and 0 otherwise. The allocation variable w_{ifk} is equal to 1 if cargo i is allocated to tank f in sequence number k, and 0 otherwise. The load variable l_{ifk} denotes the volume of cargo i allocated to tank f in sequence number k.

Sequencing constraints

$$z_{00} = 1, \tag{4.21}$$

$$z_{j1} = x_{0j}, \qquad j \in N \mid (0, j) \in A, \tag{4.22}$$

$$\tag{4.23}$$

Constraint (4.21) ensures that the start node at anchorage is assigned the first number in the sequence, while constraints (4.22) impose the start of the sequence.

$$\sum_{k \in N} z_{ik} = 1, \qquad i \in N, \tag{4.24}$$

$$\sum_{i \in N} z_{ik} = 1, \qquad k \in N, \tag{4.25}$$

Constraints (4.24) say that a node can be assigned to only one number in the sequence. Constraints (4.25) ensure that one sequence number is assigned to only one particular node.

$$z_{jk} - z_{i,k-1} - x_{ij} \geq -1, \qquad (i,j) \in A, k \in N \setminus \{0\}, \tag{4.26}$$

Constraints (4.26) ensure that if the ship sails from i to j and node i was number $k-1$ in the sequence, node j is number k in the same sequence.

Allocating and loading constraints

$$w_{if0} = W_{if}, \qquad i \in N^-, f \in F, \tag{4.27}$$

$$l_{if0} = V^I_{if}, \qquad i \in N^-, f \in F, \tag{4.28}$$

Constraints (4.27) and (4.28) ensure that when the ship leaves anchorage, the tanks contain all cargoes that are going to be delivered in port.

$$w_{ifk} - w_{if,k-1} + z_{ik} \geq 0, \qquad i \in N^-, f \in F, k \in N \setminus \{0\}, \tag{4.29}$$

$$w_{ifk} + \sum_{q=0}^{k} z_{iq} \leq 1, \qquad i \in N^-, f \in F, k \in N \setminus \{0\}, \tag{4.30}$$

$$l_{ifk} - l_{if,k-1} + V^I_{if}(1 - w_{ifk}) \geq 0, \qquad i \in N^-, f \in F, k \in N, \tag{4.31}$$

$$l_{ifk} - V^I_{if} w_{ifk} \leq 0, \qquad i \in N^-, f \in F, k \in N, \tag{4.32}$$

Constraints (4.29)-(4.32) ensure that if a delivery node is visited, the cargo is unloaded and the tank has space for a new one.

$$w_{ifk} - w_{if,k-1} \geq 0, \qquad i \in N^+, f \in F, k \in N \setminus \{0\}, \tag{4.33}$$

$$l_{ifk} - l_{if,k-1} \geq 0, \qquad i \in N^+, f \in F, k \in N \setminus \{0\}, \tag{4.34}$$

Constraints (4.33) and (4.34) secure that if a cargo is picked up, the same cargo is on the tank the rest of the route.

$$\sum_{q=0}^{k-1} w_{ifq} + z_{ik} \leq 1, \qquad i \in N^+, f \in F_i^N, k \in N \setminus \{0\}, \tag{4.35}$$

Constraints (4.35) force w_{ifq} to be zero in every sequence until node i is visited and the cargo is picked up.

$$\sum_{f \in F_i^N} \overline{K_f^F} w_{ifk} - V_i z_{ik} \geq 0, \qquad i \in N^+, k \in N, \tag{4.36}$$

$$\sum_{f \in F_i^N} l_{ifk} - V_i z_{ik} = 0, \qquad i \in N^+, k \in N, \tag{4.37}$$

Constraints (4.36) and (4.37) secure that if a cargo is picked up, all of the cargo volume is allocated to one or more compartment(s).

$$\underline{K_f^F} w_{ifk} \leq l_{ifk} \leq \overline{K_f^F} w_{ifk}, \qquad i \in N^+, f \in F, k \in N, \tag{4.38}$$

Constraints (4.38) restrict the quantity (in volume) of cargo i on a tank f to be between the minimum and maximum capacity requirements on that particular tank.

$$y_{0j} \leq Q^{-} x_{0j}, \qquad (0,j) \in A \mid j \in N^C, \tag{4.39}$$

$$y_{i,n+1} \leq Q^{+} x_{i,n+1}, \qquad (i,n+1) \in A \mid i \in N^C, \tag{4.40}$$

$$y_{i,j} \leq (Q^{+} + Q^{-}) x_{ij}, \qquad (i,j) \in A \mid i,j \in N^C, \tag{4.41}$$

Constraints (4.39) and (4.40) connect the x- and y-variables out from and in to anchorage, forcing the load on the ship to never exceeds the total delivery or pickup load, respectively. Constraints (4.41) connect x- and y-variables between all cargoes.

$$\sum_{f \in F} l_{if,n+1} = V_i, \qquad i \in N^{+}, \tag{4.42}$$

$$\sum_{f \in F} w_{ifn} = \sum_{f \in F} w_{if,n+1}, \qquad i \in N^{+}, \tag{4.43}$$

Constraints (4.42) and (4.43) ensure that the allocation at the end node, when the ship reaches anchorage in the last sequence number, is the same as the allocation in the second last sequence number.

Compatibility constraints

$$\sum_{i \in N_f^F} w_{ifk} \leq 1, \qquad f \in F, k \in N, \tag{4.44}$$

In a given sequence, maximum one cargo can be allocated to a specific tank. This is stated in constraints (4.44).

$$\sum_{j \in N_i^N} \sum_{e \in F_{ijf}} w_{jek} - |F|(1 - w_{ifk}) \leq 0, \qquad i \in N^C, f \in F_i^N, k \in N, \tag{4.45}$$

Constraints (4.45) enforce that if cargo i is on tank f, cargo j is not stored in prohibited tanks, given by F_{ijf}. This ensures that no pair of neighboring tanks contain incompatible cargoes.

Integer and convexity constraints

$$z_{ik} \in \{0,1\}, \qquad i \in N, k \in N, \tag{4.46}$$

$$w_{ifk} \in \{0,1\}, \qquad i \in N, f \in F, k \in N, \tag{4.47}$$

$$l_{ifk} \geq 0, \qquad i \in N^+, f \in F, k \in N. \tag{4.48}$$

Constraints (4.46) and (4.47) impose binary requirements on the sequence variables and the allocating variables, respectively. Constraints (4.48) impose non-negativity requirements on the loading variables.

Chapter 5

Solution methods for the problem without tank allocation

This chapter presents two solution methods for solving the pickup and delivery problem with time windows and draft limits (PDPTW-DL), described in Section 4.2. The first solution method is using a commercial MIP-solver to solve the problem. This is only briefly described in Section 5.1, as it is a direct implementation of the problem in a MIP-solver. The other solution method is to solve the problem by using a dynamic programming (DP) algorithm. This method is well suited for solving problems like the PDPTW-DL, and it uses the structure in the problem to find solutions in a short time. Relevant literature about dynamic programming as well as a detailed description about the construction of the DP-algorithm is presented in Section 5.2.

5.1 Implementation in a MIP-solver

The PDPTW-DL can be implemented directly in a MIP-solver, defining the sets, parameters, objective function and the different constraints. A common solution method used by the MIP-solver is Branch and Bound, and the solver begins by finding the optimal solution to the LP-relaxation of the problem without the integer constraints. The branching on integer variables with non-integral solutions is repeated until a solution that satisfies all of the integer constraints is found (FrontlineSolvers, 2015).

5.2 Dynamic programming

Dynamic programming is a method of solving a complex problem, by breaking it down into a collection of simpler subproblems and solving each subproblem separately. The relations between the subproblems are used to find a solution to the original problem. This solution method is usually used for solving a shortest path problem with resource constraints or one of its variants, which often occur as a subproblem for a wide variety of vehicle routing problems (Irnich and Desaulniers, 2005). The subproblem generates routes complying with path and resource structural constraints. The solution approach systematically build new paths on a graph whose nodes are the states representing the nodes in the network, and the arcs represent transitions from one state to another.

Dumas et al. (1991) and Dumas et al. (1995) have proposed forward dynamic programming algorithms to solve a constrained shortest path problem and a TSPTW, respectively. Both problems are trying to find the path with the minimum total traveling cost, building partial paths in the graph which are extended along the arcs to construct all feasible paths. Unlike Ropke and Cordeau (2009), who seek the shortest path from a source node to a sink node without forcing the vehicle to visit every node, these problems require the least-cost feasible path that visits every node exactly once, similar to our problem. The optimal solution to the problem in this thesis is a visiting sequence for a given set of nodes such that the ending time, and not the traveling cost, of the path is minimized. This is similar to the problem described by Fagerholt and Christiansen (2000a), but here the allocation of cargoes is included in the DP-algorithm, forcing allocation feasibility at each step of the visiting sequence. This is in contrast to the DP-algorithm constructed in this thesis, because the algorithm only considers route generation and not tank allocation.

In all the articles presented in the latter paragraph, a number of domination and elimination tests are presented with the aim of speeding up the algorithms, reducing the state spaces, and thus the memory space used. This shows the importance of including such a test in a dynamic programming algorithm. Both Dumas et al. (1991) and Dumas et al. (1995) are using a criterion with the name "the same location criterion", where the algorithm can be accelerated by removing a set of arcs between nodes which are at the same physical location, but with different time windows. Our problem has several nodes in the same terminal with difference time windows, implying that this criterion may be appropriate to include in the algorithm.

In this thesis, a forward dynamic programming algorithm is developed with similarities to the mentioned papers described above. The optimal solution to the algorithm is the visiting sequence for a given set of nodes such that the completion time of the path is minimized, without considering tank allocation. In the following sections, the dynamic programming algorithm is presented, including a pseudocode and a more detailed description of how it is implemented.

5.2.1 Dynamic programming formulation

Using the same notation as in Section 4.2, the problem is defined on a graph $G = (N, A)$, where N is the set of all nodes and A is the set of all arcs in the network. The nodes corresponds to cargoes, indexed by i and j. Associated with each node is a time window $[\underline{T_i}, \overline{T_i}]$ and a weight of cargo i, Q_i, which is negative if it is a delivery cargo and positive if it is a pickup cargo. Associated with each arc is T_{ij}, the time the ship uses from the start of loading/unloading cargo i to the start of loading/unloading cargo j. Now, it is also useful to distinguish between the terminals where the cargoes must be picked up or delivered. Let B be the set of terminals, indexed b. Let D_b be the draft limit in terminal b, in terms of weight of the cargoes on board. The total ship capacity is, as before, denoted by K.

The technique used to solve the problem is a forward dynamic programming algorithm. It starts at node 0 and extends to all nodes, and hence is developing feasible paths through the network. Each partial path is described by a label $\mathcal{L}_a(i, t_i, l_i, N^V, l_{ID})$, where a is an unique label ID. Let i be the current node and let t_i be the time at which the ship starts to load/unload at that node. The accumulated load after serving cargo i is denoted l_i, and N^V is the set of all visited nodes given as the sequence in which the nodes are visited. Let l_{ID} be a unique number associated with label $\mathcal{L}$, where $l_{ID} = \sum\limits_{i \in N^V} 2^i$.

The optimal solution is the route using the least amount of time, with the requirement of being a feasible path starting in node 0, ending in node $n+1$, and visiting each node in the network exactly once. The feasible paths found must satisfy resource requirements between the starting and ending node.

The arrival time at a node j, coming from node i, is calculated as max a_j, $t_i + T_{ij}$. This implies that the ship needs to wait at a terminal, or outside the terminal, if arriving too early. The accumulated load at j is calculated by adding the load at j to the accumulated load from the previous node, $l_j = l_i + Q_j$. When leaving node 0, the accumulated load

l_0 equals the total delivery load, Q^-. The accumulated load when arriving the end node, l_{n+1} equals the total pickup load, Q^+.

During the execution of the algorithm, the labels with paths where not all nodes are visited are stored in the set of *unprocessed labels*, $\mathcal{U}$. If all nodes are visited, the label is stored in $\mathcal{C}$, a set containing the *best completed label.* $\mathcal{C}$ is dynamically updated if the algorithm finds completed routes with a better solution with respect to time, than the one already stored in $\mathcal{C}$.

The algorithm consists of two central procedures, the label extension step and the dominance step, briefly described in Algorithm 5.1. Before the DP-algorithm is run, the same preprocessing as described in Section 4.2.2 has been done in order to reduce the number of feasible arcs.

In the label extension step a label $\mathcal{L}_a \in \mathcal{U}$ is chosen to be extended. All feasible extensions with respect to resource and path-structural constraints are constructed, and $\mathcal{L}_a$ itself is removed from $\mathcal{U}$. Thus, the extension step replaces one label from $\mathcal{U}$ by all of its feasible one-node extensions. A breadth-first strategy is used to decide which label to extend, where one of the labels with the fewest nodes visited is chosen. Among these, the label with the shortest time used is chosen. After extending a label, a dominance check is run against all other labels, and dominated labels are removed from $\mathcal{U}$. The reason the labels with the fewest visited nodes are extended first, is that it is convenient that dominated labels are removed as early as possible to prevent them from being extended further. The reason the label with the shortest time is chosen, is that this is more likely to dominate other labels, and you might avoid comparing labels that both would have been dominated by this label. After extending a label to all feasible nodes, the label is removed from $\mathcal{U}$ and replaced by the new, not dominated, labels. The decision of whether a path should be extended to a certain node is done on the basis of a set of rules regarding time windows, draft limits and ship capacity, see Section 5.2.2. Let $\mathcal{G}$ be the set of these rules.

The domination step is done after every time a label has been extended. The new labels extended from $\mathcal{L}_a$ are stored in the set of *extended labels*, $\mathcal{E}_a$. These labels are compared to labels in $\mathcal{U}$ where the current node is in the same terminal as the current node in the extended label, and have visited exactly the same nodes, but not in the same order. If possible, the dominance algorithm reduces the sets $\mathcal{E}_a$ and $\mathcal{U}$, and both sets are therefore changed dynamically throughout the execution of the algorithm. The set of rules used for deciding which labels dominate others are described in Section 5.2.3.

Algorithm 5.1 Generic Dynamic Programming Algorithm

```
(* Initialize *)
Calculate $\mathcal{L}_0$
set $\mathcal{U} = \{\mathcal{L}_0\}$
while $\mathcal{U} \neq \emptyset$ do
    (* Label extension step *)
    choose a label $\mathcal{L}_a \in \mathcal{U}$ and remove $\mathcal{L}_a$ from $\mathcal{U}$
    set $\mathcal{E}_a = \emptyset$
    if $N^V(\mathcal{L}_a) \neq N$, i.e. all customers are not visited, then
        for all arcs $(i(\mathcal{L}_a), j) \in A$ where the extension is complying with rules in $G$ do
            extend to node $j$ and then add the new label $\mathcal{L}_{new}$ to $\mathcal{E}_a$
    else
        extend to end node, $n + 1$, and then add the new label $\mathcal{L}_{new}$ to $\mathcal{C}$ if $\mathcal{L}_{new}$
        dominates the label already in $\mathcal{C}$
    (* Dominance step *)
    for all labels $\mathcal{L}_{new} \in \mathcal{E}_a$ do
        apply dominance algorithm to $\mathcal{L}_{new}$ and all labels from $\mathcal{U}$ ending in the same
        terminal $b$ and with the same $l_{ID}$
        if label from $\mathcal{U}$ dominates $\mathcal{L}_{new}$ then
            remove $\mathcal{L}_{new}$ from $\mathcal{E}_a$
        else if $\mathcal{L}_{new}$ dominates label from $\mathcal{U}$ then
            remove label from $\mathcal{U}$
    add $\mathcal{E}_a$ to $\mathcal{U}$
(* Solution step *)
return $\mathcal{C}$, i.e. the label with the minimum time used for completing the route
```

Several remarks to the pseudocode should be made.

1. If only the label extension step is performed, and not the domination step, the algorithm computes all feasible paths.
2. The dominated labels are removed from their corresponding sets at the end of the domination step, but before they again are processed in the label extension step. A check is done when comparing two labels to examine if one of the labels already is dominated (but not yet removed). The domination step is not done for labels already dominated.
3. If all nodes are visited, i.e. the last extension to be done is to the end node, the label is stored in $\mathcal{C}$. Every time a label is extended to the end node, this label is compared to the one already stored in $\mathcal{C}$. If the solution is better, this label is the new label stored in $\mathcal{C}$, and the one stored is deleted. By doing this, $\mathcal{C}$ always consists of the best path with respect to time throughout the execution of the algorithm.

Sections 5.2.2 and 5.2.3 describe the criteria that must be satisfied in order to extend a path and execute the domination criteria, respectively. These rules are developed with the intention of reducing the state space and to accelerate the algorithm and the overall labeling procedure.

5.2.2 Path extensions

The decision of whether a path should be extended to a certain node is done on the basis of a set of rules regarding time windows, draft limits and ship capacity. The next sections present these rules, how they are implemented, and the aim of using them. The first three rules are mandatory constraints that *must* be satisfied in order to extend to a new node. The remaining are rules made to reduce symmetry and prevent extensions that later give infeasible solutions, with the aim of reducing the run time of the algorithm.

Rule 1: Stopping rule

This rule examines if there exist nodes in the network, excluding the end node, that still have not been visited. If this is the case, the code iterates over the nodes not yet visited. For each possible new extension the rules below are run. If the end node is the only node remaining, the path is extended to the end node. The new label can be stored in $\mathcal{C}$ if the completion time is less than the time of the path already stored in $\mathcal{C}$. The end node is never restrictive with regard to draft limits or time windows, so this extension is always possible.

Rule 2: Time window rule

This is a rule that examines the remaining nodes after possibly extending from the current node i to the new node j. If the arrival time at j plus the time between the node j and one of the remaining nodes exceeds the upper limit of the time window of any of the remaining nodes not yet visited, then the path is infeasible and the extension should not be done.

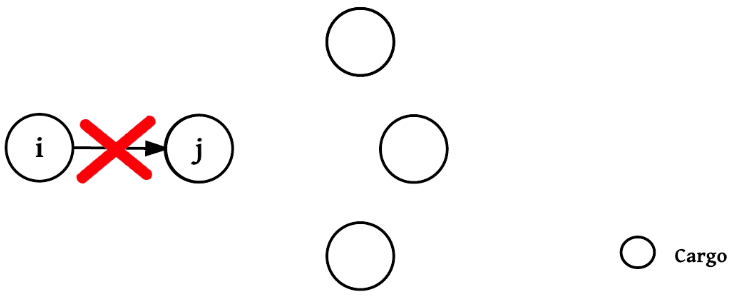


Figure 5.1: If the arrival time at j plus the travel time to any of the remaining nodes exceeds the upper time window limit of any of the remaining nodes not yet visited, the extension to j should not be done

Rule 3: Draft limit rule

This is a rule to ensure that the draft limit constraints are satisfied. The rule distinguishes between whether the new node, j, that is the one considered to be extended to, is a pickup or a delivery node. If the new node, j, is a pickup cargo, the weight of the cargoes after loading the pickup cargo is considered. This weight must be less than the minimum of the draft limit at j's associated terminal and the ship capacity. If not, the extension should not be done. If j is a delivery node, the total weight of the cargoes on board before delivering this cargo is considered. This weight must be less than the minimum of the draft limit at j's associated terminal and the ship capacity.

Rule 4: Reducing symmetry

This is a rule that is done with the aim of reducing symmetrical solutions. Let i be the current node and let j be the possible new extension considered, and let k be a node in the same terminal as j. The test iterates over all nodes k in the same terminal as j. If k is a delivery node, j is a pickup node, both the lower and upper limits of the time window at k are lower than the ones at j, k is not yet visited, and the arc between i and k exists, it is always better or equally good to visit k before j. Hence, we can forbid the extension to node j.

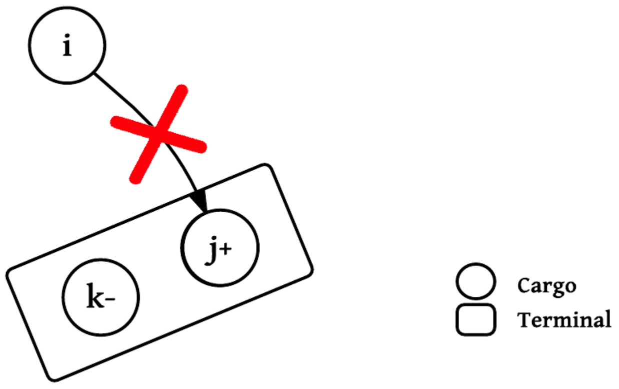


Figure 5.2: If both time window limits at delivery node k are lower than those at pickup cargo j, delivery cargo k should be served before j

Rule 5: Draft limit rule for special case I

This is an extended draft limit rule that examines the remaining nodes after possibly extending from the current node i to the new node j. The rule only applies if the remaining nodes after visiting the new node are all pickup nodes. If, for any of the remaining pickup nodes, the load at that node plus the accumulated load at j exceeds the draft limit at that pickup node, an extension to j should not be done.

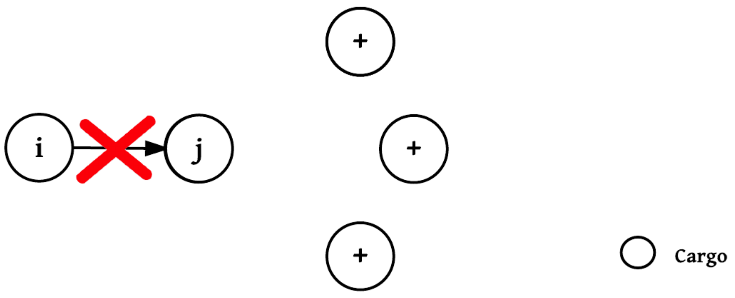


Figure 5.3: If the accumulated load at j plus the load of any of the remaining pickup cargoes exceeds the draft limit at that pickup node, the extension to j should not be done

Rule 6: Draft limit rule for special case II

This is a test that examines draft limits in a special case. Let i be the current node and let j be the possible new extension considered. The remaining nodes after visiting j comprise of one delivery node k and the rest, if any, are pickup nodes. If the accumulated load after visiting j exceeds the draft limit at k, then the extension to j should not be done.

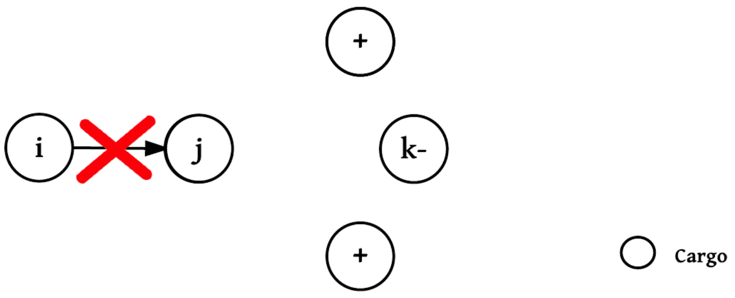


Figure 5.4: If the accumulated load at j exceeds the draft limit at delivery cargo k, the extension to j should not be done

Rule 7: Draft limit rule for special case III

This is a test that examines draft limits in another specific case. Let i be the current node and let j be the possible new extension considered. If j is a pickup node, and the accumulated load at j plus the sum of the weights of the remaining delivery cargoes exceeds the draft limit of any of the remaining pickup cargoes, the extension to j should not be done. Hence, even after serving all of the delivery cargoes, if the draft limit at one of the remaining pickup nodes is violated, the extension should not be done.

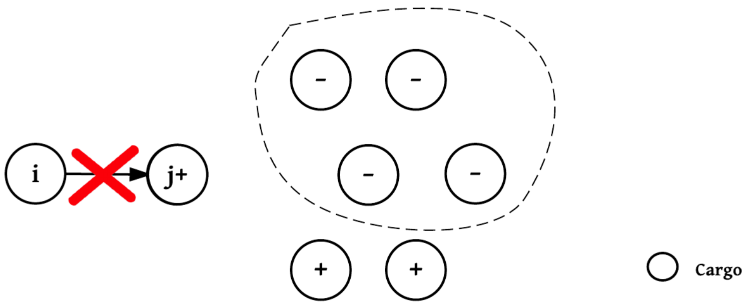


Figure 5.5: If the accumulated load at j plus the sum of the weights of the remaining delivery cargoes exceeds the draft limit of any of the remaining pickup cargoes, the extension to j should not be done

Rule 8: Draft limit rule for special case IV

This is another test examining draft limits in a specific case. Let i be the current node and let j be the possible new extension considered. If j is a pickup node, all the remaining nodes after visiting j are delivery nodes, and none of them have a draft limits larger than the accumulated load after visiting j, then the extension to j should not be done.

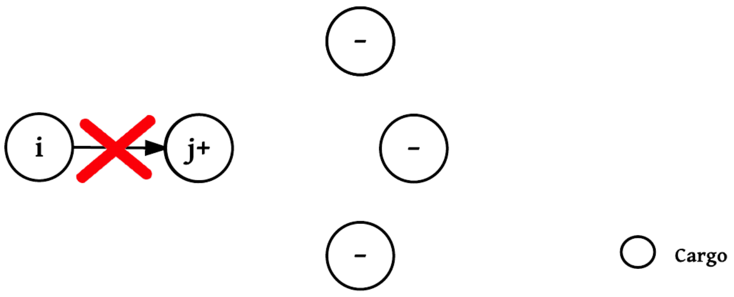


Figure 5.6: If the accumulated load at j exceeds every draft limit of the remaining delivery cargoes, the extension to j should not be done

Rule 9: Draft limit rule for special case V

This is a draft limit rule for when there are only two nodes left. Let i be the current node, let j be the possible new extension considered, and let k be the last node to be visited. If j is a pickup cargo, and the draft limit at k is less than the total weight of the pickup cargoes, the path is infeasible. Hence, i should not be extended to j.

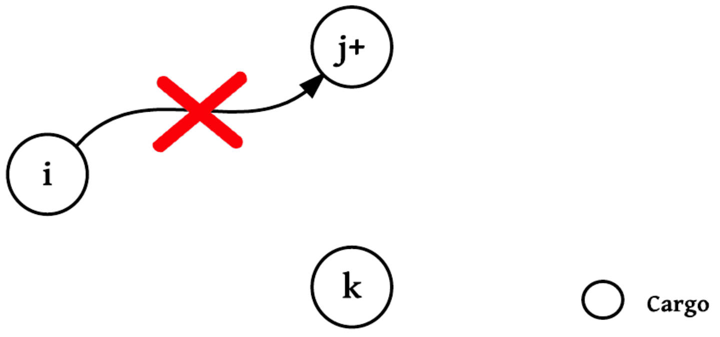


Figure 5.7: If k is the last remaining cargo after this extension and the draft limit at k is lower than the total pickup load, the extension to j should not be done

Rule 10: Draft limit rule for special case VI

This rule is a draft limit rule for a case when extending to a terminal with only pickup nodes left to visit. Let i be the current node and let j be the possible new extension considered. If the draft limits at none of the remaining unvisited nodes are bigger or equal than the accumulated load in j, the extension to j should not be done.

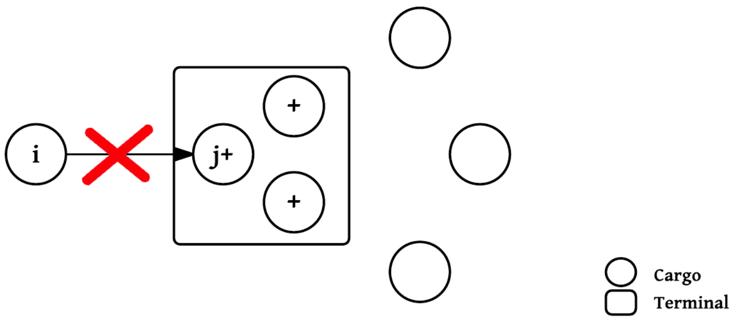


Figure 5.8: If j is in a terminal with only pickup cargoes and the accumulated load at j exceeds the draft limit at any of the remaining unvisited nodes, the extension to j should not be done

Rule 11: Loading rule for delivery cargoes

This and the following rule are made to influence in which sequence the nodes in a path are visited. Let i be the current node and let j be the possible new extension considered. This rule applies if all nodes in the terminal associated with j, including j, are delivery nodes. If there exists a node in the same terminal as j, call it k, where the arc (i,k) exists, both of k's lower and upper time window limits are lower than those of j, the loading time at k is lower than the time at j and the weight of cargo k is bigger than the weight of cargo j, k should be visited before j. Forbidding a path extending to j excludes a path which is worse or equally good as the path extending to k.

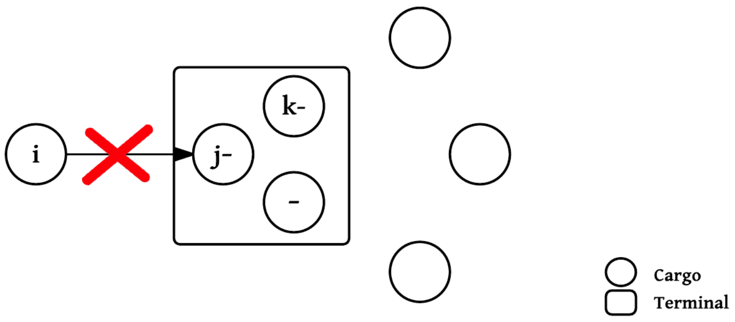


Figure 5.9: If all cargoes in j's associated terminal are delivery loads, the weight of j is less than the weight of k, and there are no time windows or loading times interfering, it is always better to visit k first, and the extension to j should not be done

Rule 12: Loading rule for pickup cargoes

This is a similar test as Rule 11, only for pickup nodes. Let i be the current node and let j be the possible new extension considered. For this rule to apply, all nodes in j's associated terminal must be pickup nodes. If the time windows or loading times are not interfering, it is desirable, or does not affect the problem later, to serve the smallest pickup cargo first. I.e. if there exists a node in the same terminal as j, call it k, where the arc (i,k) exists, both of k's lower and upper time window limits are lower than those of j, the loading time at k is lower than the time at j and the weight of cargo k is less than the weight of cargo j, it is better or equally good to extend to k before j. Hence, the extension to j should not be done.

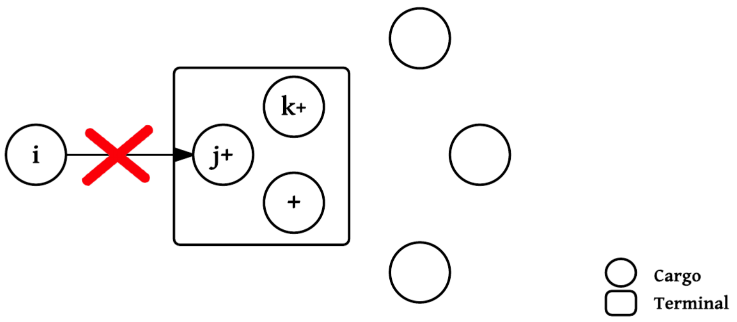


Figure 5.10: If all cargoes in j's associated terminal are pickup loads, the weight of k is less than the weight of j, and there are no time windows or loading times interfering, it is always better to visit k first, and the extension to j should not be done

5.2.3 Dominance step

As described in Section 5.2.1, the extended labels in $\mathcal{E}$ are compared to labels in $\mathcal{U}$ with the aim of dominate labels. If the current node in both labels are in the same terminal, and exactly the same nodes are visited (the same l_{ID}), not in the same order, the dominance rules apply. The set of criteria used for deciding which labels that dominate others is described in the following sections.

The basis for comparison is the time for when all operations are finished in the current node in a label. Let t_a^C be the sum of the arrival time in label a, the loading time for the current node in label a, and the potential washing time for the current node in label a.

$$t_a^C = t_a + L_a + W_a$$

Let this sum be called the *current time* for label a. The comparison between current times ensures that exactly the same operations in total are done, and that all operations at a node are finished.

If two labels have different current times

Let $\mathcal{L}_a$ be one label from the set of extended labels, $\mathcal{E}$, and let $\mathcal{L}_b$ be one label from the set of unprocessed labels, $\mathcal{U}$. If $t_a^C < t_b^C$, label $\mathcal{L}_a$ dominates label $\mathcal{L}_b$ and $\mathcal{L}_b$ is removed from $\mathcal{U}$. If $t_a^C > t_b^C$, label $\mathcal{L}_b$ dominates label $\mathcal{L}_a$ and $\mathcal{L}_a$ is removed from $\mathcal{E}$.

If two labels have equal current times

Let $\mathcal{L}_a$ be one label from the set of extended labels, $\mathcal{E}$, and let $\mathcal{L}_b$ be one label from the set of unprocessed labels, $\mathcal{U}$. If $t_a^C = t_b^C$ and exactly the same nodes have been visited, these paths are symmetrical for this problem. Therefore, when $t_a^C = t_b^C$, $\mathcal{L}_b$ dominates $\mathcal{L}_a$ in order to avoid two labels with symmetrical paths. This is randomly chosen with the intention to avoid two symmetrical labels with the same time used.

Chapter 6

Solution methods for the problem with tank allocation

This chapter presents the solution methods proposed for solving the pickup and delivery problem with time windows, draft limits and tank allocation (PDPTW-DLTA), described in Section 4.3. The first solution method suggested for solving the PDPTW-DLTA is to use a MIP-solver, mentioned in Section 6.1. Another, more extensive method developed is a heuristic. The heuristic consists of a dynamic programming (DP) algorithm to solve the routing problem and a MIP-solver finding a feasible tank allocation. Using dynamic programming on routing problems offers the possibility of storing several solutions. It was considered convenient to use a DP-algorithm finding routes before solving the tank allocation problem (TAP) for fixed routes. The method is fully described in Section 6.2.

6.1 Implementation in a MIP-solver

The problem in Section 4.3 can be implemented directly in a MIP-solver, defining its sets, parameters, objective function and the different constraints. This is the same approach as the one briefly described in Section 5.1.

6.2 Heuristic

Figure 6.1 provides an overview of how the PDPTW-DLTA is solved by the heuristic. First, a DP-algorithm similar to the one presented in the previous chapter is used for solving the PDPTW-DL, generating feasible routes for the ship trying to minimize the total time used in port. This is described in Section 6.2.1. Then, these paths are used as input to a tank allocation problem (TAP) trying to find a feasible tank allocation, presented in Section 6.2.2. The TAP is solved using a MIP-solver. It is solved for each path until a feasible solution is found, and the first path to be feasible is the best solution. If a feasible tank allocation is found, the tank allocation in every node is provided. The final solution of the PDPTW-DLTA comprise of the solution found by solving the PDPTW-DL and the solution found by solving the TAP.

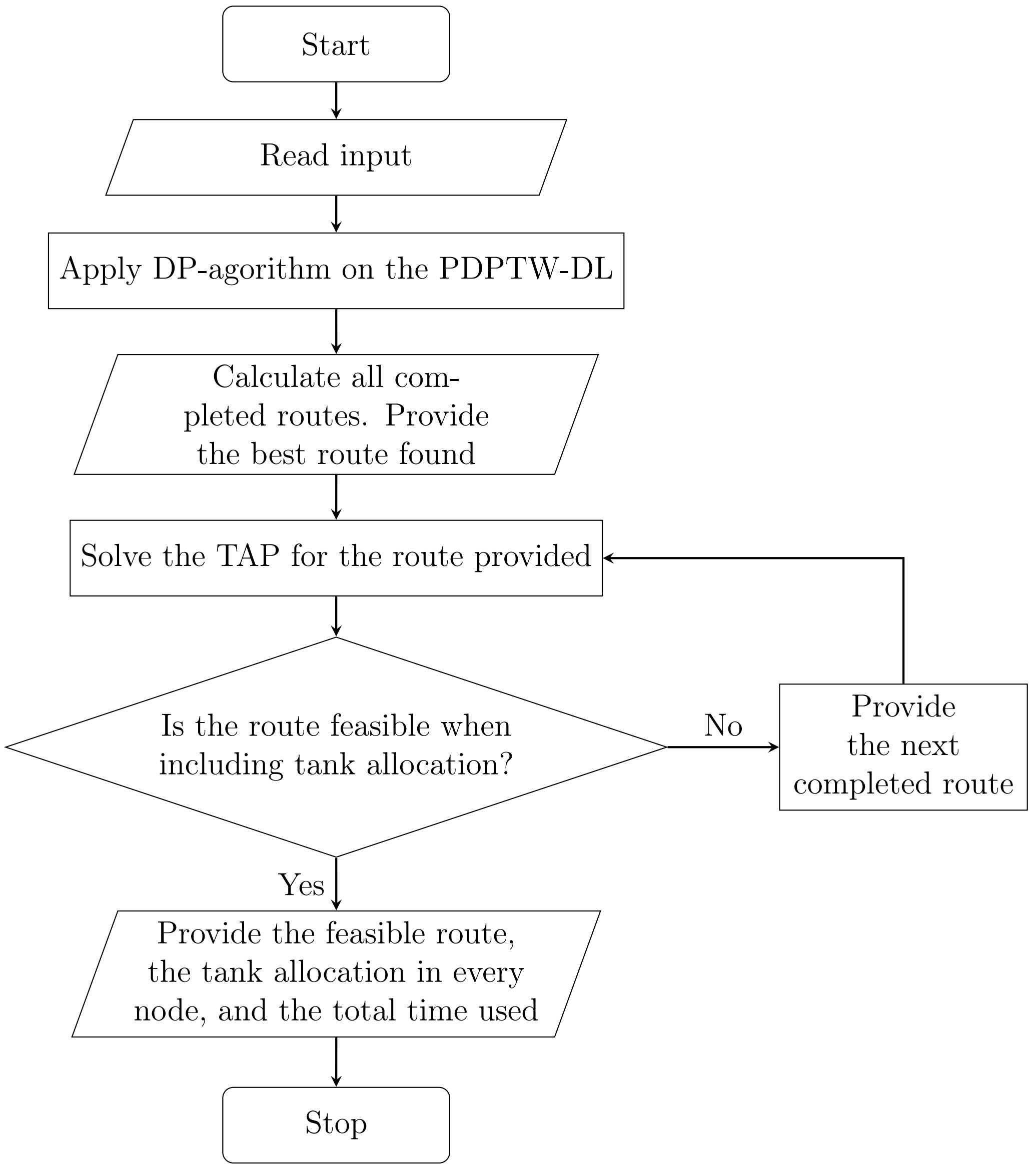

Start
Read input
Apply DP-agorithm on the PDPTW-DL
Calculate all completed routes. Provide the best route found
Solve the TAP for the route provided
Is the route feasible when including tank allocation?
No
Provide the next completed route
Yes
Provide the feasible route, the tank allocation in every node, and the total time used
Stop

Figure 6.1: A flowchart for how the PDPTW-DLTA is solved

The complete solution algorithm for the PDPTW-DLTA is a heuristic, even though the DP-algorithm and the MIP-solver separately can be seen as exact methods. The reason for calling the solution algorithm a heuristic is because the DP-algorithm can exclude optimal solutions for the TAP. The rules presented in Section 5.2.2 may result in exclusion of routes that make the TAP feasible. If two labels have paths which have visited exactly the same nodes, and the current terminal is the same, one path is dominated if the time used is longer than the time used on the other path. When solving the TAP for a particular instance, a dominated route may be the route with the best completion time securing a feasible tank allocation. This route could have been dominated by another route, that later proves to be infeasible for the TAP, because it was using less time at the point when the two routes were compared.

The result is that the completion time is longer than necessary or that the heuristic finds the problem to be infeasible on this instance.

However, if the best path provided by the DP-algorithm is feasible for the tank allocation problem, we have the optimal solution. Because of this, we will, after solving the PDPTW-DL and before solving the TAP, always have an optimistic bound for the overall problem. Hence, the time used on the best ship route provided by the DP-algorithm is a lower bound for the overall problem, and if it is feasible for the TAP, it is the optimal solution of the PDPTW-DLTA.

6.2.1 Dynamic programming formulation

The DP-algorithm for finding feasible routes to the PDPTW-DLTA is similar to the one presented in Section 5.2.1. Some small changes are made to account for the tank allocation. Now, *all* labels with paths where all nodes are visited, are stored in the set of *complete labels*, $\mathcal{C}$. These are all paths that are not dominated during the algorithm. This differs from the algorithm in Section 5.2.1, where only one complete label was stored in $\mathcal{C}$. Except from the set $\mathcal{C}$ and how it is updated throughout the algorithm, the label extension step is identical to the one described in the previous chapter, including the rules in Section 5.2.2. The algorithm is described in Algorithm 6.1.

In addition to reducing the state space and accelerating the algorithm and the overall labeling procedure, the dominance rules are now considering the tank allocation problem and suitable routes with respect to that. The routes generated by the DP-algorithm are used as input to the TAP, and it is therefore desirable to keep the labels most promising for the tank allocation problem and remove the ones that are equally good or worse. For the problem without tank allocation, if two compared labels use the same amount of time, which label that dominates another is a random choice just to avoid symmetrical solutions for that problem. This was described in Section 5.2.3. Now, if two compared labels have paths using the same amount of time, but one in some way is superior regarding tank allocation, it dominates the other one. This is further described in the end of this section. The other domination rules in the algorithm are identical to the ones described in Section 5.2.3.

The sorting step in Algorithm 6.1 results in a set $\mathcal{C}$ with all feasible and non-dominated paths sorted after the completion time of the specific route. The completed routes generated are used as input in the TAP, and it is therefore appropriate to sort them after completion time.

Algorithm 6.1 Generic Dynamic Programming Algorithm

```
(* Initialize *)
Calculate $\mathcal{L}_0$
**set** $\mathcal{U} = \{\mathcal{L}_0\}$
**while** $\mathcal{U} \neq \emptyset$ **do**
    (* Label extension step *)
    **choose** a label $\mathcal{L}_a \in \mathcal{U}$ and **remove** $\mathcal{L}_a$ from $\mathcal{U}$
    **set** $\mathcal{E}_a = \emptyset$
    **if** $N^V(\mathcal{L}_a) \neq N$, i.e. all customers are not visited, **then**
        **for all** arcs $(i(\mathcal{L}_a), j) \in A$ where the extension is complying with rules in $G$ **do**
            extend to node $j$ and **then add** the new label $\mathcal{L}_{new}$ to $\mathcal{E}_a$
    **else**
        extend to end node, $n+1$, and **then add** the new label $\mathcal{L}_{new}$ to $\mathcal{C}$
    (* Dominance step *)
    **for all** labels $\mathcal{L}_{new} \in \mathcal{E}_a$ **do**
        **apply** dominance algorithm to $\mathcal{L}_{new}$ and all labels from $\mathcal{U}$ ending in the same
        terminal $b$ and with the same $l_{ID}$
        **if** label from $\mathcal{U}$ dominates $\mathcal{L}_{new}$ **then**
            **remove** $\mathcal{L}_{new}$ from $\mathcal{E}_a$
        **else if** $\mathcal{L}_{new}$ dominates label from $\mathcal{U}$ **then**
            **remove** label from $\mathcal{U}$
    **add** $\mathcal{E}_a$ to $\mathcal{U}$
(* Sorting step *)
**sort** $\mathcal{C}$ in ascending order with respect to time used for completing the route
(* Solution step *)
**return** $\mathcal{C}$, i.e. all feasible, non-dominated, and complete labels
```

If two labels have equal current times

During the label extension process, all possible extensions are found and numerous symmetrical solutions may appear. Let $\mathcal{L}_a$ be one label from the set of extended labels, $\mathcal{E}$, and let $\mathcal{L}_b$ be one label from the set of unprocessed labels, $\mathcal{U}$. If the current time $t_a^C = t_b^C$ and exactly the same nodes have been visited, some solutions may be dominated in order to reduce symmetry. $\mathcal{L}_a$ dominates $\mathcal{L}_b$ if $\mathcal{L}_a$ has a more suitable for when tank allocation is included, i.e. it may be easier to find a feasible solution.

The main reason that symmetry occurs is different ways to visit nodes within the same terminal. If we compare two labels standing in the same terminal, having visited exactly the same nodes and the only difference is the visiting sequence within a terminal, the two paths are symmetrical. This is exemplified in Figure 6.2.

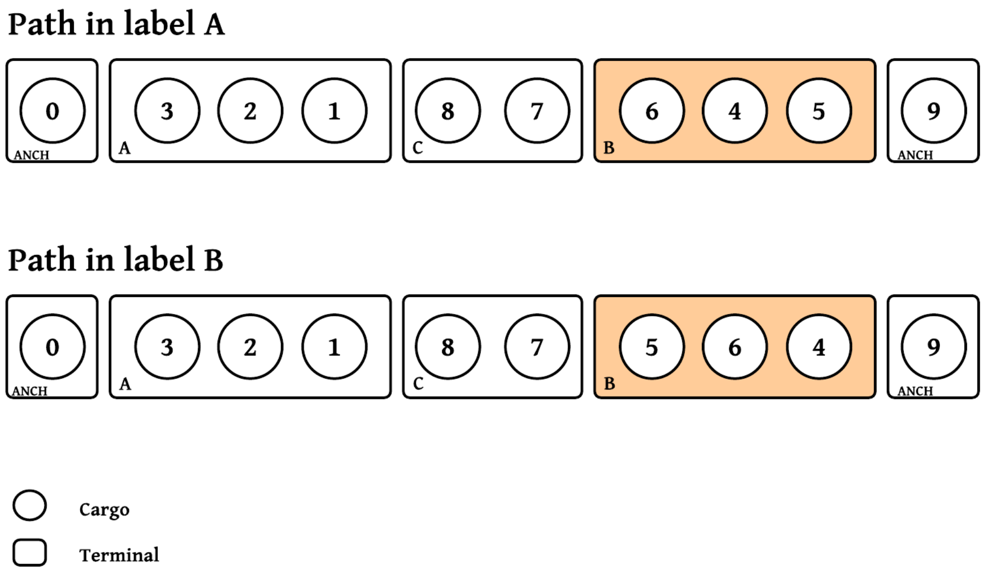


Figure 6.2: $\mathcal{L}_a$ and $\mathcal{L}_b$ have symmetrical paths because the only difference between the paths is the visiting sequence within Terminal B. The terminals are visited in the same order in both paths

Two labels that have visited exactly the same nodes at the same time are not considered symmetrical unless the terminals already visited are visited in the same order, as in Figure 6.2. For symmetrical solutions, it is a precondition that all positions in the paths in the two labels contain nodes located in the same terminal. If the two paths compared have visited the terminals in a different order, we would like to store both paths because they can influence the feasibility when tank allocation is considered. Figure 6.3 shows an example of asymmetrical paths.

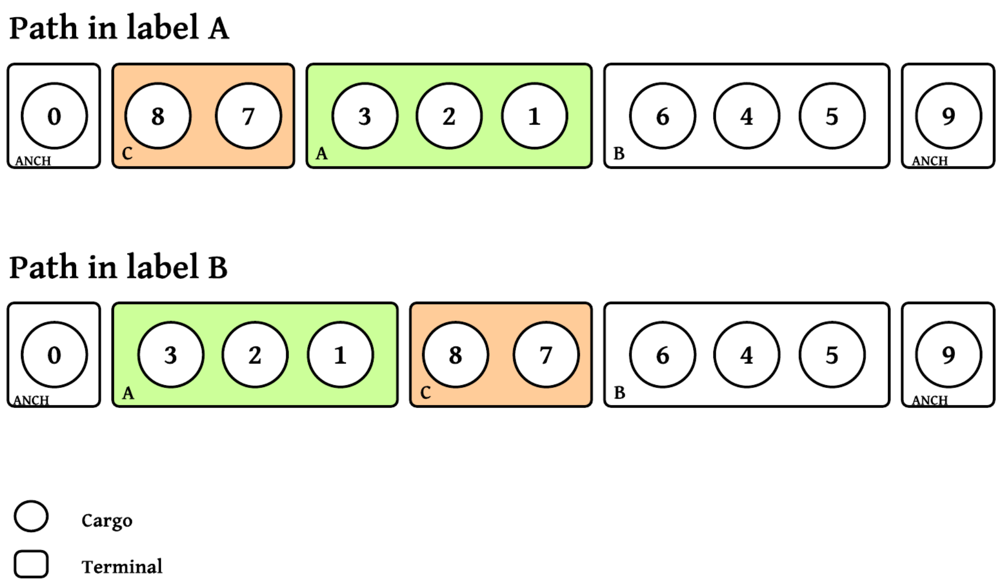


Figure 6.3: $\mathcal{L}_a$ and $\mathcal{L}_b$ have asymmetric paths because the terminals are visited in different orders. The order in which the terminals are visited can influence the feasibility when tank allocation is considered

The labels we would like to eliminate are the ones least suitable for the tank allocation. It is more beneficial to visit delivery nodes before pickup nodes because it more easily can ensure a feasible tank allocation. This is exemplified in Figure 6.4, where $\mathcal{L}_a$ dominates $\mathcal{L}_b$. This is because the delivery nodes in $\mathcal{L}_a$ are visited before the pickup nodes, in contrast to $\mathcal{L}_b$, where a pickup node is visited before the delivery nodes. To differentiate symmetrical sequences of only delivery loads, we favor to serve the biggest cargo first. For pickup nodes, we favor to serve the smallest cargo first. Because the weight of a delivery cargo is defined as a negative number and the wight of a pickup cargo is defined as positive, the desirable sequence of two nodes is achieved by always serving the cargo with the lowest weight first.

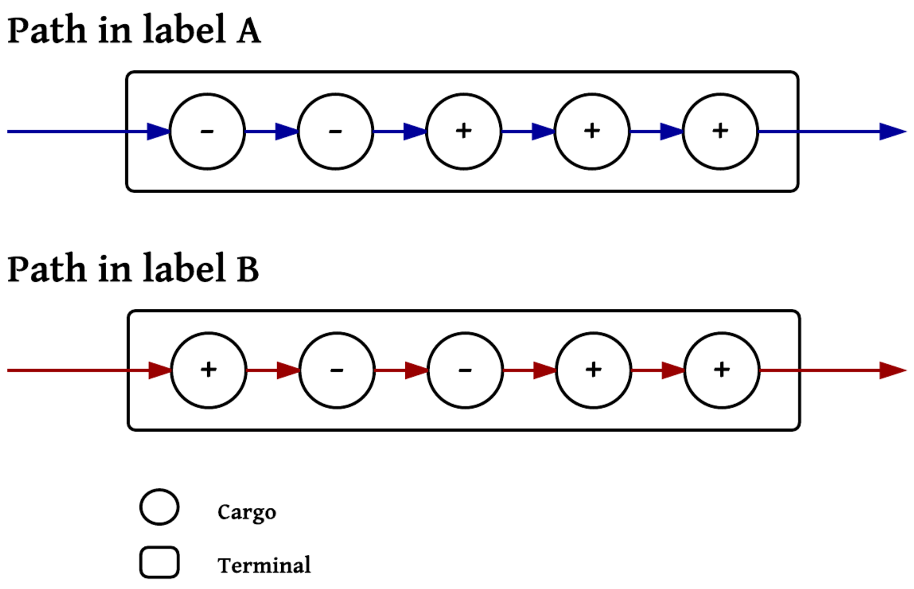


Figure 6.4: All nodes are located in the same terminal. $\mathcal{L}_a$ dominates $\mathcal{L}_b$ because the delivery nodes are visited earlier in the sequence

6.2.2 Solving the TAP for a given ship route

The dynamic programming algorithm produces a list of completed routes, $\mathcal{C}$, sorted by completion time. The fixed paths are used as input to the TAP, and the problem is solved for the paths in the list until a feasible solution is found. The first path to be feasible is the best solution. As explained earlier, this is not necessarily the optimal solution because the DP-algorithm might exclude the optimal path for the problem when tank allocation is included. This implies that the solution found when not including tank allocation, i.e. the solution to the PDPTW-DL, is a lower bound for the objective value of the PDPTW-DLTA. If a feasible solution to the PDPTW-DLTA is found, we have information about the gap. It can be calculated from the objective value of the solution found and from the lower bound.

Because the paths are fixed, the model in Section 4.3 must be modified. Some parameters in the PDPTW-DL, e.g. draft limits and time windows, are not used because the routes already are determined. The time variables t_i, the arc flow variable x_{ij}, and the flow variable y_{ij} are fixed when the path is given. The same applies for the sequence variable z_{ik}. Let the fixed z_{ik}-variable, called Z_{ik}, be an input parameter describing the path. Let Z_{ik} be equal to 1 if node i is visited as number k in the sequence of nodes visited, and 0 otherwise. Because of the fixed routes, a number of constraints can be eliminated. This applies for all the constraints included in the PDPTW-DL in Section 4.2, as well as the sequencing constraints, Constraints (4.21) - (4.26), and the constraints connecting the x_{ij}- and y_{ij}-variables, Constraints (4.39) - (4.41), in Section 4.3.

The remaining variables and parameters are included in the same way as in the model described in Section 4.3. With fixed routes, this problem is similar to the one presented by Hvattum et al. (2009).

For convenience, we repeat the notation used in Section 4.3. Let N be the set of all nodes, indexed by i and j, and let N^C be the set of all cargoes. Let A be the set of all arcs in the network. Let $N^+ \subset N^C$ be the set of pickup nodes, and let $N^- \subset N^C$ be the set of delivery nodes. The sequence number describing the time at which a node is visited is given by $k \in N$. The set of tanks on board the ship is given by F, indexed f. Let N_i^N be the set of loads in conflict with load i. Let N_f^F be the set of loads compatible with tank f, and let F_i^N be the set of tanks compatible with load i. Let F_{ijf} be the set of tanks that cannot be used for load j if load i is present in tank f. Let $\overline{K_f^F}$ and $\underline{K_f^F}$ be the maximum and minimum volume capacity of tank f, respectively, and let V_i be the volume of load i. The parameter W_{if} is equal to 1 if tank f initially contains a cargo $i \in N^C$ when leaving anchorage, and equal to 0 otherwise. The parameter V_{if}^I gives the initial volume of cargo i at tank f when leaving anchorage. The allocation variable w_{ifk}, is equal to 1 if cargo i is allocated to tank f in sequence number k, and 0 otherwise. The load variable l_{ifk} denotes the volume of cargo i allocated to tank f in sequence number k.

The full formulation can also be found in Appendix A.3.

Model formulation

Given the variables and parameters above, the TAP for a given ship route can be modeled as follows.

Allocating and loading constraints

$$w_{if0} = W_{if}, \qquad i \in N^-, f \in F, \tag{6.1}$$

$$l_{if0} = V^I_{if}, \qquad i \in N^-, f \in F, \tag{6.2}$$

Constraints (4.27) and (6.2) ensure that when the ship leaves anchorage, the tanks contain all loads that are going to be delivered in the port.

$$w_{ifk} - w_{if,k-1} + Z_{ik} \geq 0, \qquad i \in N^-, f \in F, k \in N \setminus \{0\}, \tag{6.3}$$

$$w_{ifk} + \sum_{q=0}^{k} Z_{iq} \leq 1, \qquad i \in N^-, f \in F, k \in N \setminus \{0\}, \tag{6.4}$$

$$l_{ifk} - l_{if,k-1} + V^I_{if}(1 - w_{ifk}) \geq 0, \qquad i \in N^-, f \in F, k \in N, \tag{6.5}$$

$$l_{ifk} - V^I_{if} w_{ifk} \leq 0, \qquad i \in N^-, f \in F, k \in N, \tag{6.6}$$

Constraints (6.3)-(6.6) ensure that if a delivery node is visited, the cargo is unloaded and the tank has space for a new one.

$$w_{ifk} - w_{if,k-1} \geq 0, \qquad i \in N^+, f \in F, k \in N \setminus \{0\}, \tag{6.7}$$

$$l_{ifk} - l_{if,k-1} \geq 0, \qquad i \in N^+, f \in F, k \in N \setminus \{0\}, \tag{6.8}$$

Constraints (6.7) and (6.8) secure that if a load is picked up, the same load is placed on the tank the rest of the route.

$$\sum_{q=0}^{k-1} w_{ifq} + Z_{ik} \leq 1, \qquad i \in N^+, f \in F^N_i, k \in N \setminus \{0\}, \tag{6.9}$$

Constraints (6.9) force w_{ifq} to be zero in every sequence until node i is visited and the cargo is picked up.

$$\sum_{f \in F_i^N} \overline{K_f^F} w_{ifk} - V_i Z_{ik} \geq 0, \qquad i \in N^+, k \in N, \tag{6.10}$$

$$\sum_{f \in F_i^N} l_{ifk} - V_i Z_{ik} = 0, \qquad i \in N^+, k \in N, \tag{6.11}$$

Constraints (6.10) and (6.11) secure that if a cargo is picked up, all of the cargo volume is allocated to one or more compartment(s).

$$\underline{K_f^F} w_{ifk} \leq l_{ifk} \leq \overline{K_f^F} w_{ifk}, \qquad i \in N^+, f \in F, k \in N, \tag{6.12}$$

Constraints (6.12) restrict the quantity (in volume) of load i on a tank f to be between the minimum and maximum capacity requirements on that particular tank.

$$y_{0j} \leq Q^- x_{0j}, \qquad (0, j) \in A \mid j \in N^C, \tag{6.13}$$

$$y_{i,n+1} \leq Q^+ x_{i,n+1}, \qquad (i, n+1) \in A \mid i \in N^C, \tag{6.14}$$

$$y_{i,j} \leq (Q^+ + Q^-) x_{ij}, \qquad (i, j) \in A \mid i, j \in N^C, \tag{6.15}$$

Constraints (6.13) and (6.14) connect the x- and y-variables out from and in to the anchorage, forcing the load on the ship to never exceeds the total delivery or pickup load, respectively. Constraints (6.15) connect x- and y-variables between all cargoes.

$$\sum_{f \in F} l_{if,n+1} = V_i, \qquad i \in N^+, \tag{6.16}$$

$$\sum_{f \in F} w_{ifn} = \sum_{f \in F} w_{if,n+1}, \qquad i \in N^+, \tag{6.17}$$

Constraints (6.16) and (6.17) ensure that the allocation at the end node, when the ship reaches anchorage in the last sequence number, is the same as the allocation in the second last sequence number.

Compatibility constraints

$$\sum_{i \in N_f^F} w_{ifk} \leq 1, \qquad f \in F, k \in N, \tag{6.18}$$

In a given sequence, maximum one cargo can be allocated to a specific tank. This is stated in constraints (6.18).

$$\sum_{j \in N_i^N} \sum_{e \in F_{ijf}} w_{jek} - |F|(1 - w_{ifk}) \leq 0, \qquad i \in N^C, f \in F_i^N, k \in N, \tag{6.19}$$

Constraints (6.19) enforce that if load i is on tank f, the load j is not stored in prohibited tanks, given by F_{ijf}. This ensures that no pair of neighboring tanks contains incompatible loads.

Integer and convexity constraints

$$w_{ifk} \in \{0, 1\}, \qquad i \in N, f \in F, k \in N, \tag{6.20}$$

$$l_{ifk} \geq 0, \qquad i \in N^+, f \in F, k \in N. \tag{6.21}$$

Constraints (6.20) impose binary requirements on the allocating variables. Constraints (6.21) impose non-negativity requirements on the loading variables.

Objective functions

There are different possible objective functions for this problem, also discussed by Hvattum et al. (2009). The problem considers a ship's activity for a specific port, ignoring the ship's voyage before and after this port. If two tank allocation plans visiting exactly the same nodes in the same sequence are compared, there are no differences in costs or time usage. In this problem, tank allocation planning may only be interesting in terms of feasibility. An appropriate objective function treating all feasible solutions equally is:

$$minimize \quad 0 \tag{6.22}$$

If considering future decisions, other objective functions are appropriate. The following objective function is minimizing the capacity used in the last sequence. It sums up the tank capacities of occupied tanks in the last sequence, with the aim of having as much capacity available as possible when leaving the port.

$$minimize \quad \sum_{f \in F_i^N} \sum_{i \in N^+} \overline{K_f^F} w_{if,n+1} \tag{6.23}$$

Chapter 7

Computational Study

This chapter presents the results from a computational study of the models without (PDPTW-DL) and with tank allocation (PDPTW-DLTA), described in Chapter 4. The results are examined to discuss and compare the two formulations and the solution methods used for solving them. The aim is to analyze how the models and the solution methods perform, to understand how the optimal solutions are influenced by including tank allocation, and how the different solution methods and instance sizes influence the run times.

The computational study is based on implementation of the PDPTW-DL and PDPTW-DLTA formulations in different programming languages. When a commercial MIP-solver is used, Xpress Mosel has been the modeling language. Xpress-IVE Version 1.24.04 64 bit with Xpress Optimizer Version 26.01.04 has solved the implementation. For the dynamic programming algorithm, Java 1.0.7_79 has been the modeling language and Eclipse Version 4.4.1 has been the integrated development environment. The implementations are done on a computer running Windows 7 enterprise 64-bit operating system with an Intel CoreTM i7-3770 CPU @ 3.4 GHz and 16 GB of installed memory.

To show how the models work and how the sizes of the instances affect the run time, different test instances were made and tested. The input data needed to fully describe a test instance is presented in Section 7.1. Section 7.2 gives an overview of the test instances generated. Section 7.3 presents a technical testing performed to analyze which rules that should be included in the DP-algorithm. This is followed by testing and analyses of the results from solving the models with and without tank allocation, in Sections 7.4 and 7.5, respectively. Here, the results obtained when using both solution methods are discussed and compared. Then, the two models are compared in Section 7.6.

7.1 Input data

Based on information from the case company in this thesis, which is a Norwegian shipping company, instances used for the computational study are constructed. Some of the parameters are exactly as in reality, while others are estimated to be as realistic as possible. We aim to imitate the factual situation as closely as possible in order to evaluate the results in a way that gives valuable input to decision makers in shipping companies facing similar problems.

The information about the instances is divided into four different categories; geographical information, cargo information, terminal information, and information about the ship. The cargo information and the terminal information provide data about all cargoes and terminals included in the relevant instance, while the ship information includes data of all tanks on the ship. Each instance is structured in the same way in an Excel workbook, with four sheets corresponding to each of the information categories, such that the data is readable data for the algorithm developed in Java. Table 7.1 shows a listing of all the parameters needed to fully describe an instance. It can be seen that when tank allocation is included, some extra parameters are needed to fully describe an instance.

Information category	**Parameters PDPTW-DL**	**Extra parameters PDPTW-DLTA**
Geography	Distance Matrix	
Cargo	Node number, Associated terminal, Pickup or delivery, Loading time, Required washing time, Cargo load [tonnes], Lower time window limit, Upper time window limit	Density, Cargo type
Terminal	Terminal number, Draft limit [tonnes], Waiting time, Entering time	
Ship	Total ship capacity[tonnes]	Tank number, Tank min capacity, Tank max capacity, Tank type

Table 7.1: All parameters needed to describe an instance

There are additional parameters used in the model(s), all described in Chapter 4, not listed here, but they are derived given these input parameters. The next paragraphs explain how these parameters are chosen for the computational study.

7.1.1 Geography

The input data are calculated for a chemical tanker, using Galveston Bay, Houston, Texas, as the case port. In this port, the case company can visit up to about 17 terminals, and may serve several customers at each terminal. The computational study is considering instances where customers are distributed at 3-11 terminals, depending on how many cargoes the ship is going to load/unload. Figure 7.1 illustrates the chosen terminals in the port, described by A, B, C, D, E, F, G, H, I, J, and K, where the anchorage is located at sea outside the port.

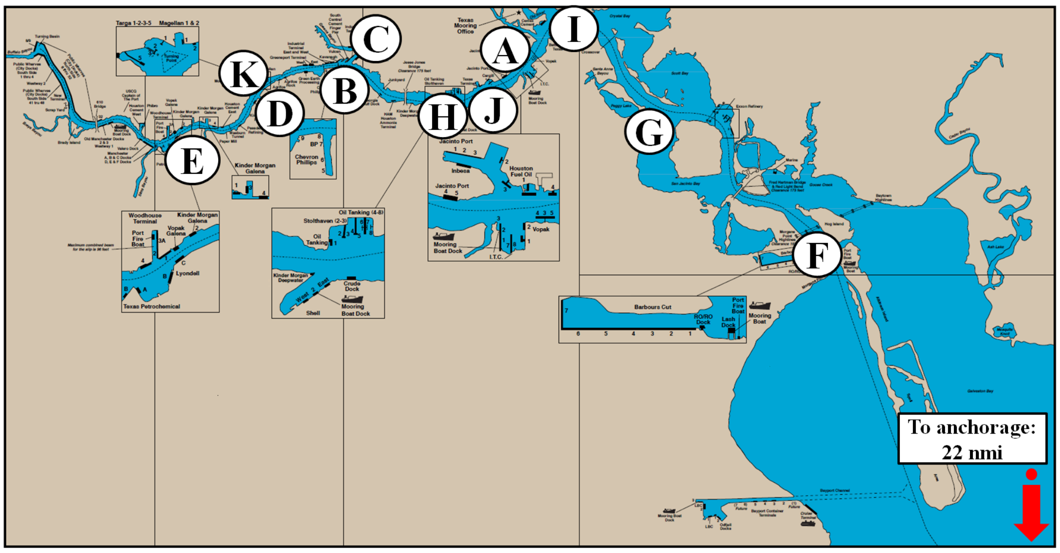


Figure 7.1: 11 points in Galveston Bay are chosen to model the terminals where the cargoes are going to be picked up or delivered (Texas Mooring Inc., 2015)

To be able to calculate the total time used in port, the relevant distances in port and the sailing speed were found. The distances between the anchorage and the different terminals were found using Google Maps Distance Calculator, and presented as a distance matrix. Based on a time table consisting of approximated sailing times between different real points in Galveston Bay and the distance matrix, a sailing speed corresponding to 4.3 knots was calculated. This sailing speed was used on every instance to calculate the sailing times. Each instance contains a distance matrix with the distances between the terminals included in that instance.

7.1.2 Cargo information

The cargoes a ship has to deliver and pick up in port are a result of strategic and tactical decisions made by the shipping company, and are partly predetermined through long-term agreements. The number, types and sizes of cargoes may vary, and the computational study is therefore done with instances of different cargo specifications. Each instance contains information about every cargo included in that instance, and the cargoes are described by a unique node number from 1 to the total number of cargoes, n. The anchorage has node number 0 or $n+1$, for the first and last time it is visited, respectively. Each cargo has information about which terminal it is located in, and whether it is a pickup or a delivery cargo, which has been randomly selected. The rest of the cargo information is more thoroughly described below.

Cargo size

The sizes of the cargoes used in the computational study reflect realistic cargo sizes, and are in some cases such that the ship capacities and draft limits may be restrictive. The cargoes are such that the total weight of the delivery cargo and the total weight of the pickup cargo, i.e. the load on board the ship when entering and leaving the port, respectively, are randomly chosen between 60 and 90% of the total ship capacity. Given a sum of the total weights of the delivery and pickup cargoes in an instance, the total weights are randomly distributed on the cargoes in that instance. This implies that if the number of cargoes is increased the cargoes become smaller, given that we look at the same ship type.

The size of the cargoes are given in tonnes. The cargoes sizes are ranging from 115 to 7,225 tonnes. For the PDPTW-DLTA formulation, the cargoes are to be placed to tanks with capacities given in volume. Thus, the cargoes must be expressed in terms of volume. This is done by ascribing a product type to each cargo, and using its density and weight in tonnes to express the cargo size in terms of volume.

Cargo type

The product types used came from information about which types of products a chemical tanker can carry, some of them shown in Figure 1.2. Among these are chemical liquids, vegetable oils, and brake fluid.

The PDPTW-DLTA formulation can account for incompatibilities between cargoes and tanks, and between different cargoes, by categorizing the tanks and cargoes into types. The models are developed such that the information about incompatibility can, if necessary, be ignored.

Incompatibilities between different cargoes typically apply for cargoes of hazardous materials that can react chemically if they are stored in adjacent tanks. Because of little information about these materials and an increased need for input data, the computational study performed in this thesis is only considering incompatibilities between tanks and cargoes, and not between different cargoes. Incompatibility between different cargoes is considered by Hvattum et al. (2009).

There are two types of tanks considered in this computational study, further discussed in Section 7.1.4. The cargoes are divided into three cargo types to describe the compatibility to the two different tanks; Cargo type 1 indicates compatibility with tank type 1, cargo type 2 indicates compatibility with tank type 2, while cargo type 3 indicates compatibility with both.

Loading time and washing time

The time it takes to load/unload a cargo depends on product type and size of the product. The loading times are approximated to be between 0.5 and 14 hours. The cargoes that are to be delivered are assigned varying washing times, between 3 and 15 hours, to illustrate different washing and disposal requirements for different types of products. Both the loading times and the washing times are constructed numbers based on information from the case company, trying to reflect the factual situation.

Time windows

The time windows for when the cargoes can be served are approximated based on assumptions of which time windows the charterers can operate with. The widths of the time windows are set relatively large to avoid predetermination of the routes sailed, ranging from a width of 2 to 24 days. When the number of cargoes increases, the time windows are wider. An assumption made is to use the same time windows for some cargoes in the same terminal, giving the ship an opportunity to do several operations in a terminal directly after each other as long as the capacity of the ship and the draft limits allow it. This is done to illustrate that some cargoes might belong to the same customer, requiring equal time windows on different cargoes.

7.1.3 Terminal information

A test instance consists of some or all of the terminals illustrated in Figure 7.1. Each terminal is assigned a unique terminal number from 1 to the total number of terminals, n. Anchorage is assigned the terminal number 0. The terminal parameters are explained below.

Entering time and waiting time

The time it takes for the ship to enter a terminal depends on the ship and/or the terminal. Based on indications regarding logical entering times provided by the case company, the entering times are chosen to be in the range of 0.5 to 3 hours.

As discussed in Section 4.1, the waiting times for the terminals can vary a lot. Based on information from the case company, the waiting times are chosen integers between 4 and 13 hours, even though they are uncertain and in some cases can be significantly higher.

Draft limits

As mentioned in Chapter 2, the draft limits can be expressed as tonnes of load on board the ship when considering only one ship. The computational study is considering five different ship types (further described in the next section), and the draft limits in the terminals are therefore depending on which ship type that is used for each particular instance.

Ship types 1 and 2 are fictive ships, and here the draft limits are chosen based on the sizes of the cargo loads, lower than the total ship capacity which is ranging from 9,000 to 10,000 tonnes.

For ship type 3, 4 and 5, the following information has been used to decide the draft limits at the terminals:

1. **Draft S. Dwt:** the worst-case loaded draft a ship can have in the summer, measured in meters. Also known as the ship's total capacity
2. **Dwt(summer):** the maximum weight a ship can safely carry, measured in metric tons
3. **TPC S.Draft:** Tonnes per Centimetre Immersion - the mass which must be loaded or unloaded to change the ships draft by 1 cm

Table 7.2 shows the parameters for the different ship types.

Ship type	**Draft S. Dwt** $[m]$	**Dwt(summer)** $[t]$	**TPC S.Draft** $[t/cm]$
3	9.79	19,805	29.30
4	10.72	37,479	50.00
5	11.50	40,048	52.70

Table 7.2: Parameters used for deciding the draft limits

We assume that the worst-case draft in meters corresponds to the case where the ship is fully loaded, i.e. when all weight on board the ship is equal to Dwt(summer). By assuming a linear relationship between the draft of the ship and the weight on board, it is possible to express the draft limits as a function of the weight of the cargoes on board the ship.

The draft limits in the terminals chosen are not exactly the same as the real draft limits. The reason for this is that we want the draft limits to possibly be restrictive for some routes. The draft limits in the different terminals are therefore constructed as randomly chosen numbers between 70 - 100% of each ship type's total capacity. Table 7.3 presents the range of draft limits for each type.

Ship type	**Draft limits** $[t]$
3	16,289 - 18,047
4	29,289 - 33,642
5	26,635 - 36,844

Table 7.3: Draft limits for each ship type

7.1.4 Ship information

The case company used for the computational study is operating chemical tankers with carrying capacities ranging from 4,000 to 50,000 dwt. The instances in this thesis consist of five different ship types. For the problem without tank allocation, the total ship capacity, given in dwt, is the only ship-parameter needed. For the problem with tank allocation, information regarding the ship's tanks is also necessary. The information about ship types, tank capacities and tank types is explained more closely below. The algorithm developed for the initial placement of the delivery cargoes on board the ship when entering the case port is also described.

Ship types

The instances with the fewest number of cargoes consider fictive ships, while the larger instances consider real ships used by the case company. The ship types are presented in Table 7.4.

Ship type	**Ship name**	**# of tanks**	**Tank capacities** $[m^3]$	**Tank types**	**Total capacity** $[dwt]$
1	Fictive ship 1	4	1,500 - 5,000	Stainless steel	11,000
2	Fictive ship 2	7	800 - 2,700	Stainless steel	11,000
3	Bow Fuji - Usuki 19-20	22	307 - 1,274	Stainless steel	19,805
4	Bow Faith - KVAERNER1	52	147 - 3,058	Stainless steel, zinc	37,479
5	Bow sea - POLAND	40	352 - 2,593	Stainless steel	40,048

Table 7.4: Ship types used in the instances

Figures 7.2, 7.3, and 7.4 illustrate the tank configurations of the ship types 3, 4 and 5.

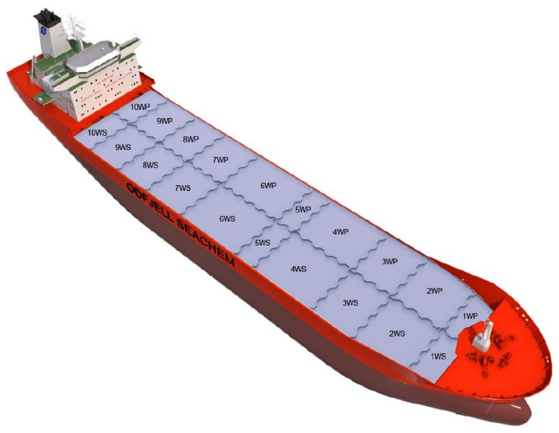


Figure 7.2: Tank configuration of the ship *Bow fuji*

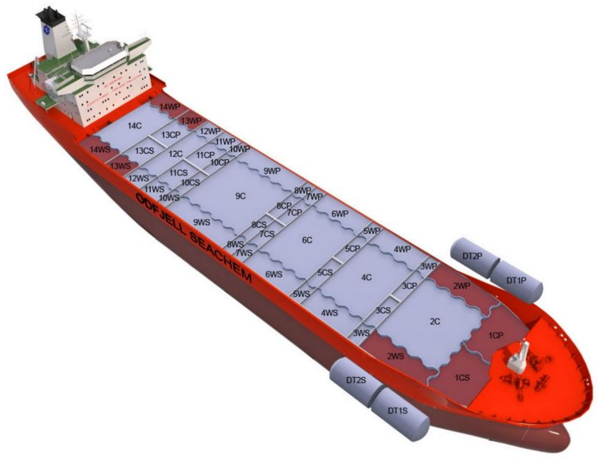


Figure 7.3: Tank configuration of the ship *Bow faith*

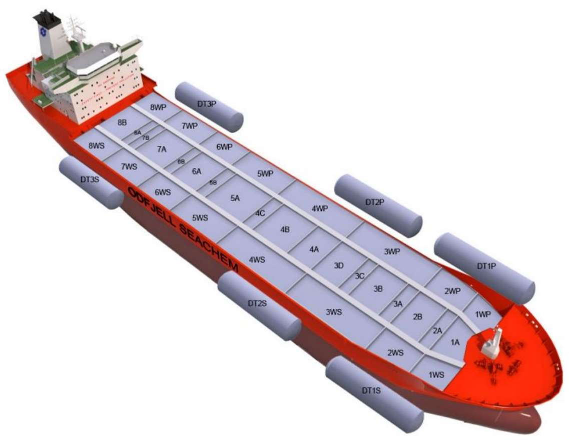


Figure 7.4: Tank configuration of the ship *Bow sea*

Tank capacities

The maximum tank capacities for the fictive ships (type 1 and 2) are chosen such that their sum in volume is roughly equivalent to the total capacity given in dwt, using an average density of 1,000 kg/m^3. The information about the real, existing ships is gathered from specifications given by the case company. As sloshing and spilling in the tanks are assumed not to be a problem, the minimum tank capacities are set to zero.

Tank type

As mentioned, the tanks and cargoes are divided into types to account for incompatibility. The ships considered have two tank types. Type 1 is illustrating stainless steel and type 2 is illustrating zinc. If incompatibility is included in the model, the sets describing the cargoes compatible with a given tank f, N_f^F, and the tanks compatible with a given cargo i, F_i^N, are computed based on the cargo and tank types.

The set of tanks F_{ijf} describes the tanks that cannot be used by cargo i if cargo j is present on tank f, and is important if certain cargoes cannot be placed in adjacent tanks. As previously mentioned, this type of incompatibility is disregarded in this thesis.

Initialization of cargo placements

When the ship is entering the port, it contains all the delivery cargoes that are to be delivered. The initial placement of cargoes are done using a greedy algorithm implemented in Java, described in Algorithm 7.1.

Recall that N^- is the set of delivery cargoes. For the algorithm, some new sets and parameters are included. Let F^{free} be the set of the remaining available tanks. Let F_i^* be the number of tanks needed for placing cargo i, based on which tanks that are not already assigned to a cargo.

Algorithm 7.1 Initial Cargo Placement Algorithm

```
(*Initializing step*)
Let $F^{free}$ be all tanks on board the ship

(* Sorting step *)
sort $F^{free}$ in ascending order with respect to tank capacity
sort $N^-$ in descending order with respect to the cargoes' weight

(* Cargo placing step*)
for all nodes $i \in N^-$ do
    find the minimum number of compatible tanks, $F_i^*$, needed for placing cargo $i$
    if the number of tanks needed is more than available and compatible tanks then
        the problem is infeasible, i.e. there are not enough tanks to carry all delivery
        cargoes
    else
        find available and compatible tanks such that the capacity of the remaining,
        unused tanks is maximized
        remove the tanks used from the set of available tanks, $F^{free}$
return initial placements, i.e. which cargoes that are allocated to which tanks

(* Volume placing step*)
for all nodes $i \in N^-$ do
    distribute the volume of cargo $i$ even on the tanks used for that particular cargo
return volume placements, i.e. how much of each cargo that is placed on which tanks
```

Some remarks to the pseudocode should be made.

1. The number of tanks needed is calculated based on which tanks that are not already allocated to other cargoes. The same applies for the determination of which tanks to use. This means that when the biggest cargo is allocated to tanks, all the tanks compatible with this cargo are available. The number of available tanks is therefore decreased throughout the cargo placing step.
2. The volume of a cargo is distributed evenly on the tanks assigned for that particular cargo. If one or several tanks have maximum capacities less than the capacity required to accomplish this, these tanks are filled up to their maximum capacity. The remaining volume is then divided on the remaining tanks, with the aim of an even distribution of the volume not yet allocated. This continues until the cargo is allocated to tanks.

The aim with the initialization is to occupy as little capacity as possible when leaving anchorage. The algorithm is developed as a greedy heuristic, making the locally optimal choice at each stage. The biggest cargo is allocated to tanks first, finding the best way to allocate that particular cargo. As this is a heuristic approach, there may be another combination of tanks used that in total occupies less capacity than the one found.

7.2 Generating the test instances

Based on the input data described in the last section, test instances were generated, summarized in Table 7.5. The instances vary with the number of cargoes (|N|), the number of terminals (|B|), the number of pickup and delivery cargoes, the size of the cargoes, and the ship types, trying to illustrate that these aspects may vary from visit to visit in the same port.

Additionally, there was made some instances considering cargo and tank incompatibility, to be able to analyze how the results are influenced by adding complexity. These instances are made from the instances excluding the incompatibility information.

The instances are divided into seven groups based on the number of cargoes. In total, 98 instances were made; 40 of them which did not take tank allocation into consideration (Excl. TA) and 58 which did (Incl. TA). 18 out of 58 of the instances with tank allocation were tested also considering incompatibility between cargoes and tanks (Incl. TA and incomp).

|N|	|B|	# of pickups	# of deliveries	Cargo sizes (tonnes)	Ship type	# of instances Excl. TA	Incl. TA	Incl. TA and incomp.
5	3	2-3	2-3	1,180-7,225	1	6	6	2
8	3	2-4	4-6	262-4,898	2	5	5	2
12	4	6-7	5-6	515-5,691	3	7	7	2
15	5	6-11	4-9	626-5,714	3	7	7	2
20	8	9-11	9-11	1,012-5,715	4	5	5	5
25	11	13-16	9-12	1,055-5,579	4	5	5	5
30	11	13-16	14-17	115-4,754	5	5	5	0

Table 7.5: Overview of the test instances used for the computational study

The terminal information and the geography are the same in each data set, just adding more terminals when more cargoes are included.

All instances have been given the following notation: Number of cargoes_number of pickup cargoes_number of delivery cargoes_ instance name. Instance (5_3_2_1) is e.g. the first instance with five cargoes, consisting of three pickups and two deliveries.

7.3 Algorithm setting

When performing the label extension step in the DP-algorithm presented in Section 5.2.1, several rules were included to ensure feasibility and reduce symmetry when extending to a new node. Recall that three of the rules are mandatory rules that must be satisfied in order to extend to a new node. The remaining rules are performed with the aim of reducing the run time of the algorithm.

In this section, a technical analysis is performed in order to decide the impact of the rules included to reduce the run time (i.e. all rules except the mandatory ones). The goal of this analysis is to decide which rules that should be included in the model. This is done by comparing the run times with and without the rules. More rules may require longer run time, but fewer rules result in more extensions done and labels created, which again result in longer run time. An analysis is done based on three cases that include different rules:

1. **All rules included:** All rules are included in the same order as presented in Section 5.2.2.

2. **Selected rules included:** Only the rules that were restricting on the instances tested when all rules were included, i.e. the rules that reduced the number of possible extensions, are included in the algorithm. The rules are implemented in descending order with respect to the number of times they applied, i.e. the rule that applies most is tested first.
3. **No rules included:** None of the rules are included.

In the analysis, 12 instances have been tested. Two instances from each instance group of 5 nodes up to 25 nodes were chosen. Table 7.6 shows how many times each rule applies when all rules are included in the label extension step.

	Rule number								
Instance	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
5_3_2_1	0	8	0	0	0	0	0	0	0
5_2_3_2	0	0	0	0	0	0	2	0	0
8_4_4_1	27	9	4	0	0	0	8	0	0
8_4_4_2	0	0	5	0	0	0	5	0	0
12_7_5_1	216	24	8	0	0	0	16	0	0
12_6_6_2	511	36	7	0	0	0	0	0	0
15_7_8_1	5,722	71	55	0	0	0	81	0	0
15_7_8_2	3,128	0	18	0	0	0	150	1,221	0
20_11_9_1	1,532	0	0	0	0	0	0	0	0
20_9_11_2	3,310	0	0	0	0	0	46	0	0
25_14_11_1	12,864	4	0	0	0	0	40	0	0
25_14_11_2	30,066	0	0	0	0	0	211	2,353	0

Table 7.6: The number of times the rules apply for the testes instances

It can be seen that rule 4 stands out by applying in all instances except from three. This means that the rule prohibits extensions to specific nodes during the execution of the algorithm, and thus reducing the number of labels created. It can also be seen that the number of times the rule applies increases with the number of nodes visited. Rules 5, 6, 10, and 11 apply less frequently and only a few times compared to rule 4. Rule 11 applies only in two instances, but in return this rule prohibits several extensions when these instances are considered. Rules 5, 6 and 10 apply only a few times, but are in comparison to rule 11 frequent in six, six and nine instances, respectively. Based on this, the rules 4, 5, 6, 10, and 11 are selected to be included in the test case with applied rules.

Table 7.7 shows the test results when only the rules 4, 5, 6, 10, and 11 are included and performed in the algorithm. The run times for the two other cases are also included in the table in order to give a comprehensive overview of the test results.

Instance	**Run time *[sec]***			**# of labels generated**		
	All rules	Selected rules	No rules	All rules	Selected rules	No rules
5_3_2_1	1.0	1.1	1.0	34	34	42
5_2_3_2	1.1	0.6	2.0	27	27	29
8_4_4_1	1.0	1.2	1.2	642	642	719
8_4_4_2	1.1	1.1	0.9	998	998	1,008
12_7_5_1	0.9	0.9	1.1	1,236	1,236	2,180
12_6_6_2	1.1	1.0	1.4	1,655	1,655	3,531
15_7_8_1	1.3	1.3	3.2	16,912	16,912	48,357
15_7_8_2	1.9	1.9	4.3	31,091	31,091	62,612
20_11_9_1	1.1	1.1	1.3	2,925	2,925	8,171
20_9_11_2	1.5	1.4	1.6	18,153	18,153	29,662
25_14_11_1	3.1	3.0	10.4	38,742	38,742	84,782
25_14_11_2	8.0	8.0	79.7	77,558	77,558	252,130

Table 7.7: The run times and number of labels generated for all three cases

When only the selected rules are included, the rules are performed in the order 4, 11, 10, 6, and 5. This order is chosen after the number of times they applied and were restricting during the execution of the algorithm. Rule 11 is performed before 10, 6, and 5 because it applies more frequently than the other rules when considering each instance. It can be seen from column two and three that the differences in run times, when all rules and only the applied rules are included, are insignificant. Naturally, the numbers of labels generated are the same when all rules or just the selected ones are included.

If case three, where no rules are included, is compared to case one and two, the run times are increased for the instances with 15 and 25 cargoes. For the instances with 20 cargoes, the increase is present, but less significant. For all instance groups, the number of labels generated is significantly increased when no rules are included. This implies that performing the rules are more efficient with respect to computational time compared to skipping them. Not including the rules leads to more extensions and more labels generated, resulting in longer run times to finish the algorithm.

The prominent increase in run time and number of labels generated when no rules are performed, make case three least suitable for future analyses. Generally, it tends to be insignificant differences in run times between the first two cases where all or some of the rules are included in the algorithm. Including the rules is much cheaper with respect to run time compared to omitting them. Because the run times are almost the same when reducing the number of rules performed, it can be argued for including all of them. The reason for this is that the rules that are not restricting in the above instances can be restricting when the algorithm is performed with other data sets. Even though some rules did not apply in the tested instances, they should be included in the algorithm because the inclusion of more rules does not result in a significantly longer run time. Based on this analysis, the conclusion is that all rules should be included. The order in which they are included is 4, 11, 10, 6, 5, 7, 8, 9, and 12. First, the rules that were restricting in the above instances are in the same order as in the case when only the selected rules were included. Then the remaining rules, i.e. those that did not apply in any of the instances that were tested, are included in the order presented in Section 5.2.2.

7.4 Testing of formulation without tank allocation

This section presents the results from solving the problem without tank allocation (PDPTW-DL) when it was solved with the two solution methods explained in Chapter 5. Table 7.8 in section 7.4.1 summarizes the results obtained by the two solution methods. First, the results are used to compare the performance of the methods. This is followed by a more comprehensive analysis of the results found using the MIP-solver, before a similar analysis of the results found by the DP-algorithm is presented.

7.4.1 Results and comparison of the solution methods

Table 7.8 shows the test results for the PDPTW-DL when the MIP-solver and the DP-algorithm are used, respectively. The objective values and the run times for when the problem is solved for all instances with both methods are expressed in the table. In addition, the percentage gap between the lower bound and the best solution found for when the problem was solved with a MIP-solver is presented. A gap greater than 0.0% means that the solver cannot guarantee to have found the optimal solutions within the set cut-off time. Due to the context of the problem, it is preferable that the problem is solved quite fast, and a maximum run time was set to be 3 hours (10,800 seconds). If the DP-algorithm provides solutions before the set cut-off time, these solutions are always the optimal ones. N/A is an abbreviation for not applicable, and means that the problem was not solved for the particular instance. "N. s. f." is short for no solutions found.

Instance	**MIP-solver**			**DP-algorithm**	
	Objective value *[hrs]*	Gap	Run time *[sec]*	Objective value *[hrs]*	Run time *[sec]*
5_3_2_1	66.3	0.0%	0.1	66.3	1.1
5_2_3_2	87.3	0.0%	0.0	87.3	1.2
5_3_2_3	56.0	0.0%	0.0	56.0	1.0
5_3_2_4	56.0	0.0%	0.1	56.0	1.1
5_2_3_5	64.0	0.0%	0.0	64.0	1.0
5_2_3_6	74.8	0.0%	0.0	74.8	1.0
8_4_4_1	94.8	0.0%	1.0	94.8	1.3
8_4_4_2	77.5	0.0%	4.0	77.5	1.2
8_3_5_3	103.4	0.0%	0.8	103.4	1.4
8_4_4_4	80.1	0.0%	6.3	80.1	1.3
8_2_6_5	100.5	0.0%	2.0	100.5	1.3
12_7_5_1	125.6	0.0%	62.1	125.6	0.9
12_6_6_2	121.4	0.0%	899.3	121.4	1.2
12_7_5_3	164.6	0.0%	647.8	164.6	1.3
12_6_6_4	122.1	0.0%	249.5	122.1	1.5
12_6_6_5	154.6	8.7%	10,800	154.6	1.4
12_6_6_6	131.9	0.8%	10,800	131.9	1.4
12_6_6_7	123.4	0.0%	1,197	123.4	1.4
15_7_8_1	172.0	32.5%	10,800	157.4	1.6
15_7_8_2	183.3	36.9%	10,800	151.9	2.0
15_7_8_3	163.8	29.1%	10,800	149.4	2.0
15_7_8_4	179.6	34.4%	10,800	157.9	1.7
15_11_4_5	145.1	23.1%	10,800	115.4	2.0
15_6_9_6	186.5	42.6%	10,800	161.1	1.9
15_7_8_7	162.8	28.7%	10,800	148.4	1.6
20_11_9_1	N/A	N/A	N/A	284.9	1.5
20_9_11_2	N/A	N/A	N/A	333.3	1.8
20_11_9_3	N/A	N/A	N/A	311.6	1.8
20_9_11_4	N/A	N/A	N/A	320.8	3.3
20_11_9_5	N/A	N/A	N/A	292.9	15.0
25_14_11_1	N/A	N/A	N/A	403.4	3.1
25_14_11_2	N/A	N/A	N/A	406.3	8.0
25_14_11_3	N/A	N/A	N/A	396.4	12.7
25_13_12_4	N/A	N/A	N/A	400.8	160.1
25_16_9_5	N/A	N/A	N/A	399.5	42.2
30_16_14_1	N/A	N/A	N/A	464.7	393.2
30_15_15_2	N/A	N/A	N/A	N. s. f.	10,800
30_16_14_3	N/A	N/A	N/A	464.4	1,415
30_13_17_4	N/A	N/A	N/A	N. s. f.	10,800
30_14_16_5	N/A	N/A	N/A	N. s. f.	10,800

Table 7.8: Test results from solving the PDPTW-DL with a MIP-solver and the DP-algorithm, respectively

From Table 7.8 it is clear that the DP-algorithm shows a significant reduction in run time compared to the MIP-solver. The DP-algorithm shows tolerably run times for all the tested instances except from three of the instances with 30 cargoes, while the MIP-model becomes inoperable when the problem is solved for instances consisting of 12 nodes or more. It can be seen that the MIP-solver has found the optimal solutions (the same solutions as found by the DP-algorithm) for instances 12_6_6_5 and 12_6_6_6, without being able to prove that these are optimal. Gaps are therefore provided for these instances.

The DP-algorithm uses significantly longer time when solving the problem for instance 25_13_12_4 compared to the other instances with the same number of cargoes, probably because of a data set that is not very restrictive. That the run times can vary significantly because of differences in how restricting the data are, is a source of uncertainty and may indicate that the algorithm can perform poorly on some instances. It may be hard to predict how long time the algorithm might use to solve an instance of a certain size. This is a source of uncertainty in other exact methods too. We can see examples of the same when the MIP-solver is used, e.g. for instance 12_6_6_4. It is a known weakness for exact methods that the run time may be hard to predict. The run time of a heuristic may be easier to predict. As the number of iterations that needs to be done is known, it may be possible to set a cut off time to obtain a feasible solution.

Both methods find the same optimal solution, but they may provide different routes. This is because of the presence of symmetrical routes. One example is instance 5_2_3_2. Figure 7.5 shows the optimal routes provided when the problem is solved with a MIP-solver and the DP-algorithm, respectively. Both routes use 94.3 hours, but the sequences in which the customers are served are different.

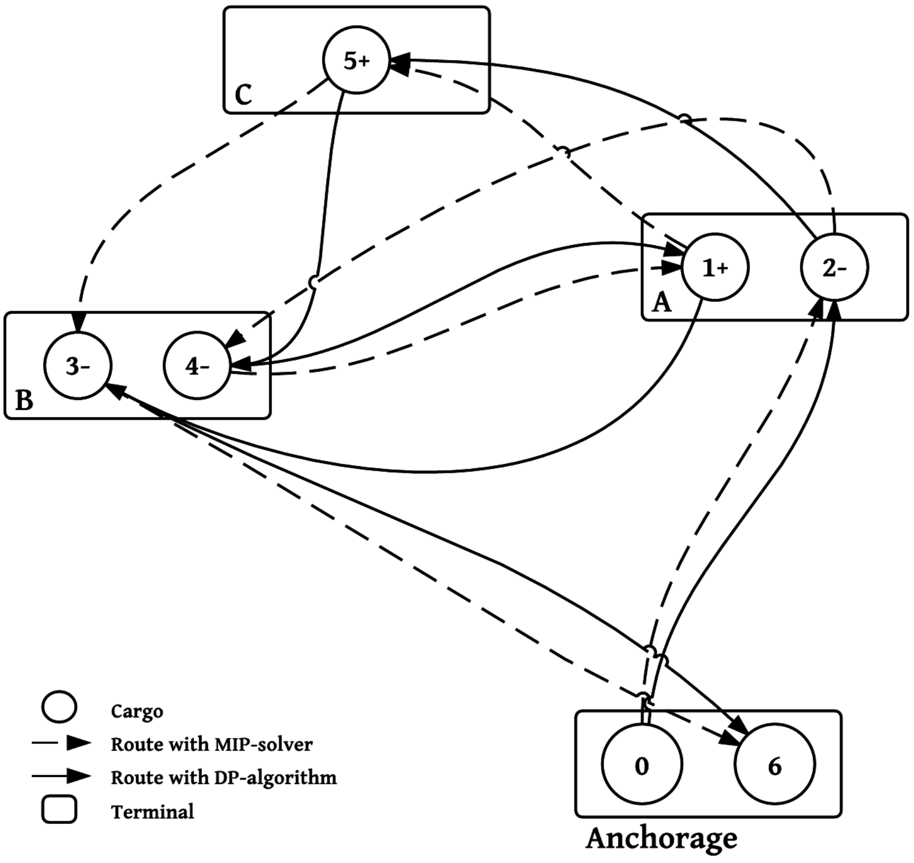


Figure 7.5: Routes provided when the problem is solved both with a MIP-solver and with the DP-algorithm for instance 5_2_3_2

For both methods, the same preprocessing is done on the dataset in order to remove impossible arcs and to perform symmetry breaking. The DP-algorithm allows dominance checks and different rules to be done during the extension of paths, excluding some solutions. Because symmetry breaking is performed during the extension of paths, the DP-algorithm can rule out a lot more symmetrical solutions than what is done when the MIP-solver is used. As mentioned, the best route provided by the MIP-solver is not necessarily the same as the one given by the DP-algorithm. Several optimal routes exist, and the DP-algorithm may discard some of the many symmetrical solutions.

How many cargoes the case company handles during a port stay vary a great deal, but are only in exceptional cases more than 25 cargoes. The DP-algorithm is able to handle instances of realistic sizes better than the MIP-solver, finding optimal routes also for the bigger instances. The DP-algorithm is considered to solve the problem within a tolerable run time, especially compared to the MIP-solver, given that the problem is solved for instances similar as the ones tested in this study.

To summarize, the DP-algorithm performs better than the MIP-solver and is able to solve the problem for instances of realistic sizes. The run times are greatly reduced and the DP-algorithm manages to find optimal solutions for instances where the MIP-solver cannot prove optimality.

The next sections provide a further analysis of the results obtained when the problem was solved using a MIP-solver and a DP-algorithm, respectively.

7.4.2 Analysis when the problem without tank allocation is solved using a MIP-solver

The results obtained when the MIP-solver is used to solve the problem show that optimal solutions are found for all the 5- and 8-cargoes instances in a short time. When the number of cargoes is increased to 12, it takes significantly longer time to solve the problem to optimality. When the number of cargoes is further increased to 15 nodes, the solver cannot guarantee to have found optimal solutions before the cut-off time is exceeded. This also applies for the instances 12_6_6_5 and 12_6_6_6. For these instances, gaps between 0.8 and 42.6% are obtained.

Within each group of instances, there may be large variations in run times. The reason for this is differences in how restricting the data instances are. The significant increases in run times when the number of cargoes is increased is because the number of possible routes grows with the number of cargoes in the port. More routes have to be examined in order to find the optimal one. Based on this, the problem was not attempted solved using a MIP-solver for instances with 20, 25, and 30 cargoes.

When a route includes serving all cargoes in a particular terminal sequentially, the order in which they are served makes no difference for the solution and leads to symmetrical solutions. Some of the symmetrical solutions are avoided due to the preprocessing, but not all. Figure 7.6 shows all symmetrical solutions that may appear in a terminal with three cargoes.

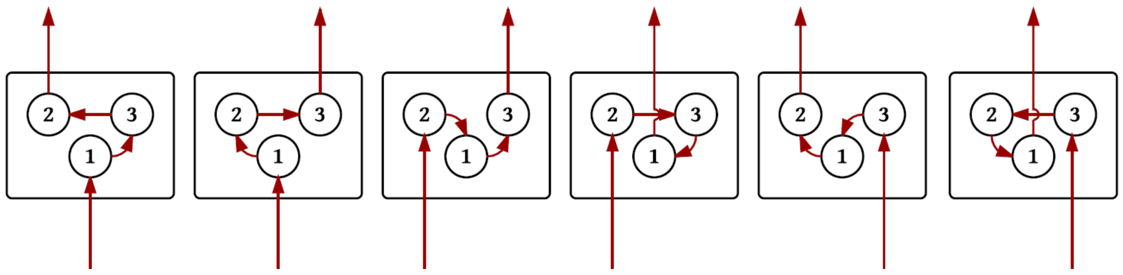


Figure 7.6: Symmetrical solutions within a terminal with three cargoes

The run times the MIP-solver uses to solve the problem for larger data instances are too high to be used in a practical and realistic context. The solution method is able to solve the problem to optimality in a short time for instances of small sizes, but is suffering when instances are of larger sizes. The solver is suffering when the data instances are less restricting (e.g. small cargoes or larger time windows) and therefore have a larger solution space. Knowing that ship routing and scheduling problems with pickup and delivery and time windows are generally computationally hard to solve for relatively larger instance sizes, we are not surprised that the MIP-solver performs poorly when the problem also including draft limits is solved for large instance sizes.

7.4.3 Analysis when the problem without tank allocation is solved using the DP-algorithm

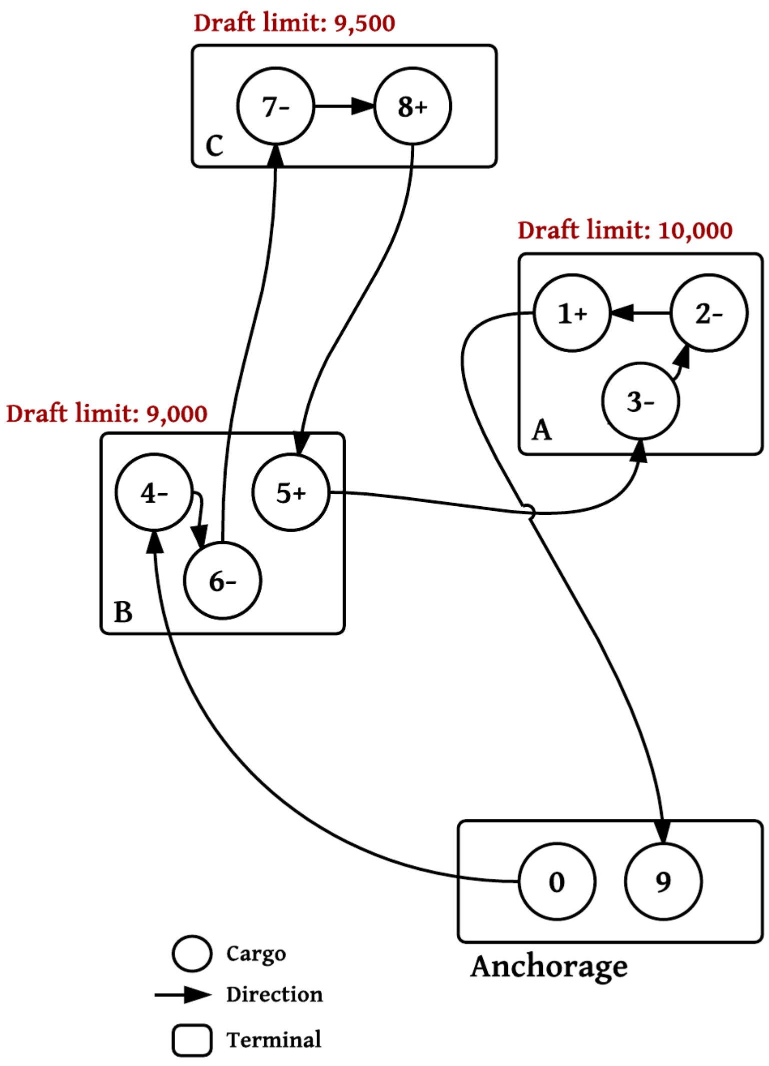
The results in Table 7.8 show that the DP-algorithm finds optimal and feasible solutions for the PDPTW-DL within the set cut-off time for the 5-, 8-, 12-, 15-, 20-, and 25-cargoes instances. For the instances with 30 cargoes, the algorithm can only find optimal solutions for two out of five instances within the set cut-off time. For the instances with 5, 8, 12, 15, and 20 cargoes, the algorithm solves the problem to optimality relatively fast. When the number of cargoes is increased to 25 and 30, the overall run times increase significantly, especially for the instances with 30 nodes. More labels are created, and the dominance step may be executed more times, resulting in longer run times when the number of cargoes is increased. When the problem is solved for the three instances with 30 cargoes that require longer time than the set cut-off time, the solutions are found after 50,439, 32,494, and 49,723 seconds, respectively. Table 7.9 shows the average number of labels generated within each group of instances, clarifying the fact that larger instances result in an increased number of labels created.

Instance group	**Average number of labels generated**
5	33
8	591
12	1,700
15	22,775
20	41,849
25	164,544
30	873,031

Table 7.9: The average number of labels generated within each instance group

It can be seen that the algorithm uses longer time to find the optimal solution when the problem is solved for instances 20_11_9_5 and 25_13_12_4, compared to the other instances with the same number of cargoes. The reason for this can be the composition of pickups and deliveries, their sizes and their location, or the size of the time windows. This data influence the number of labels that are generated. The number of labels generated is 11 times the number of labels generated for instance 25_14_11_1. If the data is such that more extensions are performed, less dominance is possible, or the data results in more possible paths, the algorithm uses longer time to find optimal and all other possible solutions. When the cargo types, cargo sizes, and where they are located are randomly chosen, it can be difficult to evaluate and compare instances because of different execution in the DP-algorithm, and thus the differences in run times are assumed to be somewhat arbitrary. This applies for all groups of instances. In general, one should be careful in drawing conclusions based on differing run times from individual instances. Numerous coincidences can influence the run time, giving a particularly fast/slow search in individual cases.

Figure 7.7 shows the output route from instance 8_3_5_3. This is the optimal route when complying with the time windows, draft limits and other restrictions imposed. The reason why the optimal route visits terminal B two times is because of the draft limit in the terminal. When cargoes 4 and 6 are delivered, the draft limit in terminal B is still too low for the ship to pick up cargo 5. Therefore, the ship must sail to terminal C and deliver cargo 7, which is a large cargo, before it can return to terminal B to pick up the last cargo here. The ship could have sailed directly to terminal C and served cargo 7, but due to different lower time window limits it is better to serve cargoes 4 and 6 in terminal B first.



Arc	Load [t]
0-4	8024
4-6	7292
6-7	6608
7-8	2240
8-5	3332
5-3	7476
3-2	6150
2-1	5236
1-9	9605

Figure 7.7: Route provided when solving the problem for instance 8_3_5_3

When the problem was solved for instance 12_7_5_1, rule 4 was clearly decisive for the solution found. Rule 4 says that as long as both the lower and upper time window limits are lower for a delivery cargo than for a pickup cargo in the same terminal, and that the pickup node are not visited, the ship should visit the delivery node before the pickup node. The delivery cargoes 2, 5, 8, and 10 have smaller upper and lower time window limits than those for the pickup cargoes in the same terminals; 1, 4, 7, and 9. In the optimal solution found, the delivery cargoes 2, 5, 8, and 10 are therefore served before the pickup cargoes in the same terminal.

One of the reasons for lower run times when using the DP-algorithm instead of the MIP-solver, is fewer symmetrical solutions. The number of symmetrical solutions is reduced by including the rules presented in Section 5.2.2, and by executing the dominance algorithm. This reduces the number of possible extensions, and affects the run times in a positive way.

7.5 Testing of the formulation with tank allocation

This section presents the results from solving the problem including tank allocation (PDPTW-DLTA), both when solving it exactly with a MIP-solver and when solving it with the heuristic. First, the choice of objective function for the TAP to include in the heuristic is explained. In Section 7.5.2, the results from testing 40 instances with both solution methods are presented and compared. This is followed by individual analyses of each of the solution methods. All the results presented before Section 7.5.5 are from testing on instances excluding incompatibility. Several instances are tested also when incompatibility between cargoes and tanks are included, analyzed in Section 7.5.5.

7.5.1 Objective function setting

In Section 6.2.2, the two following objective functions were presented as possibilities for the TAP solved as a MIP.

$$minimize \quad 0 \tag{6.22 revisited}$$

$$minimize \quad \sum_{f \in F_i^N} \sum_{i \in N^+} \overline{K_f^F} w_{if,n+1} \tag{6.23 revisited}$$

When solving the PDPTW-DLTA, the objective for the tank allocation is to find a feasible allocation. As feasibility is the primary concern of the TAP, the preferable objective function is the one first finding a solution. When using the objective function (6.23), the implementation in the MIP-solver is such that the program stops when the first solution is found. Recall that the DP-algorithm for solving the PDPTW-DL provides several paths for each instance. The TAP is solved for the paths, starting at the one with the lowest completion time, until a feasible tank allocation plan is found. The two objective functions are tested on 12 instances, and the run times for when the first feasible solutions are found are recorded. As can be seen from Table 7.10, the model seems to be slightly faster when using (6.22) as the objective function, and is therefore chosen to be used in further testing.

	Minimize capacity (6.23)	**Minimize 0 (6.22)**
Instance	Run time	Run time
12_7_5_1	0.3	0.3
12_6_6_2	0.3	0.3
12_7_5_3	0.3	0.3
15_7_8_1	0.4	0.4
15_7_8_2	0.4	0.4
15_7_8_3	0.4	0.4
20_11_9_1	2.4	2.1
20_9_11_2	1.9	1.9
20_11_9_3	2.1	2.1
25_14_11_1	3.4	3.0
25_14_11_2	3.3	3.1
25_14_11_3	3.4	3.2

Table 7.10: Run times for when the first solutions are found when solving the TAP with different objective functions

7.5.2 Results and comparison of the solution methods

Table 7.11 summarizes the test results for the PDPTW-DLTA when the MIP-solver and the heuristic are used, respectively. The cut-off time was set to be 3 hours (10,800 seconds) for the PDPTW-DLTA solved with a MIP-solver and for the PDPTW-DL solved with the DP-algorithm. The TAP solved with a MIP-solver was not assigned a cut-off time because a feasible solution was found, or infeasibility was proved, within a short time. In the table, the run times (Time) for solving the problem using both methods are presented. The objective value (Obj.val.), which is the total time the ship uses in the port, and the lower bound (LBD), which is the optimistic bound for the objective value, are also included. The objective value and the lower bound retrieved when the problem is solved with a MIP-solver, are read directly from the MIP-solver. The objective value of the heuristic corresponds to the completion time of the chosen path, the first path which is found feasible when the TAP is solved. The lower bound of the heuristic corresponds to the completion time of the path using the least amount of time in port. The gaps provided are calculated from the best lower bound found with the two methods, aiming to illustrate how good the objective values actually are.

"Inf." is short for infeasible and "N. s. f." is short for no solutions found. "N/A" is an abbreviation for not applicable, and means that the problem was not solved for that particular instance.

Instance	MIP-solver				Heuristic			
	Obj.val. [hrs]	LBD [hrs]	Gap from best LBD	Time [sec]	Obj.val. [hrs]	LBD [hrs]	Gap from best LBD	Time [sec]
5_3_2_1	Inf.			0.0	Inf.			0.1
5_2_3_2	87.3	87.3	0.0%	0.1	87.3	87.3	0.0%	0.1
5_3_2_3	Inf.			0.0	Inf.			0.0
5_3_2_4	56.0	56.0	0.0%	0.1	56.0	56.0	0.0%	0.0
5_2_3_5	64.0	64.0	0.0%	0.1	64.0	64.0	0.0%	0.0
5_2_3_6	81.3	81.3	0.0%	0.0	81.3	74.8	0.0%	0.1
8_4_4_1	Inf.			88.6	Inf.			0.1
8_4_4_2	77.5	64.4	0.0%	10,800	77.5	77.5	0.0%	0.1
8_3_5_3	103.4	103.4	0.0%	22.5	103.4	103.4	0.0%	0.1
8_4_4_4	80.1	80.1	0.0%	8,882.2	80.1	80.1	0.0%	0.1
8_2_6_5	100.5	100.5	0.0%	34.5	100.5	100.5	0.0%	0.1
12_7_5_1	N. s. f.			10,800	125.6	125.6	0.0%	0.4
12_6_6_2	150.38	107.9	19.3%	10,800	121.4	121.4	0.0%	0.3
12_7_5_3	N. s. f.			10,800	164.6	164.6	0.0%	0.3
12_6_6_4	139.9	117.9	12.7%	10,800	122.1	122.1	0.0%	0.4
12_6_6_5	179.0	110.5	13.6%	10,800	154.6	154.6	0.0%	0.3
12_6_6_6	159.8	109.4	17.4%	10,800	131.9	131.9	0.0%	0.4
12_6_6_7	N. s. f.			10,800	123.4	123.4	0.0%	0.4
15_7_8_1	N. s. f.			10,800	157.4	157.4	0.0%	1.5
15_7_8_2	196.9	110.5	22.9%	10,800	151.9	151.9	0.0%	3.4
15_7_8_3	199.6	107.0	25.2%	10,800	149.4	149.4	0.0%	3.2
15_7_8_4	216.1	110.5	26.9%	10,800	157.9	157.9	0.0%	1.7
15_11_4_5	N. s. f.			10,800	115.4	115.4	0.0%	2.8
15_6_9_6	N. s. f.			10,800	161.1	161.1	0.0%	1.6
15_7_8_7	N. s. f.			10,800	148.4	148.4	0.0%	2.5
20_11_9_1	N/A				284.9	284.9	0.0%	2.2
20_9_11_2	N/A				333.3	333.3	0.0%	3.7
20_11_9_3	N/A				311.6	311.6	0.0%	4.4
20_9_11_4	N/A				320.8	320.8	0.0%	8.3
20_11_9_5	N/A				292.9	292.9	0.0%	34.1
25_14_11_1	N/A				403.4	403.4	0.0%	8.1
25_14_11_2	N/A				406.3	406.3	0.0%	15.3
25_14_11_3	N/A				396.4	396.4	0.0%	34.3
25_13_12_4	N/A				400.9	400.9	0.0%	476.4
25_16_9_5	N/A				399.5	399.5	0.0%	100.6
30_16_14_1	N/A				464.7	464.7	0.0%	1,239
30_15_15_2	N/A				N. s. f.			10,800
30_16_14_3	N/A				464.4	464.4	0.0%	4,850
30_13_17_4	N/A				N. s. f.			10,800
30_14_16_5	N/A				N. s. f.			10,800

Table 7.11: The test results from solving the PDPTW-DLTA with a MIP-solver and the heuristic, respectively

With respect to run times, the heuristic is undoubtedly superior over the MIP-solver. The results indicate that the heuristic is successful when solving problems for up to 25 cargoes, whereas the MIP-solver is barely successful when solving problems for up to 8 cargoes, within the set time frame. Note that this indication applies to data sets which are similar or more restrictive than the data sets tested, as the run times are greatly affected by how restrictive the data is. The fact that the heuristic is splitting the problem into two parts; first finding ship routes, and then a feasible tank allocation, as opposed to the MIP-solver which is solving this simultaneously, has an evidently important effect on run times. When finding feasible routes, the DP-algorithm is able to exclude symmetrical solutions and extensions of paths that are inexpedient. This leads to a decrease in the number of possibilities necessary to explore. When handling the routing and tank allocation simultaneously, there are more possibilities to examine which is unsurprisingly affecting the run times.

For all instances where a solution was found, the gaps are 0%, meaning that the optimal solution was found. All in all, the optimal solution was found for 34 out of 37 tested and feasible instances when using the heuristic, as opposed to 8 out of 22 tested and feasible instances for the MIP-model. This demonstrates that the heuristic is a more solid solution method. However, the heuristic does not provide any solutions before the optimal one is found. This is a drawback compared to the MIP-solver, which specifies the best solution found when the set cut-off time is reached.

The gaps presented in Table 7.11 are calculated from the best lower bound of the two solution methods. If the gaps had been calculated from the objective value and the optimistic bounds of their respective solution method, eight of the instances run with the MIP-solver would have higher gaps while only one of the instances solved with the heuristic would have higher gaps. This indicates that the heuristic provides better lower bounds than those obtained from solving the MIP-problem. The lower bound of the heuristic corresponds to the completion time of the fastest path provided by the DP-algorithm. For all instances except two, the tank allocation problem was solved for the first path generated by the DP-algorithm. For the remaining, the tank allocation was solved for the second path. For one of the instances, the second path had the same completion time as the first one, and thus the gap is the same regardless of which lower bound it is calculated from. For the other instance, the second path had a higher completion time than the first, explaining that the lower bound is lower than the solution found, see instance 5_2_3_6.

If we had not solved the problem to optimality for the 5_2_3_6 instance using a MIP-solver, we could not be sure that 81.3 hours actually was the optimal solution. The heuristic could have excluded a path allowing the tank allocation to be feasible with a completion time higher than the completion time of path number one, but lower than the completion time of path number two. There does not exist a path with the same completion time as path number one that makes the tank allocation feasible that would have been excluded, i.e. it would have been saved. The domination criteria is such that the the path best suited to ensure possible tank allocation is saved when comparing symmetrical solutions. Asymmetrical solutions with equal completion time are all saved.

The data sets in the generated instances are, as far as possible, chosen to be within reason of what the shipping company may experience, e.g. the geography is real, the tank sizes are based on real ship info and, the time windows are fairly wide. The heuristic is much better suited to cope with instances of larger sizes than a MIP-solver, and as real instances may reach 25 cargoes, and only exceptionally more, the heuristic is more appropriate when solving realistic instances.

7.5.3 Analysis when the problem with tank allocation is solved with a MIP-solver

Table 7.11 shows the results when the PDPTW-DLTA is solved with a MIP-solver. The problem is implemented directly as modeled in Section 4.3, finding the best route and a feasible tank allocation simultaneously. The problem is not solved for instances consisting of 20, 25, and 30 cargoes, because the problem without tank allocation neither was solved for these instances.

The results show that the MIP-solver finds optimal solutions in a short time when the problem is solved for the 5-cargoes instances. As expected is the run times increasing when the number of cargoes is increased to eight, and when the number of cargoes is increased to 12, the MIP-solver cannot guarantee to have found optimal solutions for any of the instances. For three of the instances with 12 cargoes and four of the instances with 15 cargoes, the solver could not find any feasible solution within the set cut-off time. For the remaining instances, the solver provides solutions with gaps. When calculating gaps with the best lower bound of the two solution methods, the gaps are between 12.7% and 26.9%. When using the lower bounds provided by the MIP-solver only, the gaps are between 15.7% and 48.9%. This indicates that the MIP-solver is finding solutions to the problem that is better than what it is able to prove.

It can be seen from Table 7.11 that the problem is infeasible for several instances in the 5- and 8-cargoes instance groups, meaning that there are no routes that can be found given the restrictions imposed. In some cases, the routes are feasible with respect to time windows and draft limits, but due to tank and cargo specific restrictions, the problem is infeasible.

The results show that the solver finds optimal solutions relatively fast when the problem is solved for instances 8_3_5_3 and 8_2_6_5, compared to the other instances with 8 cargoes. One reason for this can be the number and/or the sizes of delivery cargoes. In instances 8_3_5_3 and 8_2_6_5 there are five and six delivery cargoes, respectively. Because these cargoes are on board the ship and occupy all tanks when leaving the anchorage, there are less options for which terminals the ship can approach. This is compared to the other feasible instances, where one tank is available for loading and the ship can sail to serve both pickup and delivery cargoes. With more opportunities for cargoes to serve, due to one or several available tanks, the run time increases in order to examine all these possible routes. This argument emphasizes the fact that the run times are highly dependent on the input and how restricting the data is.

Even though the solver uses a long time to find optimal solutions for some of the instances with eight cargoes, the solver finds feasible solutions within relatively short time. This is shown in Table 7.12, which presents the run times and corresponding times for when the best solutions were found. The best solutions found can be compared to the solutions obtained when the problem without tank allocation was solved. By doing this, it it possible to evaluate how good the feasible solutions are without having to wait for the solver to prove optimality, knowing the gap between the best solution found and the optimistic one. However, if the PDPTW-DLTA is solved without knowing the solutions when tank allocation is disregarded, this is not a possibility.

Instance	**Run time** *[sec]*	**Time when best solution found**
8_4_4_1	Infeasible	-
8_4_4_2	10,800	1.2
8_3_5_3	22.5	1.2
8_4_4_4	8,882 3	96.8
8_2_6_5	34.5	1.8

Table 7.12: Comparison between the run times and when the best solutions are found

From this analysis, it is discovered some positive features by using the MIP-solver. When solving the problem for instances where a solution is found, the best solution is found quite early. Also, when solutions are provided with a gap, the solutions found are implied to be better than what can be proved by the MIP-solver. However, the conclusion of how the solver performs is the same as the one when tank allocation was disregarded. The MIP-model suffers when the number of cargoes increases or the data imposed is less restricting. When tank allocation is added to the problem, the model suffers for even smaller instances than the problem without tank allocation. Assuming that the model is run without knowing the solutions from the PDPTW-DL, the run times are too high for the model to be used in a practical and realistic context.

7.5.4 Analysis when the problem with tank allocation is solved with the heuristic

In this section, we analyze the results from solving the problem with tank allocation (PDPTW-DLTA) using the heuristic, presented in Table 7.13. As the heuristic consists of solving the pickup and delivery problem with time windows and draft limits (PDPTW-DL) by using the DP-algorithm solving the TAP by using a MIP-solver, the results are presented for each of these problems. The run times (Time) and the objective values corresponding to the completion time of the chosen paths (Obj. val.) for solving the PDPTW-DL and the TAP, respectively, are presented. The total number of labels generated by the DP-algorithm and the number of completed labels, i.e. the number of feasible paths provided from solving the PDPTW-DL, are also given. For the TAP, the number of nodes explored in the branch and bound tree by the solver and the chosen path are given. "1" corresponds to the path with the lowest completion time, "2" to the path with the next lowest completion time and so on. If more paths are explored in the TAP, the run times in the table are the sum of the run times of solving each path. The number of nodes explored when the MIP-solver is used corresponds to the number of nodes of the chosen path. "N. s. f." is short for no solution found. "Inf." means that the TAP is infeasible for all of the given paths.

Instance	**PDPTW-DL**				**TAP**			
	Obj.val. [hrs]	# of gen. labels	# of paths	Time [sec]	Obj.val. [hrs]	# of nodes	Chosen path	Time [sec]
5_3_2_1	66.3	35	2	0.0	Inf.			0.1
5_2_3_2	87.3	28	3	0.0	87.3	1	2	0.0
5_3_2_3	56.0	31	1	0.0	Inf.			0.0
5_3_2_4	56.0	52	2	0.0	56.0	1	1	0.0
5_2_3_5	64.0	29	1	0.0	64.0	1	1	0.0
5_2_3_6	74.8	24	2	0.0	81.3	1	2	0.1
8_4_4_1	94.8	699	1	0.0	Inf.			0.1
8_4_4_2	77.5	1,138	3	0.1	77.5	1	1	0.1
8_3_5_3	103.4	353	1	0.0	103.4	1	1	0.1
8_4_4_4	80.1	770	4	0.0	80.1	1	1	0.1
8_2_6_5	100.5	333	1	0.0	100.5	1	1	0.1
12_7_5_1	125.6	2,194	6	0.0	125.6	1	1	0.3
12_6_6_2	121.4	2,749	4	0.0	121.4	1	1	0.3
12_7_5_3	164.6	238	6	0.0	164.6	1	1	0.3
12_6_6_4	122.1	4,348	8	0.1	122.1	1	1	0.3
12_6_6_5	154.6	886	5	0.1	154.6	1	1	0.3
12_6_6_6	131.9	4,852	6	0.1	131.9	1	1	0.3
12_6_6_7	123.4	4,057	2	0.1	123.4	1	1	0.3
15_7_8_1	157.4	30,898	3	1.1	157.4	1	1	0.4
15_7_8_2	151.9	52,396	7	3.0	151.9	1	1	0.4
15_7_8_3	149.4	53,476	16	2.8	149.4	1	1	0.4
15_7_8_4	157.9	33,760	7	1.3	157.9	1	1	0.4
15_11_4_5	115.4	47,049	3	2.1	115.4	1	1	0.7
15_6_9_6	161.1	33,341	8	1.2	161.1	1	1	0.4
15_7_8_7	148.4	25,312	1	0.7	148.4	1	1	1.8
20_11_9_1	284.9	3,125	2	0.1	284.9	1	1	2.1
20_9_11_2	333.3	36,728	3	1.8	333.3	1	1	1.9
20_11_9_3	311.6	45,300	1	2.4	311.6	1	1	2.1
20_9_11_4	320.8	74,737	4	6.3	320.8	1	1	2.0
20_11_9_5	292.9	191,814	6	32.0	292.9	1	1	2.0
25_14_11_1	403.4	61,823	13	5.0	403.4	1	1	3.0
25_14_11_2	406.3	101,902	8	12.1	406.3	1	1	3.1
25_14_11_3	396.4	168,561	3	31.1	396.4	1	1	3.2
25_13_12_4	400.9	724,211	2	473.3	400.9	1	1	3.0
25_16_9_5	399.5	252,402	2	97.3	399.5	1	1	3.3
30_16_14_1	464.7	1,113,244	4	1,236	464.7	1	1	3.6
30_15_15_2	N. s. f			10,800	N. s. f			
30_16_14_3	464.4	1,895,752	4	4,847	464.4	1	1	3.5
30_13_17_4	N. s. f			10,800	N. s. f			
30_14_16_5	N. s. f			10,800	N. s. f			

Table 7.13: Test results when the PDPTW-DLTA is solved with the heuristic

As can be seen from the table, the DP-algorithm is solving the problem in a short time for most instances. The run times are not noteworthy increased before the instances have 20 or 25 cargoes. The MIP-solver is solving the TAP in a stable and short time, within four seconds, for all instances. So, the largest proportion of the total run time comes from solving the PDPTW-DL part of the problem with the DP-algorithm. The proportion of the total time that comes from solving the PDPTW-DL compared to the TAP is increasing significantly as the number of cargoes is increased.

The most important finding from Table 7.13 is that the heuristic seems to work well for instances of realistic sizes. The heuristic is stable, providing optimal solutions for nearly all the tested instances. The TAP is solved within a short run time for all instances. The fact that the run time is so low even for the instances of the largest sizes, may be because the ship in these instances consist of many tanks, leading to many stowage options.

Table 7.13 presents the number of completed paths from the PDPTW-DL, i.e. the number of feasible paths from solving the problem without tank allocation. It can be seen that the number of completed paths is ranging from 1 to 16, depending on the instance tested. One example is instance 12_6_6_2, where four completed labels were saved, shown in Table 7.14. The first three labels have equal objective values but different routes, i.e. they are asymmetrical solutions. Label 4 provides a route with a longer completion time than the routes from labels 1-3. Even though label 4 has a longer completion time, it may secure feasibility and is therefore saved. Thus, all the completed labels are used as input to the TAP.

Label	**Obj.val.** *[hrs]*	**Route**
1	121.4	[0 5 4 2 1 3 6 9 11 10 12 8 7 13]
2	121.4	[0 4 2 1 3 5 6 9 11 10 12 8 7 13]
3	121.4	[0 5 4 2 1 3 9 11 10 12 6 8 7 13]
4	125.6	[0 2 1 5 4 6 9 11 10 12 8 7 3 13]

Table 7.14: Example from solving instance 12_6_6_2. The path from label 4 uses longer time than the paths from labels 1-3.

It can be seen from Table 7.13 that the problem is infeasible for three instances, i.e. it was not possible to find a feasible tank allocation for any of the paths found by the DP-algorithm. As the solution method is a heuristic which possibly can have excluded feasible paths for the TAP, this do not prove that the instances are infeasible for the PDPTW-DLTA.

The number of nodes explored in the branch and bound tree when solving the TAP is one for all instances. The first node explored by the MIP-solver is the LP-relaxation of the problem. This means that for all instances tested in this problem, the LP-relaxation is equal to a feasible solution, i.e. there is no gap between an instance's LP-solution and a feasible IP-solution.

If we disregard the three infeasible instances and the instances 5_2_3_2 and 5_2_3_6, all solutions for the TAP are found for the first path generated by the DP-algorithm. The reason for this is that the ship, in most instances, are not so full that the allocation of cargoes to tanks becomes problematic. The domination criteria in the DP-algorithm are such that the paths are able to handle the TAP as best as possible, and the total ship capacity is taken into account when finding the paths. So, the reason that the first path is not chosen is because of a lack of number of available tanks and/or their tank capacity. This applies for the instances 5_2_3_2 and 5_2_3_6.

To look at an example where not the first path is chosen, consider instance 5_2_3_6. In this instance, a ship with three tanks is used. These tanks are all occupied by delivery cargoes when entering the port. Path number one, with the completion time of 74.8 hours found by the DP-algorithm, is visiting a delivery node, and then a pickup node in the same terminal as the first cargo. The sum of the delivery loads and this pickup load do not exceed the ship capacity or any relevant draft limits, but the available tank is too small to pick up the second cargo. Path number two, with the completion time of 81.3 hours, is serving the same delivery cargo first, but is then serving another delivery cargo in a different terminal. This makes the tank allocation feasible, and the solution to the PDPTW-DLTA for this instance is therefore 81.3 hours. This is illustrated in Figure 7.8.

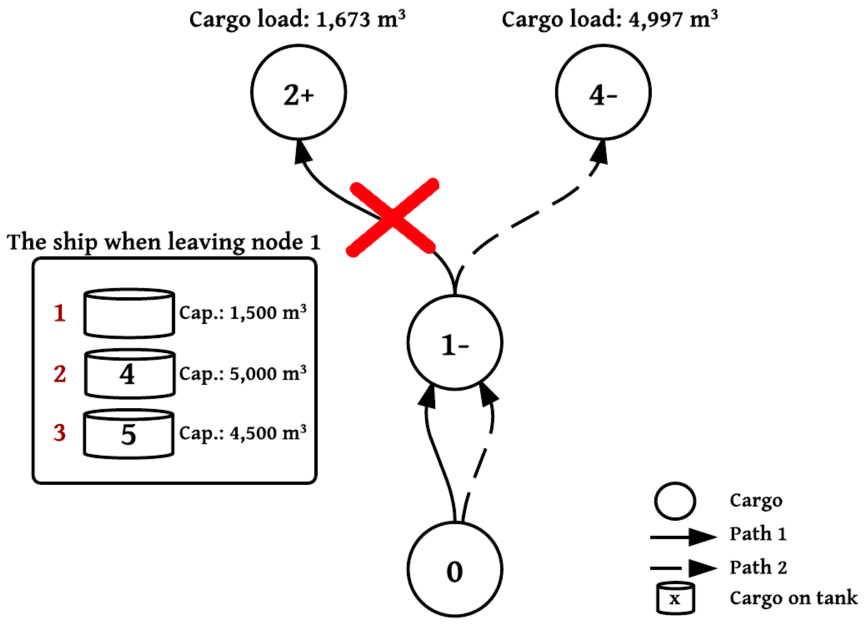


Figure 7.8: Path 1 is not feasible when tank allocation is added, because tank 1 has lower capacity than the volume of cargo 2

Because the MIP-solver also found the solution of 81.3 hours for this instance, we know that the solution is optimal. If it had not, we would have recognized that a solution up to 6.5 hours faster might exist, without knowing its path. In case the first path appears infeasible, it could have been considered preferable to keep more paths with a completion time not far from the best path. This could be done by changing the domination criterion in the DP-algorithm. When comparing two labels standing in the same terminal and having visited exactly the same nodes, one label could dominate another if the completion time of that label is a set time limit higher than the completion time of the other, as opposed to if it is any higher at all. If this time limit is set to be three hours, solutions three hours slower than the best path found would have been saved. For the instance considered in this example, we would have more knowledge about how good the solution found is. However, due to an increase in the number of labels generated in the DP-algorithm, the run time would naturally increase. This possibility has not been tested and evaluated in this thesis.

Further testing

The ship type used in the 15-cargo instances consists of 22 tanks, and the type used in the 20-, and 25-cargo instances consists of 52 tanks. Both the total weight of the delivery cargoes and the total weight of the pickup cargoes are between 60 and 90% of the total ship capacity. The solutions obtained when the problem is solved with tank allocation are to a large extend the same as the solutions for the problem without, due to the many stowage options. It is therefore considered interesting to examine how fast the TAP is solved when the total weights of the pickup and delivery cargoes are increased, and hence decreasing the number of available tanks at each node.

Based on three already existing instances, three new instances are generated. Now, both the total weights of the deliveries and pickups each are increased to be from 90 - 97% of the total ship capacity, with the intention of occupying more tanks throughout the route. The draft limits are randomly chosen to be between 90 and 100% of the ship's total capacity, to avoid infeasible routes due to draft limits. Table 7.15 presents the results when the heuristic was used for solving the PDPTW-DLTA, first finding complete paths and then finding a feasible tank allocation.

Instance	**PDPTW-DL**				**TAP**			
	Obj.val. [hrs]	# of gen. labels	# of paths	Time [sec]	Obj.val. [hrs]	# of nodes	Chosen path	Time [sec]
15_7_8_3	149.6	41,802	13	1.9	149.6	1	1	0.5
20_11_9_5	295.6	60,349	6	3.6	295.6	1	1	2.1
25_14_11_1	392.5	74,453	42	7.5	392.5	1	1	3.7

Table 7.15: Test results from solving the PDPTW-DLTA using the heuristic on additional instances with increased total load

When the problem was solved for these instances, an increase of load on board the ship does not impose a higher run time. As seen from the table, the TAP was solved for the first paths provided by the DP-algorithm. These results do not imply that the TAP is more difficult to solve if there are fewer tanks available and hence fewer stowage options. We tried to increase the loads even more, but could not find an example where the PDPTW-DL was solved and the TAP either was infeasible or solved for paths other than the first. This indicates that the DP-algorithm provides paths that make the TAP easy to solve, and that the heuristic works well even for instances where the ships are more fully loaded.

Improving the allocation of cargoes to tanks

After solving the PDPTW-DLTA for an instance and finding a feasible solution, the TAP can be solved again in order to improve the allocation of cargoes to tanks with the aim of securing a better starting point for the next port call. After the path with the lowest completion time is found, the problem with the objective function (6.23), minimizing occupied capacity in the last sequence, can be solved for this particular path with a set cut-off time. There might exist more than one path with the same completion time, and it could be considered to solve the TAP again for all these paths in order to choose the one with the best allocation of cargoes when leaving the port.

If the TAP should be solved again to improve the allocation of cargoes, for which cut-off times, and if paths with the same completion time should be examined, is up to the shipping company to decide. The value of the solutions must be compared to the added inconvenience in terms of run times.

This section is included to show that the TAP with the objective function of minimizing capacity can be used to provide advantageous solutions in a rather short time. Table 7.16 summarizes the results from testing the TAP when the capacity is minimized, with a cut-off time of five minutes (300 sec), compared to the results from when the first feasible tank allocation plan is accepted. The testing is done on 12 instances with both objective functions. For each instance the results presented are the run times and the gap between the best solution found and the lower bound for when the capacity is minimized. Additionally, the number of nodes examined in the branch and bound tree used by the solver when the model is solved with both objective functions is presented, as well as the occupied capacity in the last sequence, both in terms of number of tanks used and occupied volume.

Instance	**Minimize capacity (6.23)**					**Minimize 0 (6.22)**			
				Cap. occupied				Cap. occupied	
	Run time	Gap	# of nodes	#of tanks	$[m^3]$	Run time	# of nodes	#of tanks	$[m^3]$
12_7_5_1	1.1	0.0%	1	17	16,296	0.3	1	18	20,512
12_6_6_2	35.0	0.0%	37,602	16	15,834	0.3	1	19	19,830
12_7_5_3	1.0	0.0%	1	17	16,328	0.3	1	19	19,862
15_7_8_1	300.0	0.1%	93,845	19	19,113	0.4	1	22	21,697
15_7_8_2	1.2	0.0%	15	15	13,786	0.4	1	18	18,683
15_7_8_3	0.7	0.0%	1	19	17,960	0.4	1	20	21,697
20_11_9_1	300.0	0.4%	53,349	39	25,377	2.1	1	42	34,694
20_9_11_2	300.0	0.4%	72,927	39	29,264	1.9	1	42	35,709
20_11_9_3	300.0	1.4%	20,374	46	32,856	2.1	1	44	38,001
25_14_11_1	300.0	0.4%	7,488	46	29,683	3.0	1	46	38,645
25_14_11_2	300.0	2.9%	3,642	42	29,776	3.1	1	47	37,752
25_14_11_3	300.0	1.3%	4,810	42	26,191	3.2	1	46	36,450

Table 7.16: Results from when capacity is minimized compared to when the first feasible solution is found for the TAP

Note that the total capacity of the ship in the 12- and 15-cargo instances is 21,390 m^3. The ship used in the 20- and 25-cargo instances has a total capacity of 40,778 m^3.

In all of the instances presented above, the capacity used in the last sequence is unsurprisingly bigger when using (6.22) as the objective function instead of (6.23), both in terms of number of tanks used and occupied volume. Table 7.17 shows an example of how the tank allocations are in the end node when the two different objective functions are used for instance 15_7_8_2. The table contains information about which cargoes that are allocated to which of the 22 tanks on board the ship. Blank spaces correspond to empty tanks.

		Minimizing capacity (6.23)		Minimizing 0 (6.22)	
Tank	**Max. capacity tank [m^3]**	Cargo number	Volume on tank [m^3]	Cargo number	Volume on tank [m^3]
1	624.0	6	537.3	2	188.7
2	611.0	2	611.0	3	611.0
3	1,220	6	1,220	6	1,220
4	1,234	9	1,225		
5	1,242			8	442.6
6	1,242	9	1,242	12	371.5
7	592.0	12	371.5	10	592.0
8	593.0	8	593.0		
9	1,187	9	1,187		
10	1,189	6	1,189	6	452.3
11	1,273			10	20.8
12	1,274			9	426.9
13	1,273			9	1,273
14	1,274			6	1,274
15	686.0	3	538.7	9	686.0
16	671.0	10	671.0		
17	1,181	8	1,031	3	1,109
18	1,181	3	1,181	8	1,181
19	1,268	10	1,210	9	1,268
20	1,268			10	1,268
21	307.0	2	191.7	2	307.0
22	307.0			2	307
Total capacity occupied			13786.0		18012.0

Table 7.17: Tank allocation in the last sequence for instance 15_7_8_2

The table shows an example of how much the tank allocation can be improved by using (6.23) on an instance after it is found feasible. The number of available tanks when using (6.22) is four, as opposed to seven when using (6.23). The total capacity occupied is 64.5% of the total capacity when using (6.23), as opposed to 84.2% of the total capacity when using (6.22). The flexibility of having available capacity is of great value to the shipping company.

So, if it is of interest to the shipping company to optimize a ship's tank allocation plan after finding a feasible route, objective function (6.23) can be used and the TAP can be solved again for the particular instance and ship route.

7.5.5 Tank allocation and incompatibility

As explained in Section 7.1, incompatibility is specified by introducing cargo types and tank types. A tank type is determined based on tank linings, and a cargo type is determined based on its compatibility to one or more tank type(s). Incompatibilities between particular cargoes and tanks are specified for 18 instances and tested with the heuristic. The testing was not done using the MIP-solver, because the run time would be inappropriately long for most of these instances. The test results are shown in Table 7.18. The results obtained when the problem is solved without incompatibility for the same instances are included in the same table for comparison. Only the run times for solving the TAP are included, because the run times when solving the PDPTW-DL with the DP-algorithm do not depend on the incompatibility.

Instance	**TAP *without* incomp.**		**TAP *with* incomp.**	
	Completion time *[hrs]*	Run time *[sec]*	Completion time *[hrs]*	Run time *[sec]*
5_3_2_1	Infeasible	0.1	Infeasible	0.0
5_2_3_2	87.3	0.0	Infeasible	0.1
8_4_4_1	Infeasible	0.1	Infeasible	0.0
8_4_4_2	77.5	0.1	Infeasible	0.0
12_7_5_1	125.6	0.3	125.6	0.3
12_6_6_2	121.4	0.3	121.4	0.3
15_7_8_1	157.4	0.4	157.4	0.4
15_7_8_2	151.9	0.4	151.9	0.4
20_11_9_1	284.9	2.1	Infeasible	0.0
20_9_11_2	333.3	1.9	333.3	1.8
20_11_9_3	311.6	2.1	311.6	1.9
20_9_11_4	320.8	2.0	320.8	2.2
20_11_9_5	292.9	2.0	292.9	2.0
25_14_11_1	403.4	3.0	403.4	3.0
25_14_11_2	406.3	3.1	406.3	3.1
25_14_11_3	396.4	3.2	396.4	3.0
25_13_12_4	400.9	3.0	400.9	3.8
25_16_9_5	399.5	3.3	399.5	78.8

Table 7.18: Test results from solving the TAP without and with incompatibility

The results show that the problem becomes infeasible for three additional instances, to those already infeasible without tank allocation. The problem is infeasible for instance 8_4_4_2 because there are not enough compatible tanks to allocate the cargoes initially. Without incompatibility, the problem found a feasible initial tank allocation and a solution for this instance. This implies that instance 8_4_4_2 is unrealistic. The ship owner would have to assign a different ship type for this particular journey, because the ship would not be able to initially allocate the cargoes that are to be delivered. For instances 5_2_3_2 and 20_11_9_1, the problem is infeasible because there are not enough compatible tanks to sail any of the given routes and allocate the cargoes in a feasible way.

It can be seen that the run times for several of the instances are slightly reduced when incompatibility is added. The reason for this can be a smaller solution space, due to more restrictions and fewer variables, implying fewer possibilities for where the cargoes can be allocated. There also exists some instances that result in higher run times when the problem is solved with incompatibility. Even though there are fewer possible solutions, the solver can require longer time to prove optimality because more aspects must be taken into account within each possible solution. This may apply for instance 25_16_9_5, where the solver uses 75.5 seconds more to find a feasible tank allocation when incompatibility is included.

For all the instances where the problem finds feasible tank allocations, the solutions are the same as those obtained when the problem was solved without incompatibility. Several of the cargoes are compatible with both types of tanks, and the ship has at certain occasions many tanks available, such that the incompatibilities are not preventing the same paths to be feasible.

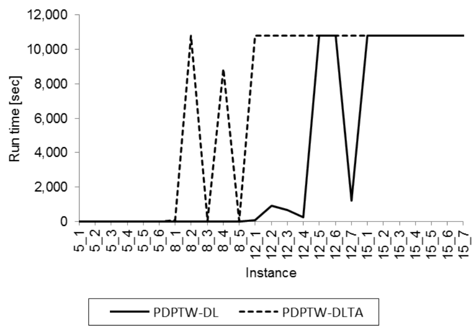
Except for instance 25_16_9_5, the run times for when the problem is solved for the remaining instances are not increased significantly when adding incompatibility. Solving the TAP without incompatibility may result in tank allocations that in reality are infeasible due to tank and cargo restrictions. This information may be crucial for the shipping company. Hence, including incompatibility provides a better and more realistic tank allocation without greatly influencing the run times or possible routes sailed.

7.5.6 Real case

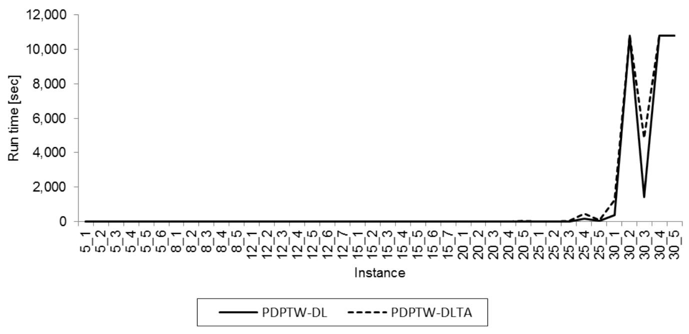
As an intended part of the computational study, a real-case port call provided by the case company was analyzed. An instance was made with the information provided and the problem was solved using the heuristic. The instance was made with the purpose of comparing the model outputs to the route actually sailed. The model suggests a route marginally better than the route actually sailed, but due to characteristics about this instance the comparison gets somewhat problematic. The analysis provide a good impression of how our model relates to the chain of events experienced during a real port call, but as it is hard to draw any conclusions from the testing, the analysis is placed in Appendix B.

7.6 Comparison between the models with and without tank allocation

When the results from the model without tank allocation (PDPTW-DL) are compared to the results with (PDPTW-DLTA), it is clear that the time required finding optimal solutions increases when tank allocation is added. Figure 7.9 illustrates how the run times are affected when tank allocation is included, comparing the run times for the PDPTW-DL and PDPTW-DLTA for both solution methods. Note that the names of the instances in the two figures have been shortened by excluding the number of pickups and deliveries.



(a) MIP-solver



(b) Heuristic

Figure 7.9: Graphic illustration of run times when the PDPTW-DL and PDPTW-DLTA are solved with both solution methods

If Figures 7.9a and 7.9b are compared, it is evident that the run times when using the heuristic are significantly lower than those incurred when the MIP-solver was used, emphasizing the fact that the heuristic is the superior solution method. This was discussed in Sections 7.4.1 and 7.5.2. It can also be seen that the differences in run times when solving the PDPTW-DL and PDPTW-DLTA, respectively, are less significant when the heuristic is used.

The solution method of using a MIP-solver performs fairly differently when comparing the model with and without tank allocation. With tank allocation, the number of possible solutions increases drastically because of the routing and stowage are done simultaneously, causing increased run times. This can be exemplified with instance 8_4_4_2, where the solver is not able to prove optimality for the PDPTW-DLTA within the set cut-off time, compared to the PDPTW-DL where the optimal solution was found in a short time. For several of the instances with 12 and 15 cargoes, the MIP-solver is not able to find feasible solutions within the set cut-off time when tank allocation is added. Without tank allocation, the solver provides solutions with gaps between 0.0% and 42.6% for these instances. Generally, the MIP-solver provides feasible solutions with bigger gaps when tank allocation is added.

As described in Section 6.2.1, the dominance check in the DP-algorithm is a bit more complicated for the PDPTW-DLTA than the one implemented for solving the PDPTW-DL. The examination of finding the best label may influence the run times, forcing a more extensive dominance check than the one performed for the PDPTW-DL. However, this expansion influences the run times to a less extent compared to when the MIP-solver is used to solve the tank allocation simultaneously as the routing.

The problem without tank allocation simplifies the ship to contain one big tank, only considering the total capacity. By not considering the requirement of available and suitable tanks, the real problem is relaxed and simplified. Because of this, some routes might be prohibited when tank allocation is taken into account. Some instances become infeasible, and in this study this applies for the instances 5_3_2_1, 5_3_2_3, and 8_4_4_1. For instance 5_2_3_6, both problems are solved to optimality, but a different route with a higher objective function value is provided when the model with tank allocation is solved. The shortest route found is not in compliance with the tank restrictions imposed, resulting in an alternative route where the cargoes can be allocated to tanks in a rightful manner.

Except for the infeasible instances and instance 5_2_3_6, the best routes, i.e. the first paths, are feasible also when tank restrictions are added. Even though the completion times are the same, the routes may differ. One example is instance 12_7_5_1. The route found with the DP-algorithm when the PDPTW-DLTA is solved, is more suitable for tank allocation than the one found when solving the PDPTW-DL, due to the different dominance criteria used within each problem. The difference occurs when comparing two labels using the same amount of time. The DP-algorithm for solving the problem without tank allocation is saving a randomly chosen label, while the DP-algorithm for solving the problem with tank allocation is solving the label(s) most suitable to handle the TAP.

Figure 7.10 shows the visiting sequences for the optimal paths within terminal D for both problems. It can be seen that the route obtained from the PDPTW-DLTA serves both delivery cargoes first, compared to the path given from the PDPTW-DL, making it easier to perform the tank allocation.

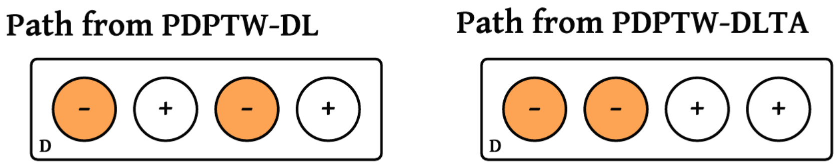


Figure 7.10: The visiting sequences in Terminal B for when the both problems are soved for instance 12_7_5_1

Chapter 8

Practical implications

In this thesis, a pickup and delivery problem with time windows and draft limits (PDPTW-DL) and a pickup and delivery problem with time windows, draft limits and tank allocation (PDPTW-DLTA) are solved and tested with two solution methods each. In Chapter 7, the results were analyzed mainly from a technical perspective. In this chapter, the models and solution methods are discussed from a practical perspective in order to assess advantages and disadvantages seen from a shipping company's point of view.

The problems are modeled such that the total time in port is minimized. This is done because the voyage duration greatly affects the voyage earnings. Providing the routes with the lowest durations may influence the performance of each voyage and the total performance of the shipping company. The models and accompanying solution methods can be used to develop a tool that can be used as support for the shipping company when making its decisions regarding routing and tank allocation.

For the models to be beneficial as a decision tool, they need to take enough aspects into account, making them good representations of the real problems. But they also need to provide solutions within a reasonable run time, depending on the context in which the models are used. A better reality representation will almost always lead to a higher run time. According to Nygreen et al. (1998), the reality representation and solution speed are classical trade-offs in operations research modeling. On the other hand, a model providing a better reality representation will, by assessing more details, require more input. The amount of input needed might get overwhelming, such that the job of collecting it is considered too extensive to adopt the decision support tool. Some of the input may also be hard to collect or are unknown.

The PDPTW-DLTA model is a better reality representation than the PDPTW-DL. The first suggested solution method for both problems is solving them by direct implementation in a commercial MIP-solver, and is an evident example of the mentioned trade-off between run time and reality representation. The solving of the PDPTW-DLTA requires a significantly higher run time than solving the PDPTW-DL. The second suggested solution method is the DP-algorithm for the PDPTW-DL and the heuristic for the PDPTW-DLTA. When comparing the DP-algorithm solving the PDPTW-DL and the heuristic solving the PDPTW-DLTA, the run times are only marginally increased, which is a compelling feature of these methods.

The DP-algorithm and the heuristic were proven superior over the method of using a MIP-solver for the PDPTW-DL and the PDPTW-DLTA, respectively, with respect to run times. The greatest disadvantage of the heuristic is that it might exclude optimal solutions. An extreme case is that the best solution leading to a feasible tank allocation is excluded, and the ship ends up sailing a path with a higher completion time. However, we always have knowledge about the gap, as the lower bound corresponds to the completion time of the fastest route provided by the DP-algorithm. The heuristic seems to be very stable, as nearly all the tested instances in this thesis were solved to optimality.

A positive feature of the solution method of using a MIP-solver is that it provides feasible solutions relatively fast and presents them during the solving of the problems. This is in contrast to the DP-algorithm solving the PDPTW-DL, which do not present any solutions before all possible solutions are found. But, for instances of a certain size, the MIP-solver demands too much run time for solving the problems for it to be beneficial in a practical context. Because the DP-algorithm and the heuristic are able to solve the problems for instances of realistic sizes, they can be seen as more suitable solutions methods in a practical context than using a MIP-solver to solve the PDPTW-DL and the PDPTW-DLTA, respectively. Hence, the further analysis in this section is only considering the DP-algorithm and the heuristic.

The shipping company may choose to use the DP-algorithm to solve the PDPTW-DL for decision support. For most of the instances, this solution method finds the same solutions as the heuristic when solving the PDPTW-DLTA, but the solutions may be too optimistic. In some cases the instances get infeasible when including tank allocation, meaning that the ship is not able to serve all cargoes while complying with all restrictions. If the ship is too late to pick up a cargo, it may lose it, resulting in lost profits and a possible negatively affected relationship between the ship owner and the charterer.

The solution to the PDPTW-DL solved with the DP-algorithm provides an optimistic bound for the solution to the PDPTW-DLTA and is thus lower than what can be guaranteed in reality because the tank allocation is simplified. Due to the risk of getting an infeasible route, in addition to not getting a tank allocation plan, a decision maker might find the PDPTW-DL too incomplete. To achieve flexibility to pick up a various range of products, a chemical tanker is characterized by numerous tanks with different properties. Making an allocation plan is an important part of the planning problem a chemical tanker faces. It might seem like an obvious choice to use the heuristic to solve the PDPTW-DLTA, because it provides a more realistic routing choice and objective value, hardly at any expense of run time. A prerequisite however, is that the needed information for input data is available. For the tank allocation problem, the objective function minimizing the total capacity used can be chosen, as it improves the tank allocation with respect to capacity used without requiring long run times.

The model PDPTW-DLTA allows incompatibility between cargoes and tanks and between different cargoes to be included. Some of the tested instances got infeasible when incorporating incompatibility. This is valuable information for the shipping company, as it might need to reevaluate this port call and possibly consider to use a different ship type. Incorporating this aspect in the model, as done in the PDPTW-DLTA formulation, leads to a more precise decision tool. However, it demands more input from the shipping company. If incompatibility between tanks and cargoes should be considered, tank types and cargo types must be specified. If compatibility between different cargoes should be taken into account, all the tanks that the remaining cargoes could be placed on given each placement of each cargo must be specified, this is a comprehensive job.

As mentioned in Section 4.1, an assumption that the ship at all times is stable and has a reasonable trim was made. If numerous tanks are needed to transport one cargo, the model allocates the volume of this cargo as evenly as possible on these tanks. Beyond this, the models have to be further developed to comply with stability and trim restrictions in a more realistic and rightful manner.

The case company may use the DP-algorithm or the heuristic to support the planning of routing for a particular port call, by running it before entering the port, and obtain the best route given the current information. However, it may be convenient to run the model several times during the port call. This is applicable if the shipping company receives new information about the input parameters during the port call, especially about the waiting times in the terminals. As described in Section 4.1, the ship will, after sending out tenders, receive berth advices containing assumed waiting times for the terminals. The ship will follow the procedure of sending out tenders and receiving updated berth advices after each terminal visit. It is therefore natural to assume that the case company can benefit from being able to run the model between the service of the different cargoes, as long as the run times are low enough. After serving a cargo, the ship is able to find a new, optimal route based on the new information, knowing that some cargoes already have been served. This process is illustrated in Figure 8.1.

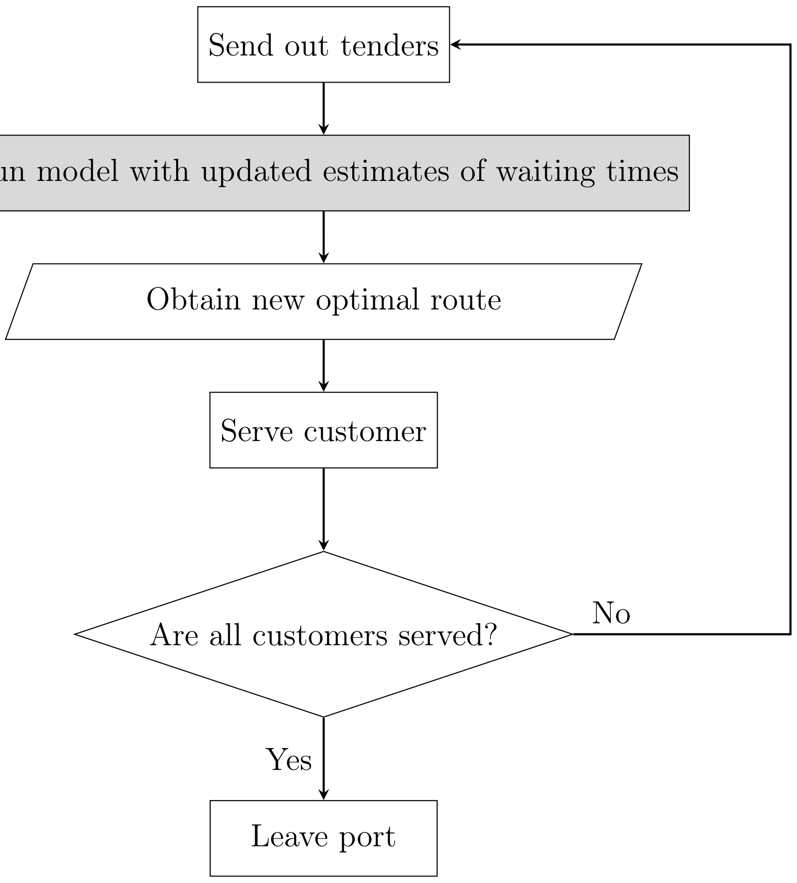

Send out tenders
Run model with updated estimates of waiting times
Obtain new optimal route
Serve customer
Are all customers served?
No
Yes
Leave port

Figure 8.1: A flowchart for how the models can be used in a practical context

If the case company generates the route before entering the port, the routing can be seen as a tactical decision where the route is predetermined for the next few weeks. If the ship instead updates its route after finishing at a terminal, the routing decision can be seen as an operational decision where the route is determined from day to day. The decision regarding how often the model should be run influences the case company's requirements for run time.

The heuristic provided in this thesis may be used in a practical context. It meets the requirements regarding run times also when tank allocation is included, and is able to find solutions for instances of realistic sizes. Some more aspects can be modeled to even better describe the factual situation, probably affecting the solutions and run times. How many aspects that should be included must be decided on the basis of how extensive the job of collecting the input data is, by how the run times are influenced, and by the added value provided by including them.

Chapter 9

Concluding remarks

In this thesis, we have formulated two mathematical models that could be used for developing a decision support tool for a chemical shipping company facing a routing problem in a particular port. The models consider port operations conducted by a ship in the port in order to pick up and deliver cargoes while complying with restrictions regarding time windows, draft limits, and tank allocation. The objective is to minimize the total time used in port. Chemical tankers spend up to 40% of their time in port, despite the fact that the trade routes are all over the world, so improving port efficiency can greatly influence the companies' performance. The two models have been developed with the goal of improving port efficiency, by providing a routing plan taking enough aspects into account to make them useful in a practical context.

The first model presented in this thesis is a pickup and delivery problem considering time windows and draft limits (PDPTW-DL). To our knowledge a pickup and delivery problem with draft limit has not been previously explicitly studied. The second model is a pickup and delivery problem considering time windows, draft limits and tank allocation (PDPTW-DLTA), which better represents the factual situation faced by a shipping company. This model is more complex than the first presented, imposing higher requirements for solution methods.

Both problems have been solved using two different solution methods. The PDPTW-DL was solved with a commercial MIP-solver and with a specialized DP-algorithm. The PDPTW-DLTA was solved with a MIP-solver and with a constructed heuristic. For the PDPTW-DLTA, the MIP-solver finds optimal routes and a feasible tank allocation simultaneously. The heuristic consists of a DP-algorithm finding routes and a MIP-solver solving the tank allocation problem (TAP) for the routes provided by the DP-algorithm.

9.1 Conclusion

The objective of the problem is to find a route serving a given set of customers within a port such that the time spent in port is minimized. The two developed models were tested for 40 and 58 instances, respectively, each using two different solution methods. The results from this testing are evidently favoring the DP-algorithm when solving the PDPTW-DL and the heuristic when solving the PDPTW-DLTA.

The results clearly showed that both the PDPTW-DL and the PDPTW-DLTA are impossible to solve within a reasonable time for instances of realistic sizes when the MIP-solver is used directly. The run times are higher for the PDPTW-DLTA, because the complexity increases when tank allocation is included. The current versions of the models can be considered useless in a practical context for realistic instances when this solution method is used.

When the DP-algorithm was used to solve the PDPTW-DL and the heuristic was used to solve the PDPTW-DLTA, the run times were adequately short for instances of realistic sizes. The run times did not increase significantly when tank allocation was added. The DP-algorithm performs well, providing routes for solving the TAP as suitable as possible, which is one of the main reasons why the TAP is easily solved. The insignificant increase in run times implies that the model including tank allocation should be used to get a more nuanced and realistic representation, as long as the needed input parameters are available. Some of the routes found when solving the model without tank allocation turn out to be infeasible when tank allocation is included. This emphasizes that the PDPTW-DLTA is most suitable in a practical context, providing solutions more appropriate for the intended use.

It is evident that the DP-algorithm and the heuristic perform better and are superior with respect to run time compared to the corresponding methods of using the MIP-solver. The PDPTW-DLTA using the heuristic is preferred over the PDPTW-DL using the DP-algorithm, as it is effectively handling the trade-off between a better reality representation and run times. Using the heuristic, the DP-algorithm may exclude optimal solutions for the PDPTW-DLTA due to the dominance criteria, but this can be seen as a tolerable consequence of reduced run times. However, for all the tested instances in this problem that was solved, the optimal solutions were found. Using the heuristic is concluded to be a well functioning solution method also applicable in a practical context, and further work should mainly consider if the model and its input are close enough to reality.

9.2 Future research

The input parameters related to time are greatly influencing the total time used in port and which routes that are suggested. These parameters may be uncertain, and how they are estimated and predicted can be of importance for the result. How to utilize historical data in a logical and rightful way can be given more attention in further work. More detailed information from chemical tanker companies is needed in order to obtain more realistic approximations. The uncertainty can also be assessed by developing a stochastic model. As the case company is experiencing that its ships spend a large amount of their time in port waiting for the terminals to get ready, the waiting times may be interesting to estimate stochastically. A stochastic model provides solutions more robust with respect to changes of data, and the risk of decisions can be measured and managed (Römisch, 2009).

Several port operations can be modeled more accurately to better represent the reality. The washing of tanks and disposal of residues can be modeled more detailed to better consider the aspects regarding noxious liquids and the implications this might have. Serving of optional cargoes may be modeled in order to include the possibility of undertaking spot loads. To model the possibility of serving several customers at the same time, a super node synonymous with serving a set of cargoes in the same terminal simultaneously can be included. In addition, the time between two customers can be divided into more components than the way it was done in this thesis, also including aspects such as inspection, refueling and terminal leaving time. The stability of the ship can be modeled in a similar manner to what was done by Hvattum et al. (2009). Input data with incompatibilities between cargoes are excluded from this study, but the restrictions dealing with this are included. This aspect can therefore easily be added in the input data to make the problem more realistic. Hence, there are multiple aspects possible to embed in the models to make them a more precise reality representation, but an inclusion must be weighed against a presumed increase in run times and the extra needed input data.

The heuristic can in some ways be further developed and improved. One possibility is to change how it is executed, by e.g. solving the tank allocation problem during the DP-algorithm. For every extension that is done, the TAP can be solved to examine if there exists a feasible tank allocation. If an extension of a path is done such that the tank allocation for that path is infeasible, the label representing this path can be dominated and not further extended. How this could be implemented, and how much the run times are affected by this, could be interesting to examine.

Another approach that can be considered is the cost perspective of the problem. The objective function can be changed to contain cost elements incurred when a route is sailed. By including cost in the model, and allowing soft time windows, it is possible to consider aspects such as demurrage rates, port costs, sailing costs, and the revenues generated by taking spot loads. This was not done in this thesis because the time aspect was considered the most important issue. The demurrage rate is traditionally not a cost the shipping company wishes to take into account when planning, because it would involve speculation of whether the customer is able to accommodate the ship within the promised laydays or not. Sailing costs are considered insignificant within a port. The most significant cost parameter is probably the possible gained revenue from taking spot loads. This possibility was excluded in our model, but could be interesting to investigate in future research. Then, the time charter equivalent could be used as the objective function, taking both voyage earnings and voyage duration into consideration. This might be a good measure for a company's performance.

Bibliography

J. Amdahl, S. Berge, F. Dukan, A. Endal, J. Hals, H. Holm, G. Johnsen, T. King, C. Larsen, L. Lundby, T. Moan, D. Myrhaug, J. Odland, B. Pettersen, S. Steen, A. Sørensen, P. Werenskiold, and W. Æsøy. Ocean space technology - an ocean of opportunities. 2014.

S. Anily and J. Bramel. Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries. *Naval Research Logistics*, 46(6):654–670, 1999.

E. K. Baker. An exact algorithm for the time-constrained traveling salesman problem. *Operations Research*, 31(5):938–945, 1983.

G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: A classification scheme and survey. *TOP*, 15(1):1–31, 2007.

G. Brønmo, M. Christiansen, and B. Nygreen. Ship routing and scheduling with flexible cargo sizes. *Journal of the Operational Research Society*, 58(9):1167–1177, 2006.

G. Brønmo, M. Christiansen, K. Fagerholt, and B. Nygreen. A multi-start local search heuristic for ship scheduling—a computational study. *Computers and Operations Research*, 34(3):900–917, 2007.

M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Maritime transportation. In C. Bernhart and G. Laporte, editors, *Handbooks in operations research and management science*, volume 14, book section 4, pages 189–284. 2007.

M. Christiansen, K. Fagerholt, T. Flatberg, Ø. Haugen, O. Kloster, and E. H. Lund. Maritime inventory routing with multiple products: A case study from the cement industry. *European Journal of Operational Research*, 208(1):86–94, 2011.

M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Ship routing and scheduling in the new millennium. *European Journal of Operational Research*, 228(3):467–483, 2013.

ISL Publications and Databases, 2012. URL `www.infoline.isl.org/index.php?module=Downloads&func=prep_hand_out&lid=820`. Accessed: 2015-04-06.

P. Deseck. Charterparties, 2012. URL `http://www.maritimeknowhow.com/home/chartering-and-charterparties/chartering-and-charterparty-online`. Accessed: 2015-03-03.

M. Desrochers, J. K. Lenstra, and M. W. P. Savelsbergh. A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research*, 46(3):322–332, 1990.

Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54(1):7–22, 1991.

Y. Dumas, J. Desrosiers, E. Gelinas, and M. M. Solomon. An optimal algorithm for the traveling salesman problem with time windows. *Operations Research*, 43(2):367–371, 1995.

K. Fagerholt. Ship scheduling with soft time windows: An optimisation based approach. *European Journal of Operational Research*, 131(3):559–571, 2001.

K. Fagerholt and M. Christiansen. A travelling salesman problem with allocation, time window and precedence constraints — an application to ship scheduling. *International Transactions in Operational Research*, 7(3):231–244, 2000a.

K. Fagerholt and M. Christiansen. A combined ship scheduling and allocation problem. *Journal of the Operational Research Society*, 51(7):834–842, 2000b.

K. Fagerholt, L. M. Hvattum, T-A. V. Johnsen, and J-E. Korsvik. Routing and scheduling in project shipping. *Annals of Operations Research*, 207(1):67–81, 2013.

FrontlineSolvers, 2015. URL `www.solver.com/integer-constraint-programming`. Accessed: 2015-05-20.

X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. *Applied Soft Computing*, 11(4):3680–3689, 2011.

I. Gribkovskaia and G. Laporte. One-to-many-to-one single vehicle pickup and delivery problems. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43, book section 16, pages 359–377. 2008.

I. Gribkovskaia, G. Laporte, and A. Shlopak. A tabu search heuristic for a routing problem arising in servicing of offshore oil and gas platforms. *Journal of the Operational Research Society*, 59(11):1449–1459, 2007.

F. Hennig. *Optimization in maritime transportation: crude oil tanker routing and scheduling*, volume 2010:109. Norges teknisk-naturvitenskapelige universitet, 2010.

H. Hernández-Pérez and J-J. Salazar-González. The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. *European Journal of Operational Research*, 196(3):987–995, 2009.

L. M. Hvattum, K. Fagerholt, and V. A. Armentano. Tank allocation problems in maritime bulk shipping. *Computers and Operations Research*, 36(11):3051–3060, 2009.

International Marine, 2015. URL `http://www.international-marine.com/CargoTanks/Pages/Cargo-Tank-Coatings.aspx`. Accessed: 2015-03-03.

International Maritime Organization, 2015. URL `http://www.ntnu.no/ub/etids/IMO/index.php`. Accessed: 2015-02-03.

S. Irnich and G. Desaulniers. *Shortest path problems with resource constraints*. Springer, 2005.

A. S. Jetlund and I. A. Karimi. Improving the logistics of multi-compartment chemical tankers. *Computers and Chemical Engineering*, 28(8):1267–1283, 2004.

J. E. Korsvik, K. Fagerholt, and G. Laporte. A tabu search heuristic for ship routing and scheduling. *Journal of the Operational Research Society*, 61(4):594–603, 2010.

G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

S. A. Lawrence. *International sea transport: the years ahead*. Lexington Books Lexington, Mass, 1972.

Q. Lu and M.M Dessouky. A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research*, 175(2):672 – 687, 2006.

Milbros, 2015. URL `www.milbros.com`. Accessed: 2015-05-10.

N. Mladenović, D. Urošević, S. Hanafi, and A. Ilić. A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. *European Journal of Operational Research*, 220(1):270–285, 2012.

G. Mosheiov. The travelling salesman problem with pick-up and delivery. *European Journal of Operational Research*, 79(2):299–310, 1994.

B. Nygreen, M. Christiansen, K. Haugen, T. Bjørkvoll, and Ø. Kristiansen. Modeling norwegian petroleum production and transportation. *Annals of Operations Research*, 82(0):251–268, 1998.

Odfjell, 2012. URL `https://www.fonasba.com/wp-content/uploads/2012/10/Chemical-Market-Report.pdf`. Accessed: 2015-02-03.

Odfjell. Annual report 2013. 2013.

Odfjell, 2015. URL `http://www.odfjell.com/Terminals/Houston/Pages/default.aspx`. Accessed: 2015-03-04.

H. N. Psaraftis. A multi-commodity, capacitated pickup and delivery problem: The single and two-vehicle cases. *European Journal of Operational Research*, 215(3):572–580, 2011.

J. Rakke, M. Christiansen, K. Fagerholt, and G. Laporte. The traveling salesman problem with draft limits. *Computers and Operations Research*, 39(9):2161–2167, 2012.

D. Ronen. Cargo ships routing and scheduling: Survey of models and problems. *European Journal of Operational Research*, 12(2):119–126, 1983.

S. Ropke and J.-F. Cordeau. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, 43(3):267–286, 2009.

W. Römisch, 2009. URL `http://www.mathematik.hu-berlin.de/~romisch/papers/RoemischCSM09.pdf`. Accessed: 2015-05-30.

T. R. Sexton and Y-M. Choi. Pickup and delivery of partial loads with "soft" time windows. *American Journal of Mathematical and Management Sciences*, 6(3-4):369–398, 1986.

J.-H. Song and Kevin C. Furman. A maritime inventory routing problem: Practical approach. *Computers and Operations Research*, 40(3):657–665, 2013.

Texas Mooring Inc., 2015. URL `http://texmoor.com/portchart.asp`. Accessed: 2015-03-14.

UNCTAD. Review of maritime transport, 2014. *Review of Maritime Transport*, 2014.

P. Østensjø. A competitive norway: Chemicals shipping. *NSF-report, Centre for Research in Economics and Business Administration, Bergen: NHH*, 1992.

B. O. Øvstebø, L. M. Hvattum, and K. Fagerholt. Routing and scheduling of roro ships with stowage constraints. *Transportation Research Part C: Emerging Technologies*, 19 (6):1225–1242, 2011a.

B. O. Øvstebø, L. M. Hvattum, and K. Fagerholt. Optimization of stowage plans for roro ships. *Computers and Operations Research*, 38(10):1425–1434, 2011b.

Appendix A

Model formulations

A.1 PDPTW-DL formulation

Sets and indices

N	set of nodes, indexed by i and j
N^C	set of all cargoes, indexed by i and j
N^+	set of pickup cargoes, indexed by i and j
N^-	set of delivery cargoes, indexed by i and j
A	set of feasible arcs between i and j

Parameters

Q^+	the total load that is going to be picked up
Q^-	the total load that is going to be delivered
Q_i	the weight of cargo i
K	the total weight capacity of the ship
D_i	the draft limit, converted to tonnes, in the terminal associated with cargo i
T_{ij}	the time from the start of loading/unloading cargo i to the start of loading/unloading cargo j
$\underline{T_i}$	the earliest time for start of loading/unloading cargo i
$\overline{T_i}$	the latest time for start of loading/unloading cargo i

Variables

x_{ij}	1 if the ship sails directly from node i to j, 0 otherwise
y_{ij}	the total weight on board the ship on arc (i,j)
t_i	the time at which the ship starts to load/unload cargo i

Objective function

$$minimize \quad t_{n+1}$$

Flow constraints

$$\sum_{j\in N} x_{0j} = 1,$$

$$\sum_{j\in N|(i,j)\in A} x_{ij} = 1, \qquad i \in N^C,$$

$$\sum_{i\in N|(i,j)\in A} x_{ij} = 1, \qquad j \in N^C,$$

$$\sum_{i\in N} x_{i,n+1} = 1,$$

Cargo constraints

$$\sum_{j\in N^C} y_{0j} = Q^-,$$

$$\sum_{(i,j)\in A} y_{ij} - \sum_{(j,i)\in A} y_{ji} = -Q_j, \qquad j \in N^C,$$

Capacity constraints

$$
\begin{aligned}
0 \leq y_{0j} \leq min\{D_0, K\}x_{0j}, \qquad & (0,j) \in A \mid j \in N \setminus \{0\},\\
0 \leq y_{i,n+1} \leq min\{D_{n+1}, K\}x_{i,n+1}, \qquad & (i,n+1) \in A \mid i \in N \setminus \{n+1\},\\
Q_i x_{ij} \leq y_{ij} \leq (K - Q_j)x_{ij}, \qquad & (i,j) \in A \mid i,j \in N^+,\\
(Q_i - Q_j)x_{ij} \leq y_{ij} \leq K x_{ij}, \qquad & (i,j) \in A \mid i \in N^+, j \in N^-,\\
-Q_j x_{ij} \leq y_{ij} \leq (K + Q_i)x_{ij}, \qquad & (i,j) \in A \mid i,j \in N^-,\\
0 \leq y_{ij} \leq (K - Q_j)x_{ij}, \qquad & (i,j) \in A \mid i \in N^-, j \in N^+,
\end{aligned}
$$

Draft limit constraints

$$
\begin{aligned}
0 \leq y_{ij} \leq D_j\, x_{ij}, \qquad & (i,j) \in A \mid j \in N^-,\\
0 \leq y_{ij} \leq D_i\, x_{ij}, \qquad & (i,j) \in A \mid i \in N^+,
\end{aligned}
$$

Time constraints

$$
\begin{aligned}
t_i + T_{ij} - t_j - M_{ij}(1 - x_{ij}) \leq 0, \qquad & (i,j) \in A,\\
t_i - t_0 \geq 0, \qquad & i \in N \setminus \{0\},\\
t_{n+1} - t_i \geq 0, \qquad & i \in N \setminus \{n+1\},\\
\underline{T_i} \leq t_i \leq \overline{T_i}, \qquad & i \in N,
\end{aligned}
$$

Subtour elimination constraints

$$
x_{ij} + x_{ji} \leq 1, \qquad (i,j) \in A,
$$

Integer constraints

$$
x_{ij} \in \{0,1\}, \qquad (i,j) \in A.
$$

A.2 PDPTW-DLTA formulation

Sets and indices

N	set of nodes, indexed by i and j
N^C	set of all cargoes, indexed by i and j
N^+	set of pickup cargoes, indexed by i and j
N^-	set of delivery cargoes, indexed by i and j
A	set of feasible arcs between i and j
F	set of tanks, indexed by f
N_i^N	set of cargoes in conflict with cargo i, indexed by j
N_f^F	set of cargoes compatible with tank f, indexed by i
F_i^N	set of tanks compatible with cargo i, indexed by f
F_{ijf}	set of tanks that cannot be used for cargo j if cargo i is present in tank t, indexed by e
k	sequence number

Parameters

Q^+	the total load that is going to be picked up
Q^-	the total load that is going to be delivered
Q_i	the weight of cargo i
K	the total weight capacity of the ship
D_i	the draft limit, converted to tonnes, in the terminal associated with cargo i
T_{ij}	the time from the start of loading/unloading in node i to the start of loading/unloading in node j
$\underline{T_i}$	the earliest time for start of loading/unloading cargo i
$\overline{T_i}$	the latest time for start of loading/unloading cargo i
W_{if}	1 if tank f initially are containing a cargo $i \in N^C$ when leaving anchorage
V_i	the quantity (volume) of cargo i
V_{if}^I	the initial volume of cargo i at tank f when leaving anchorage
$\overline{K_f^F}$	the maximum volume capacity of tank f
$\underline{K_f^F}$	the minimum volume capacity of tank f

Variables

x_{ij}	1 if the ship sails directly from node i to j, 0 otherwise
y_{ij}	the total weight on board the ship on arc (i, j)
t_i	the time at which the ship starts to load/unload cargo i
z_{ik}	1 if node i is visited as number k in the sequence of nodes visited, 0 otherwise
w_{ifk}	1 if cargo i is allocated to tank f in sequence number k, 0 otherwise
l_{ifk}	the volume quantity of cargo i at tank f in sequence number k

Objective function

$$minimize \quad t_{n+1}$$

Flow constraints

$$\sum_{j \in N} x_{0j} = 1,$$

$$\sum_{j \in N|(i,j) \in A} x_{ij} = 1, \qquad i \in N^C,$$

$$\sum_{i \in N|(i,j) \in A} x_{ij} = 1, \qquad j \in N^C,$$

$$\sum_{i \in N} x_{i,n+1} = 1,$$

Cargo constraints

$$\sum_{j \in N^C} y_{0j} = Q^-,$$

$$\sum_{(i,j) \in A} y_{ij} - \sum_{(j,i) \in A} y_{ji} = -Q_j, \qquad j \in N^C,$$

Draft limit constraints

$$
\begin{aligned}
&0 \le y_{ij} \le D_j\, x_{ij}, && (i,j) \in A \mid j \in N^-,\\
&0 \le y_{ij} \le D_i\, x_{ij}, && (i,j) \in A \mid i \in N^+,
\end{aligned}
$$

Time constraints

$$
\begin{aligned}
t_i + T_{ij} - t_j - M_{ij}(1 - x_{ij}) &\le 0, && (i,j) \in A,\\
t_i - t_0 &\ge 0, && i \in N \setminus \{0\},\\
t_{n+1} - t_i &\ge 0, && i \in N \setminus \{n+1\},\\
\underline{T_i} \le t_i &\le \overline{T_i}, && i \in N,
\end{aligned}
$$

Subtour elimination constraints

$$
x_{ij} + x_{ji} \le 1, \qquad (i,j) \in A,
$$

Sequencing constraints

$$
\begin{aligned}
z_{00} &= 1,\\
z_{j1} &= x_{0j}, && j \in N \mid (0,j) \in A,\\
\sum_{k \in N} z_{ik} &= 1, && i \in N,\\
\sum_{i \in N} z_{ik} &= 1, && k \in N,\\
z_{jk} - z_{i,k-1} - x_{ij} &\ge -1, && (i,j) \in A, k \in N \setminus \{0\},
\end{aligned}
$$

Allocating and loading constraints

$$
\begin{aligned}
w_{if0} &= W_{if}, && i \in N^-, f \in F, \\
l_{if0} &= V^I_{if}, && i \in N^-, f \in F, \\
w_{ifk} - w_{if,k-1} + z_{ik} &\geq 0, && i \in N^-, f \in F, k \in N \setminus \{0\}, \\
w_{ifk} + \sum_{q=0}^{k} z_{iq} &\leq 1, && i \in N^-, f \in F, k \in N \setminus \{0\}, \\
l_{ifk} - l_{if,k-1} + V^I_{if}(1 - w_{ifk}) &\geq 0, && i \in N^-, f \in F, k \in N, \\
l_{ifk} - V^I_{if} w_{ifk} &\leq 0, && i \in N^-, f \in F, k \in N, \\
w_{ifk} - w_{if,k-1} &\geq 0, && i \in N^+, f \in F, k \in N \setminus \{0\}, \\
l_{ifk} - l_{if,k-1} &\geq 0, && i \in N^+, f \in F, k \in N \setminus \{0\}, \\
\sum_{q=0}^{k-1} w_{ifq} + z_{ik} &\leq 1, && i \in N^+, f \in F^N_i, k \in N \setminus \{0\}, \\
\sum_{f \in F^N_i} \overline{K^F_f} w_{ifk} - V_i z_{ik} &\geq 0, && i \in N^+, k \in N, \\
\sum_{f \in F^N_i} l_{ifk} - V_i z_{ik} &= 0, && i \in N^+, k \in N, \\
\underline{K^F_f} w_{ifk} \leq l_{ifk} &\leq \overline{K^F_f} w_{ifk}, && i \in N^+, f \in F, k \in N, \\
y_{0j} &\leq Q^- x_{0j}, && (0, j) \in A \mid j \in N^C, \\
y_{i,n+1} &\leq Q^+ x_{i,n+1}, && (i, n+1) \in A \mid i \in N^C, \\
y_{i,j} &\leq (Q^+ + Q^-) x_{ij}, && (i, j) \in A \mid i, j \in N^C, \\
\sum_{f \in F} l_{if,n+1} &= V_i, && i \in N^+, \\
\sum_{f \in F} w_{ifn} &= \sum_{f \in F} w_{if,n+1}, && i \in N^+,
\end{aligned}
$$

Compatibility constraints

$$
\begin{aligned}
\sum_{i \in N^F_f} w_{ifk} &\leq 1, && f \in F, k \in N, \\
\sum_{j \in N^N_i} \sum_{e \in F_{ijf}} w_{jek} - |F|(1 - w_{ifk}) &\leq 0, && i \in N^C, f \in F^N_i, k \in N,
\end{aligned}
$$

Integer and convexity constraints

$$\begin{aligned}
x_{ij} &\in \{0,1\}, && (i,j) \in A,\\
z_{ik} &\in \{0,1\}, && i \in N, k \in N,\\
w_{ifk} &\in \{0,1\}, && i \in N, f \in F, k \in N,\\
l_{ifk} &\geq 0, && i \in N^{+}, f \in F, k \in N.
\end{aligned}$$

A.3 TAP formulation

Sets and indices

N	set of nodes, indexed by i and j
N^C	set of all cargoes, indexed by i and j
N^+	set of pickup cargoes, indexed by i and j
N^-	set of delivery cargoes, indexed by i and j
A	set of feasible arcs between i and j
F	set of tanks, indexed by f
N_i^N	set of cargoes in conflict with cargo i, indexed by j
N_f^F	set of cargoes compatible with tank f, indexed by i
F_i^N	set of tanks compatible with cargo i, indexed by f
F_{ijf}	set of tanks that cannot be used for cargo j if cargo i is present in tank t, indexed by e
k	sequence number

Parameters

Z_{ik}	1 if node i is visited as number k in the sequence of nodes visited, 0 otherwise
Q^+	the total load that is going to be picked up
Q^-	the total load that is going to be delivered
W_{if}	1 if tank f initially are containing a cargo $i \in N^C$ when leaving anchorage
V_i	the quantity (volume) of cargo i
V_{if}^I	the initial volume of cargo i at tank f when leaving anchorage
$\overline{K_f^F}$	the maximum volume capacity of tank f
$\underline{K_f^F}$	the minimum volume capacity of tank f

Variables

w_{ifk}	1 if cargo i is allocated to tank f in sequence number k, 0 otherwise
l_{ifk}	the volume quantity of cargo i at tank f in sequence number k

Objective function

$$minimize \quad 0$$

Allocating and loading constraints

$$
\begin{aligned}
w_{if0} &= W_{if}, && i \in N^-, f \in F, \\
l_{if0} &= V^I_{if}, && i \in N^-, f \in F, \\
w_{ifk} - w_{if,k-1} + Z_{ik} &\geq 0, && i \in N^-, f \in F, k \in N \setminus \{0\}, \\
w_{ifk} + \sum_{q=0}^{k} Z_{iq} &\leq 1, && i \in N^-, f \in F, k \in N \setminus \{0\}, \\
l_{ifk} - l_{if,k-1} + V^I_{if}(1 - w_{ifk}) &\geq 0, && i \in N^-, f \in F, k \in N, \\
l_{ifk} - V^I_{if} w_{ifk} &\leq 0, && i \in N^-, f \in F, k \in N, \\
w_{ifk} - w_{if,k-1} &\geq 0, && i \in N^+, f \in F, k \in N \setminus \{0\}, \\
l_{ifk} - l_{if,k-1} &\geq 0, && i \in N^+, f \in F, k \in N \setminus \{0\}, \\
\sum_{q=0}^{k-1} w_{ifq} + Z_{ik} &\leq 1, && i \in N^+, f \in F^N_i, k \in N \setminus \{0\}, \\
\sum_{f \in F^N_i} \overline{K^F_f} w_{ifk} - V_i Z_{ik} &\geq 0, && i \in N^+, k \in N, \\
\sum_{f \in F^N_i} l_{ifk} - V_i Z_{ik} &= 0, && i \in N^+, k \in N, \\
\underline{K^F_f} w_{ifk} \leq l_{ifk} &\leq \overline{K^F_f} w_{ifk}, && i \in N^+, f \in F, k \in N, \\
y_{0j} &\leq Q^- x_{0j}, && (0, j) \in A \mid j \in N^C, \\
y_{i,n+1} &\leq Q^+ x_{i,n+1}, && (i, n+1) \in A \mid i \in N^C, \\
y_{i,j} &\leq (Q^+ + Q^-) x_{ij}, && (i, j) \in A \mid i, j \in N^C, \\
\sum_{f \in F} l_{if,n+1} &= V_i, && i \in N^+, \\
\sum_{f \in F} w_{ifn} &= \sum_{f \in F} w_{if,n+1}, && i \in N^+,
\end{aligned}
$$

Compatibility constraints

$$\sum_{i \in N_f^F} w_{ifk} \leq 1, \qquad f \in F, k \in N,$$

$$\sum_{j \in N_i^N} \sum_{e \in F_{ijf}} w_{jek} - |F|(1 - w_{ifk}) \leq 0, \qquad i \in N^C, f \in F_i^N, k \in N,$$

Integer and convexity constraints

$$w_{ifk} \in \{0, 1\}, \qquad i \in N, f \in F, k \in N,$$

$$l_{ifk} \geq 0, \qquad i \in N^+, f \in F, k \in N.$$

Appendix B

Real case

As introduced in the computational study, a real-case ship voyage provided by the case company was analyzed. An instance was made with information provided to resemble the real case and solved using the heuristic. The output from the heuristic was compared to the route actually sailed. There are aspects about the way the instance parameters are estimated that make this comparison troublesome. The approach used when making this instance, as well as the outcome of the testing are described below.

The information provided about this case was descriptions and time points about most of the port operations done in a port visit in Galveston Bay, Houston, the port examined in this thesis. The case data was given as printouts from an operating system called OTIS, excel sheets with the port log of all port operations done in the bay, including detailed information about cargoes and terminals and a document containing ship information. Because of some missing information we had to make some assumptions. This information was structured the same way as the other information, as in Section 7.1, to be readable for the developed models.

The ship used on this journey was ship Bow Faith, presented in Table 7.4. It was arriving from South America, and had already called several ports. The ship was empty when arriving in Houston, the port that we have been examined specifically. During the port call, four terminals were visited and nine cargoes were handled. An overview of the cargoes is given in Table B.1.

All of the cargoes were pickup cargoes. From the product names of the cargoes, we were able to find associated densities from diverse web pages, as Milbros (2015). Terminal A to D corresponds to the terminals Shell Deer Park, Westway, Stolthaven and Odfjell

Terminals, respectively. As seen in the table, some cargoes are compatible only with stainless steel tanks, type 1, and some are compatible with both stainless steel and zinc. All cargoes in this instance are thus compatible with stainless steel.

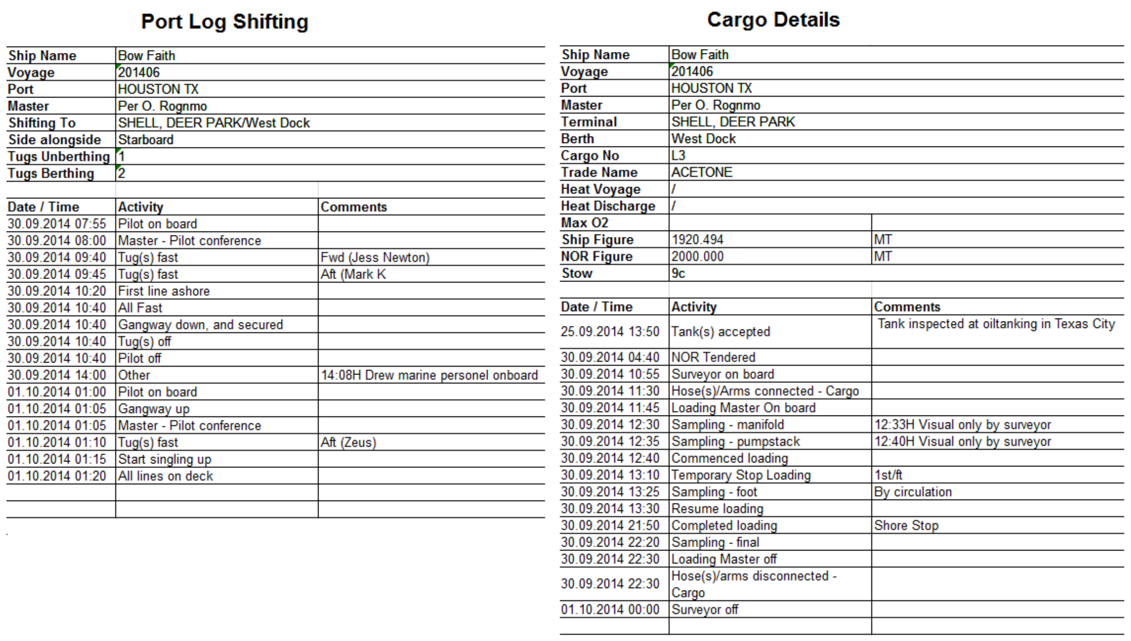
Cargo number	**Terminal**	**Cargo name**	**Cargo load [t]**	**Loading time [hrs]**	**Density [kg/m^3]**	**Cargo Type**
1	A	Acetone	1920.5	10.2	791.0	3
2	B	HP 2	1020.4	18.7	830.0	3
3	B	Nytro 11 Gbx US Transformer Oil	1738.2	4.2	882.0	3
4	C	Brakefluid 310	153.5	19.7	1030.0	1
5	C	Toluene diisocyanate (TDI)	2451.6	21.1	1214.0	1
6	C	Nonylphenol poly (4+)ethoxylate	270.0	22.2	953.0	3
7	D	Acetic anhydride	243.1	14.6	1080.0	1
8	D	DNC 701.01 Developmental polyol	685.0	15.1	1000.0	3
9	D	Ethylenediamine	208.5	15.1	899.0	3

Table B.1: Cargo information for the real case

There was no information in the data provided about the time windows for when the cargoes could be served, but as told by the case company, a logical time window could be approximately between 10 and 14 days. Based on this, we choose the lower time window limits to be 0 and the upper time window limits to be randomly chosen between 10 and 14 days.

Because the terminal names were given, we were able to find their geographical locations and approximate the distances between the different terminals using Google Distance Calculator. Anchorage was assumed to be located the same place as the first terminal in the bay, terminal D, rather than at sea outside the port. This was done because the sailing times between what was called "arrival" to the first terminal and between the last terminal and "departure", were fairly short. The terminal draft limits were found from terminal information from publicly available web sites, and was calculated the same way as described in Section 7.1.3. Because the ship was empty when arriving the area and the cargoes were relatively small, the draft limits were not restrictive in this case.

Figure B.1 shows an example of a log of a terminal shift, to the terminal Shell, Deer Park, and the cargo handling of cargo 6. As can be seen from the figure, all port operations are precisely logged. This was used to calculate entering times, waiting times and loading times. The times used in this instance are found in retrospect, i.e. we are using the entering, waiting and loading times that actually occurred. In reality these must be estimated based on either historical data or estimations from the customers.



Port Log Shifting

Ship Name	Bow Faith	
Voyage	201406	
Port	HOUSTON TX	
Master	Per O. Rognmo	
Shifting To	SHELL, DEER PARK/West Dock	
Side alongside	Starboard	
Tugs Unberthing	1	
Tugs Berthing	2	
Date / Time	**Activity**	**Comments**
30.09.2014 07:55	Pilot on board	
30.09.2014 08:00	Master - Pilot conference	
30.09.2014 09:40	Tug(s) fast	Fwd (Jess Newton)
30.09.2014 09:45	Tug(s) fast	Aft (Mark K
30.09.2014 10:20	First line ashore	
30.09.2014 10:40	All Fast	
30.09.2014 10:40	Gangway down, and secured	
30.09.2014 10:40	Tug(s) off	
30.09.2014 10:40	Pilot off	
30.09.2014 14:00	Other	14:08H Drew marine personel onboard
01.10.2014 01:00	Pilot on board	
01.10.2014 01:05	Gangway up	
01.10.2014 01:05	Master - Pilot conference	
01.10.2014 01:10	Tug(s) fast	Aft (Zeus)
01.10.2014 01:15	Start singling up	
01.10.2014 01:20	All lines on deck	

Cargo Details

Ship Name	Bow Faith	
Voyage	201406	
Port	HOUSTON TX	
Master	Per O. Rognmo	
Terminal	SHELL, DEER PARK	
Berth	West Dock	
Cargo No	L3	
Trade Name	ACETONE	
Heat Voyage	/	
Heat Discharge	/	
Max O2		
Ship Figure	1920.494	MT
NOR Figure	2000.000	MT
Stow	9c	
Date / Time	**Activity**	**Comments**
25.09.2014 13:50	Tank(s) accepted	Tank inspected at oiltanking in Texas City
30.09.2014 04:40	NOR Tendered	
30.09.2014 10:55	Surveyor on board	
30.09.2014 11:30	Hose(s)/Arms connected - Cargo	
30.09.2014 11:45	Loading Master On board	
30.09.2014 12:30	Sampling - manifold	12:33H Visual only by surveyor
30.09.2014 12:35	Sampling - pumpstack	12:40H Visual only by surveyor
30.09.2014 12:40	Commenced loading	
30.09.2014 13:10	Temporary Stop Loading	1st/ft
30.09.2014 13:25	Sampling - foot	By circulation
30.09.2014 13:30	Resume loading	
30.09.2014 21:50	Completed loading	Shore Stop
30.09.2014 22:20	Sampling - final	
30.09.2014 22:30	Loading Master off	
30.09.2014 22:30	Hose(s)/arms disconnected - Cargo	
01.10.2014 00:00	Surveyor off	

Figure B.1: Example from port log showing shifting to a terminal and details regarding the handling of a specific cargo

The waiting times are calculated as the difference between "NOR tendered"(notice of readiness) and "Pilot on board", given in cargo details. The time between "Pilot on board" and "Tug(s) fast" corresponds to the actual sailing time that we chose to estimate based on geography and ship speed. The entering time is found in the "Port log shifting" document as the difference between "Tugs fast" and "Surveyor on", because we have defined the usage of tugboat as a part of the entering time. Given in the shifting specifications is the time usage of tugboat also when leaving the terminal. We have not included a leaving time in the model, and choose to simplify the time estimates by ignoring the leaving time.

Cargoes served subsequently in the same terminal have the same logged date/time for "Surveyor off" and "Surveyor on", but with different times for the various port operations in between these times. The loading times for each cargo are assumed to be the time between "Surveyor on board" and "Surveyor off" divided by the number of cargoes served subsequently in that terminal. The loading times include inspection of tanks, which has been ignored in the other instances. We have included it in the loading times in this instance in order to make it as comparable to the actual outcome as possible.

Results

The instance was solved using the heuristic and the result given was to sail a route similar to the one actually sailed. The two paths are shown in Figure B.2 below.

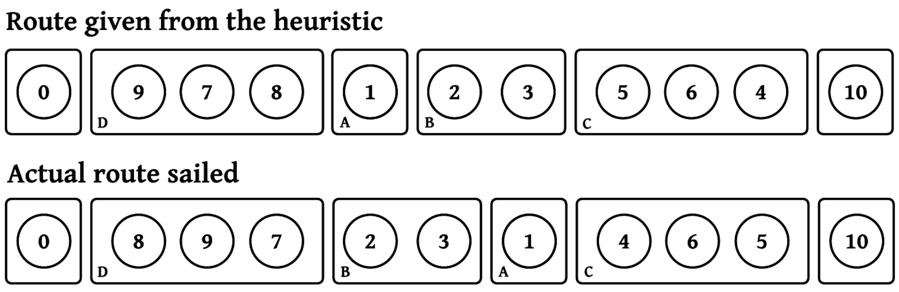


Figure B.2: Illustration of the route chosen by the heuristic and the route actually sailed

The visiting sequence within the different terminals is in this case symmetrical. This is due to the fact that all lower time window limits are set to zero, the upper time window limits are not restricting and that we only have pickup nodes. The sequence within each terminal chosen by the heuristic is to visit the cargoes with the least cargo loads first, in compliance with the preprocessing, described in Section 4.2.2. The difference in the above solutions is therefore the sequence in which the terminals have been visited. In the path provided by the heuristic, the ship visits the terminals in the order D, A, B, C. The actual path sailed visited first terminal D, before B, A, C. The DP-algorithm provides four feasible paths when solving the PDPTW-DL. After solving the TAP, the first path appears feasible and is thus the optimal solution to the problem for this instance, given the chosen input parameters. The route actually sailed is not among the paths provided by the DP-algorithm, because it has been dominated based on higher sailing times. Sailing between the terminals D, B, A, C corresponds to a sailing time of 7.4 hours while sailing between D, A, B, C corresponds to a sailing time of 7.3 hours. So, when the DP-algorithm extends the paths to terminal C, the sequence D, B, A, C is dominated. For this domination to be correct, the sailing times must be true, which does not necessarily apply as they are estimated numbers.

The solution time corresponding to the route found by the heuristic is 108.3 hours, while the actual route used 105.6 hours. These times cannot be directly compared. If the heuristic would have provided the exact same route as the one actually sailed, the times would not be equal. This is because the various port operations are not modeled precisely in all cases, and some assumptions were made. The time used to refuel petroleum, the leaving time out from a terminal and inspection time have not been modeled explicitly, but are incorporated in the other time parameters. The waiting times may be inaccurate because a notice of readiness in some cases was tendered before the ship left the preceding terminal, causing some overlap in the time points. Additionally, the estimation of sailing times may not be completely correct. This is the reason the solution time from the heuristic cannot be compared directly with the time it actually took. But if we compare the solution time given by the heuristic to the solution time the heuristic would have given if the route D, B, A, C, the difference is only 0.1 hours. The model suggest thus a route which potentially is 6 minutes faster than the one actually sailed, without being able to conclude anything because of uncertain input parameters.

If the different times could have been described more precisely, the optimal route could be evaluated in hindsight. The fact that the times are found in retrospect makes the comparison somewhat problematic. When the shipping company made the sailing decisions, it naturally did not know the exact times each port operation would require, but made their decisions based on estimations and earlier experience. It could have been interesting to investigate the output from the model based on the same information the decision makers had at the time the decisions were made, but unfortunately we did not have access to this information.