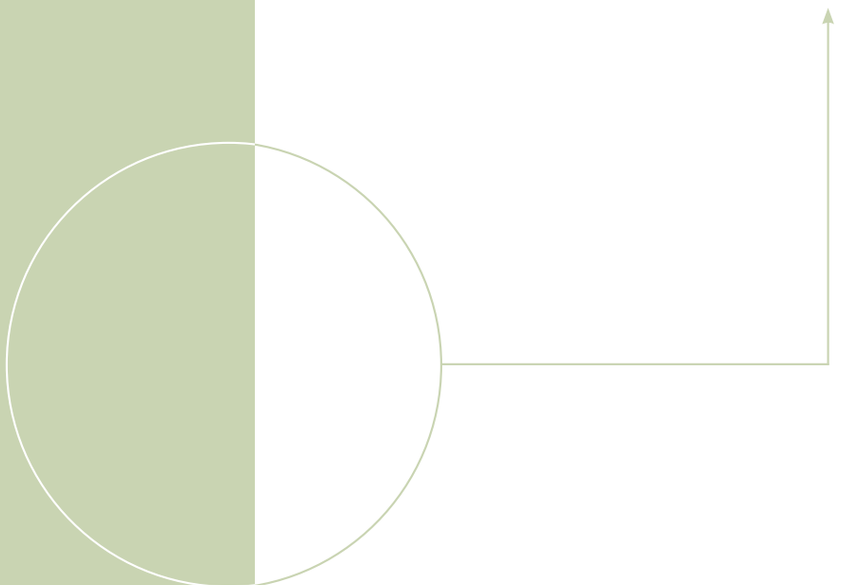
Mohammed S R Alhayek
Bjørn Løvland Manheim
Henrik Klauset Svensson

June 2019

Master's thesis

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

Master's thesis 2019
Mohammed S R Alhayek, Bjørn Løvland Manheim, Henrik Klauset Svensson



NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of
Science and Technology

Mohammed S R Alhayek
Bjørn Løvland Manheim
Henrik Klauset Svensson

Industrial Economics and Technology Management
Submission date: June 2019
Supervisor: Magnus Stålhane

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

Abstract

Natural gas (NG) is expected to play a significant role in the transition toward a lower-carbon energy mix. For long-distance trade, NG is transformed into a liquid state and transported by sea in specially built vessels. Many of the operational challenges associated with liquified natural gas (LNG) transportation can be modeled and solved using mathematical programming. This makes LNG transportation interesting from an operations research perspective. This thesis examines a short-term ship routing and scheduling problem for an LNG producer with a heterogeneous fleet of LNG vessels. This problem aims to create routes and schedules for the vessel fleet such that the producer fulfills a set of long-term contracts and at the same time exploit the opportunities in the spot market.

A review of relevant literature is conducted. The surveyed literature comprises operational LNG planning problems and routing problems that are tightly related to the problem studied in this thesis. The findings indicate that exact solution methods are insufficient to solve similar problems of realistic sizes within a reasonable time. In addition, many of the surveyed problems in the LNG literature are solved by incorporating a mixed integer linear program in a heuristic solution procedure. This is often attributed to complicating side-constraints.

In addition to a mixed integer linear program formulation, two heuristic solution methods for solving the optimization problem are presented. These two methods are referred to as The LNG Adaptive Large Neighborhood Search heuristic (ALNS) and The LNG Fix and Optimize heuristic (FO). The models are applied to realistic problem data sets. Similar to the surveyed problems in the literature review, results show that a MIP is not able to solve the largest and most complex instances within a time frame of 3600 seconds, even with onboard heuristics activated in a commercial MIP solver.

The first solution method, FO, assumes a feasible solution populated with vessel legs at hand. It then searches the solutions space by destroying (removing) some of

the vessel legs before applying a MIP, attempting to repair (rebuild) the solution for a pre-specified amount of time. The second solution method, ALNS, searches the solution space by destroying and repairing the solution many times, while adjusting methods for destroying and repairing in an adaptive manner, attempting to reach new, improving solutions. Further, two different algorithms are developed to construct initial feasible solutions as input to the solution methods. Both solution methods show promising results compared to MIP as they find high-quality solutions within a reasonable time. However, FO outperforms ALNS in terms of consistency in finding high-quality solutions.

Alongside these two solution methods, an event-based simulation program has been developed. The simulation program has been applied to evaluate solutions not only in terms of projected deterministic monetary value but also in terms of statistical robustness. One of the concrete outputs from this simulation program is an approximation of a Pareto frontier between profitability and total arrival time violation of time windows. The frontier is defined by a set of non-dominated high-quality solutions consisting of a trade-off between profitability and robustness. This frontier allows the decision maker to select a solution that fits his risk preferences.

In addition, we propose three robustness strategies to guide the proposed solution methods to find more robust and reliable solutions. The strategies comprise penalizing late arrivals, planning with exaggerated sailing times and buffer quantities. The effect of these strategies is benchmarked against a case base where no strategies are implemented. Test results are promising and indicate a significant value add for a decision maker.

Sammendrag

Naturgass er forventet å spille en viktig rolle i en overgang mot energiproduksjon med lavere klimagassutslipp, som for eksempel CO_2. I forbindelse med langdistanse handel og transport av naturgass, er gassen kjølt kraftig ned slik at den blir til væske, kalt flytende naturgass eller LNG, før væsken blir transportert med spesielt egnede skip. Mange av de operasjonelle utfordringene som oppstår i forbindelse med frakt av LNG kan modelleres og løses ved hjelp av matematisk programmering, nærmere bestemt av et lineært blandet heltallsprogram. Dette gjør LNG transport til et relevant og viktig forskningsområde innen optimering og beslutningstøtte (eng: operations research). Denne masteroppgaven omhandler rute -planlegging og de øvrige beslutningene på kort sikt for en LNG-produsent med en flåte av ulike skip. Kort sikt betyr i denne sammenheng en periode på $\sim$90 dager. Resultatet fra masteroppgaven har som mål å utforme ruter og tidsplaner for flåten av skip innen rimelig tid for relativt store problemstørrelser (>10 skip). Rutene og tidsplanene samt øvrige beslutninger må tilfredsstille en rekke krav, i tillegg til å tilfredsstille grunnprinsipper som å imøtekomme behov fra faste kunder. At fokusområdet er planlegging på kort sikt innebærer mer detaljert modellering samt at potensielle salgsmuligheter i spotmarkedet blir hensyntatt.

En studie som omhandler relevant litteratur har blitt gjennomført. Studien omhandler operasjonelle planleggingsproblemer for LNG, rute-planleggingsproblemer som er tett knyttet til problemet som denne masteroppgaven omhandler. Tidlige studier indikerer at eksakte løsningsmetoder ikke evner å finne gode løsninger på problemet innen tidsbegrensningene som er satt. Videre observeres det at mye av den relevante litteraturen tar utgangspunkt i en heuristisk løsningsmetode kombinert med bruk av et blandet heltallsprogram. Dette er mye på grunn av kompliserende tilleggs-begrensninger som gjør at problemet skiller seg fra tradisjonelle rute-planleggingsproblem for kjøretøy (VRP).

I tillegg til en formulering basert på blandet heltallsproblem metodikk, er det også presentert to heuristiske metoder for å løse optimeringsproblemet. Disse to

metodene refereres til som LNG Adaptive Large Neighborhood Search heuristic (ALNS) og LNG Fix and Optimize heuristic (FO). Modellene er testet på realistiske datasett. Resultatene fra eksakte løsningsmetoder av det blandete heltallsproblemet viser at disse metodene ikke er i stand til å løse de største og mest komplekse problemene innen en tidsramme på 3600 sekunder, dette gjelder også for kommersielle optimeringsprogram med innebygde heuristiske metoder aktivert. Disse resultatene er i tråd med problemene som har blitt forsøkt løst i relatert litteratur.

Den første heuristikken, FO, tar utgangspunkt i en gyldig løsning på optimeringsproblemet. Deretter går metoden ut på å søke etter nye, bedre, løsninger gjennom å ødelegge og reparere løsningsrommet gjentatte ganger. Ødelegging foregår ved at en delmengde av rutevariablene settes fri mens resten av rutevariablene fikseres. Reparasjonsprosedyren foregår ved at et kommersielt optimeringsprogram løser et blandet heltallsproblem der den resterende delmengden av løsningen er fiksert. Den andre heuristikken, ALNS, søker også etter nye løsninger slik som FO ved hjelp av å ødelegge og reparere løsninger gjentatte ganger. Denne heuristikken justerer parametere og hvilke metoder som blir brukt for å ødelegge og reparere adaptivt. Hovedforskjellen fra FO er at denne heuristikken bruker langt flere metoder for å ødelegge og reparere løsninger. Videre så blir to ulike algoritmer for å konstruere en initial-løsning presentert. Begge heuristikkene viser lovende resultater ved sammenlikning opp mot å løse et blandet heltallsproblem, ettersom de ofte finner gode løsninger innen relativt kort tid. FO har dog vist seg å være overlegen sammenliknet med ALNS. Den er mer konsistent, og er i tillegg en langt kraftigere løsningsmetode når den er på sitt beste. Videre så er FO vist å være bedre både når det gjelder å finne løsninger på under 250 sekunder og under 3600 sekunder.

I tillegg til disse to løsningsmetodene, så har et simuleringsprogram blitt utviklet. Simuleringsprogrammet er brukt til å evaluere løsninger, ikke kun ved hjelp av å vurdere lønnsomhet, men også ved å vurdere robusthet. En av de konkrete leveransene fra dette arbeidet er en approksimasjon av en Pareto-front mellom lønnsomhet og total forsinkelse. Denne fronten gjør det mulig for bruker å ta avgjørelser som passer med brukerens risikopreferanser. I tillegg presenteres tre robusthets-strategier. Disse strategiene harsom formål å styre de nevnte løsningsmetodene mot å finne robuste løsninger. De tre strategiene inkluderer: straff ved sen ankomst, planlegging med overdrevne seilingstider samt økte buffer-kvantiteter. Tester viser at disse strategiene fungerer godt og at de gir betydelig verdi for beslutningstaker.

Contents

List of Figures

List of Tables

1. Introduction

As the world demand for energy is higher than ever, natural gas (NG) is expected to play a significant role in fulfilling this demand. Driven by its flexibility, versatility, and abundance as it can be used in many applications like home heating, electricity generation and fuel for trucks and ships, NG is the only fossil fuel to grow its share of the world energy mix in the next two decades (*World Energy Outlook* 2018).

NG is traditionally transported by pipelines from producer to consumer. However, an increase in the number of proven reserves and the long distances between reserves and consuming markets made this alternative unfeasible or uneconomical. The increasing demand for NG and the producer's desire to capitalize on their reserves have led to developing other forms of transporting NG. Transport of NG over long distances is done efficiently by transforming NG into a liquid state which reduces its volume by a factor of 1:600. Today, the share of liquified natural gas in the global NG trade accounts for 45 %. LNG is also expected to capture 90% of the projected growth in long-distance trade of natural gas as it is more economical than pipelines across oceans and over long distances (ibid.).

The high capital-intensity and complexity of the LNG industry have forced suppliers and customers to share risk through long-term contracts with a duration of around 20 years. However, the emergence of new suppliers and markets, overcapacity in the supply side, many new supply projects under construction combined with customers pushing for more flexibility have paved the way for a structural shift. This includes the emergence of the short-term and spot market and a shift toward contractual flexibility in delivery quantities, pricing terms, and destination flexibility.

To satisfy the long-term contracts, an LNG producer usually plans deliveries for 12 months at a time. Such a plan is called the Annual Delivery Program (ADP) and consists of a delivery schedule and vessel routes for one contract year. Although the scheduling and routing process is usually done manually based on pre-specified

rules and industry experience, the increasing complexity of the LNG industry makes manual scheduling prohibitively time consuming and cumbersome. This makes an ADP rigid and inflexible in the face of rapid changes, disruptions, and the emergence of opportunities in the spot market. For this reason, many producers see an opportunity in utilizing decision support techniques to make good decisions quickly.

1.1 Purpose and Contribution

The purpose of this thesis is to investigate and solve a short-term ship routing and scheduling problem for an LNG producer. The aim of this problem is to produce an alternative for ADP when it becomes inefficient in the face of disruptions and rapid changes during a contract year. For this reason, it focuses on a short planning horizon of around three months and incorporates a high level of operational detail. The goal of this problem is to maximize the producer's profit while satisfying the customers' time and quantity requirements in a planning horizon of around three months.

Due to the complexity and large solution space in routing and scheduling problems, exact solution methods are often insufficient to solve this problem within a reasonable time. This necessitates the development of a heuristic solution approach to improve the computational time of solving this operational problem. The main contribution of this master's thesis can be summarized as follows:

- A literature survey on LNG planning problems and tightly related routing problems with a focus on the applied solution methods.
- A mathematical model of the LNG routing and scheduling problem with some inventory elements are included.
- Two different large neighborhood-based approaches capable of solving problem instances with realistic sizes and within a reasonable time.
- A simulation framework used to measure the robustness of solutions.

1.2 Structure of Thesis

The remainder of this thesis begins with a short introduction to the LNG industry in Chapter 2. A formal description of the problem is presented in Chapter 4. A literature review on the LNG delivery problem and relevant routing problems with a focus on the applied solution methods is presented in Chapter 3. We translate

the problem into a mathematical model in Chapter 5. Chapter 6 presents an adaptive large neighborhood search heuristic for solving the aforementioned problem, while Chapter 7 presents a fix and optimize large neighborhood search heuristic for solving the same problem. In addition, three robustness strategies are proposed. Chapter 8 presents a simulation model to evaluate the robustness of solutions. Chapter 9 describes how the input of the solution methods is generated. Furthermore, a computational study on the problem, the proposed solution methods, and simulation model is presented in Chapter 10 before rounding off by concluding this thesis and presenting future research opportunities in Chapter 11.

2. Background

As the world economic output is expected to double by 2040, the global energy demand is likely to grow by 25 percent. This trend will mainly be driven by population growth of two billion people, a growing middle class and improved living standards. This might result in a near doubling in electricity demand in non-OECD nations by 2040 (Exxon Mobil Corporation, 2018).

During the same period, initiatives intended to counteract the issue of global warming is likely to increase the demand for less carbon-intensive sources of energy. Natural gas (NG) is expected to play a major role in the transition toward a lower-carbon energy mix. NG emits 50% less CO_2 than coal and 30% less than oil (National Energy Technology Laboratory, 2013). Additionally, NG is also known for its flexibility, versatility, and abundance as it can be used in many applications like home heating, electricity generation, and fuel for trucks and ships. For these reasons, NG is expected to grow by 40% to increase its share in the world energy mix from 23% to 26% by 2040. (Exxon Mobil Corporation, 2018)

The current global natural gas market has evolved from a collection of local and regional markets with few suppliers and customers. However, an increase in the number of proven reserves and rapid growth in demand in countries far away from these reserves have created an international market for natural gas. This imbalance necessitated the international trade of NG by long-distance pipelines. Also, this led to the transportation of NG by the sea in specialized tankers, where the NG is in liquid form where NG is cooled down to below its boiling point at around -160°C and transported in specialized tankers (Tusiani and Shearer, 2007).

2.1 The LNG Supply Chain

In this section we present a brief introduction to the LNG supply chain.

2.1.1 Supply Chain Structure

The LNG supply chain can be divided into four parts before the end consumer consumes it. These parts are shown in Figure 2.1.

Figure 2.1: Supply chain of LNG. The images are by unknown authors and licensed under Creative Commons

Exploration and Production

The first step in the LNG supply chain is the exploration and development of the gas field. Unlike before, when natural gas was considered a less preferred by-product of oil, exploration today is often aimed directly at the discovery of gas reserves. Natural gas can be found onshore or offshore and can either be produced with the oil from an oil reservoir (associated gas) or come from a gas field (non-associated gas). The non-associated gas is called ”dry” if it contains nearly pure methane, or ”wet” if it contains heavier hydrocarbons such as propane, butane, and condensates. LNG is typically categorized as either “rich” or “lean”, depending on the hydrocarbon content mix, and therefore the calorific value, i.e. the specific energy quantity in the LNG. Lean LNG contains mostly methane with some small quantities of ethane, giving it a lower energy content than rich LNG, which contains more of the heavier hydrocarbons. The type of LNG produced depends on the origin gas/oil field and the liquefaction technique (*Liquefied natural gas: understanding the basic facts* 2005). Since the heating value is different, the price is also different for the two types of LNG. Therefore, it is important for the buyer and seller of LNG to specify the LNG type. In most cases, exploration and production are covered by agreements between the host government and the energy companies (Tusiani and Shearer, 2007).

Liquefaction and Storage

The produced gas is transported by pipeline to the liquefaction plant. To ensure that the gas has desirable combustion and liquefaction properties it has to be treated before liquefaction by removing heavy hydrocarbons and other contaminants. This can either be done at the production facilities or at the liquefaction plant. For wet gas, the extracted liquefied petroleum gasses which consist of heavier hydrocarbons than LNG and the condensate generate additional revenue streams. The liquefaction process cools the natural gas to around -160°C, transforming it to a liquid state, allowing storage and transportation at atmospheric pressure. The LNG is stored in insulated cryogenic tanks designed to keep the LNG cold until it's loaded onto LNG tankers. The plant also contains loading facilities to allow safe access for the LNG tankers (*Liquefied natural gas: understanding the basic facts* 2005).

Transportation

After liquefaction, the LNG is loaded onto LNG vessels. These vessels are specially designed to store the LNG in insulated cargo tanks. The two main types of tanks are spherical tanks (also called Moss tanks), which are self-supporting, and membrane tanks, which use their insulation and the ship for support. A side view of an LNG Moss tanker is shown below.

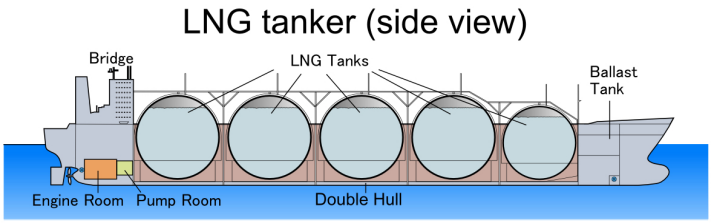


Figure 2.2: Side view of an LNG Moss tanker.[1]

[1] (The picture is under CC-BY-SA-3.0. Source: https://commons.wikimedia.org/)

Because of the challenges of maintaining the LNG at a stable temperature, about 0.15% of the cargo boils off every day (Dobrota et al., 2013). This so-called boil-off gas (BOG) needs to be removed from the tanks. The majority of today's LNG vessels have a cargo capacity in the range 130,000-150,000m^3. These conventional LNG vessels usually use a steam turbine propulsion system or duel-fuel diesel engines, where the boil-off gas can be used as fuel. During the last decade, several

Q-Max and Q-Flex vessels have joined the LNG fleet. Q-Max vessels are the largest ships that can dock at the LNG terminals in Qatar with a cargo capacity of 266,000m^3. Q-Flex vessels have a capacity between 210,000m^3 and 216,000m^3. Both of these vessel types are equipped with an onboard re-liquefaction system to liquefy the boil-off gas and transfer it back to the LNG tanks, to avoid the loss of cargo during operation. The Q-Max and Q-Flex vessels use a single-fuel diesel-mechanical propulsion system, but can be converted to use the boil-off gas as fuel (IGU, 2018).

Loading and unloading rates for the LNG tankers usually vary between $12000m^3$ per hour and $14000m^3$ per hour, depending on the size of the vessel (Tusiani and Shearer, 2007). At the liquefaction plant, the vessels' storage tanks are only loaded to 98% of their capacity, as required by the IMO. The 2% remaining storage capacity prevents liquid from entering into the ventilation pipeline and spilling into the hull structure of the vessels (Dobrota et al., 2013). In most cases, only between 98.5% and 99% of the gas is unloaded at the regasification plant. The gas left in the tanks is called "heel" and is used to keep the tanks cold during the vessel's ballast voyage. For vessels without a re-liquefaction system, the heel must also take the expected boil-off during the return leg into account. It is possible to choose to empty the tanks completely during unloading and let them go warm. This is called doing a "heel out". The vessels are then required to go through a cool-down process at the next loading port before it can start loading. This process is expensive and time-consuming, but it is necessary to minimize the vaporization of LNG and to avoid thermal shocks for the tanks. With today's technology, the cool-down process takes approximately one full day to complete; however, it takes a shorter time if the vessel has membrane tanks instead of spherical tanks since the spherical tanks have larger mass (Tusiani and Shearer, 2007).

Vessels regularly undergo planned maintenance activities at specific times and locations. During the maintenance period, the vessel tanks are emptied completely and warmed for a dry dock. When the vessel is sent back into operation, the cargo tanks are filled up with inert gas or nitrogen for purging. Inert gas contains about 14% carbon-dioxide, which will freeze at around -60 °C and produces a white powder which can block valves, filters, and nozzles, so if inert gas is used, the vessel has to be "gassed up" and cooled down at the next liquefaction terminal before it can start loading. Gassing up involves replacing the CO_2 with LNG vapor (Liquefied Gas Carrier, n.d.). The combined process of gassing up, cool-down and loading can take up to three days (Msakni and Haouari, 2018).

Regasification, Storage and Distribution

The last component of the LNG supply chain is the regasification plant, where the LNG cargoes are unloaded from the vessels, stored and vaporized, before being transferred to the distribution network by pipeline. The plant might also contain facilities for loading the LNG onto tanker trucks for road delivery. The regasification plants are usually owned by the customer, but the customer may in some cases lease capacity on a third-party access basis (Tusiani and Shearer, 2007).

2.1.2 Supply Chain Management Components

In this section, some important aspects of the LNG supply chain management are presented.

Planning Levels

Decision making and planning in the maritime industry can be categorized by different planning levels based on the length of the planning horizon and business impact of the decisions. Traditionally, planning problems can be divided into three classes: Strategic, tactical and operational. The strategic planning level involves decisions that make a long term impact on the business, usually with a planning horizon of 5 to 20 years (Rakke, 2012). A typical strategic planning problem is the determination of the optimal fleet size and mix, which usually involves a decision of reducing or extending an already existing fleet. Other common strategic decisions are contract evaluation, vessel design, and network design. Tactical planning problems usually address medium-term planning decisions, typically with a planning horizon from a few weeks to a couple of years. Common tactical planning problems are ship routing and scheduling, fleet deployment and inventory ship routing. Operational planning tends to only have a short-term impact on the business, for instance only affect one sailing leg and can be used to handle uncertain operational environments. Typical operational decisions in the maritime industry can be ship loading, speed optimization, disruption management, and booking of single orders. It is important to note that the boundaries between the different planning levels are somewhat loosely defined. For instance, many tactical planning decisions like routing and scheduling problems may include operational aspects such as speed optimization.

Market Liquidity

The high capital-intensity and complexity of the LNG industry have forced suppliers and customers to share risk. This is typically ensured through the sale and

purchase agreement (SPA) which is a long-term contract that guarantees LNG supply to the customer over a period of 10-20 years and a minimum level of cash flow to the supplier. These contracts usually have a take-or-pay clause (TOP) which assigns the price risk to the seller and the volume risk to the buyer by obligating the buyer to pay for a percentage of the annual contracted quantity even if the buyer decides not to receive it. Additionally, the SPA usually includes a delivery obligation which requires the seller to deliver a specified amount during the contract period. If the seller fails to fulfill this obligation, the seller has to pay the buyer a penalty cost or any additional costs related to the procurement of alternative fuels (Chandra, 2017). The amount of LNG specified in the SPA is either a monthly demand to be delivered, or a yearly demand, which is to be fairly evenly spread out throughout the year. There is some flexibility with respect to the total amount delivered for a given year; however, it should even out in the long-term.

Although the major part of the LNG is traded through long-term contracts, more liquid forms for buying/trading LNG are emerging. These forms include spot and short-term contracts. The International Group of Liquefied Natural Gas Importers defines short-term contracts as contracts with a duration of four years or less, while pure spot contracts are deliveries that occur less than three months from the transaction date. Deliveries defined as short-term accounted for 27 % for the LNG imports in 2017, while pure spot deliveries accounted for 20 % (GIIGNL, 2018).

Shipping

SPA states the point where the responsibility for LNG transfers from the buyer and the seller. LNG is often sold on free-on-board (FOB) or delivered ex-ship (DES) basis. In FOB shipping, the responsibility for LNG transfers at loading port where the buyer arranges the shipping service. Hence, the sale price does not include transportation costs. In DES shipping, the seller is responsible for delivery and shipping LNG to the destination port. In this case, the sale price includes shipping and insurance costs.

Historically, LNG vessels have been built in relation to a specific project and designed to serve the ports of the supplier and the customer as specified in the SPA. Also, the supplier usually managed these vessels. However, there is a shifting trend among buyers to increase operational and destination flexibility by taking an active role in the LNG shipping (Tusiani and Shearer, 2007). This is done through charter parties or direct ownership in the LNG vessels. The most used charter parties in the LNG industry are time and bareboat charter parties.

In a long-term contract, the supplier and the customer agree on an Annual Delivery

Program (ADP) before each contract year. The ADP specifies the (approximate) delivery dates/time-windows and the corresponding amounts to be delivered. The time-windows often span 1-7 days. The underlying reason for creating an ADP is to satisfy the long-term contracts. This ADP serves as an input to the operational short-term plan, which additionally integrates the opportunities of spot contracts. The ADP setup is a tactical planning problem with a typical planning horizon of 12-18 months. The ADP planning is similar to the short-term planning in the sense that it should provide an optimal fleet schedule, approximate delivery dates or time windows. It should satisfy inventory and/or demand constraints and berth constraints. (Andersson, Christiansen, Desaulniers, and Rakke, 2017)

Costs in shipping

In general, the financial performance of a shipowner depends on the method of financing building/buying the ship, costs of running day-to-day operations, and revenues generated from operating or chartering the ship. Although there is no standard cost classification standards in the shipping industry, it is common to classify ship's costs into five categories:

- Operating costs are also known as OPEX and accounts. These are the day-to-day expenses of running the ship which include costs of stores and lubricants, insurance, crew, and unplanned repairs and maintenance during sailing time.

- Periodic maintenance costs consist of the expenses involved in major upgrading and repairs. This post is usually considerable for old ships.

- Voyage costs, also known as VOYEX, are all the expenses related to a specific voyage. This includes fuel, canal dues, and port charges.

- Capital costs are known as CAPEX and include all the expenses related to financing the investment.

- Cargo-handling costs are all expenses associated with cargo loading, discharging and stowing.

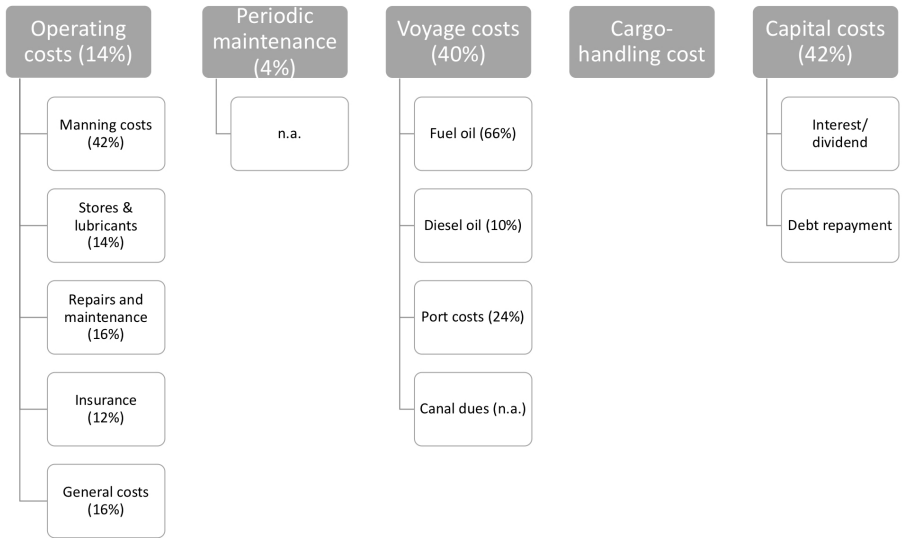


Figure 2.3: Example of costs of a bulk ship. The figure is inspired and based on (Stopford, 2013)

Note that in the operations research literature, operating costs are often used to refer to the variable part of the day-to-day expenses of a ship which include voyage and cargo-handling costs.

Chartering

LNG producers usually charter LNG vessels from each other if they are short for idle vessel capacity. In this case, chartering is a cost for the LNG producer with a lack of capacity and revenue for the producer with idle capacity. According to Stopford (2013), the shipowner's revenues can simply be estimated based on the price received per unit transported (also called freight rate) and the total capacity of the ship. The contract between the shipowner and the charterer (the shipper) is called a charter party. There are three main forms of charter parties:

- A bareboat charter: The charterer enjoys full control of the vessel and is responsible for maintenance, crew, insurance, and everything needed to use the vessel during the contract period. The shipowner gets capital expenses (CAPEX) in return.
- Time charter: The charterer pays a fixed payment per time unit which cov-

ers capital expenses (CAPEX) and operating expenditures (OPEX). On the other hand, the shipowner offers a vessel that is compliant with international conventions and experienced and competent crew. In this case, the charterer takes the market risk since he has to pay the hire regardless of market conditions, while the shipowner takes the operational risk since he does not get paid in case of breakdown.

- Voyage charter: the charterer pays a freight rate per unit transported. The shipowner is responsible for the planning and execution of the voyage and pays all expenses during that voyage. In this case, the shipowner takes the market risk in case of no cargo is available to be transported and the operational risk in case of breakdown.

Although the shipowner's revenues are limited by the ship's capacity and the freight rate that the charterer is willing to pay, many other factors can have a considerable impact on the revenues. These factors include, among others:

- Ship productivity, which concerns the number of loaded days in a financial year
- Deadweight utilization which refers to how much of the full payload is utilized during the loaded days.
- Optimizing operating speed, which determines how much cargo can be transported to customers during a period of time.

2.2 Aspects of Propulsion and Fuel Consumption for LNG Vessels

In this section, we present an overview of the main propulsion systems used in LNG vessels. Finally, some aspects of the relationship between fuel consumption and speed are discussed.

2.2.1 Propulsion Systems

Since the middle of the 2000s, the propulsion systems in LNG tankers have experienced large innovations and improvements in order to better utilize and handle the boil-off gas and reduce the voyage fuel cost. The most common LNG tanker propulsion alternatives according to IGU (2018) are described below.

Steam Turbines

The traditional propulsion system on LNG vessels is steam turbines. This system consists of boilers, usually fully or partially fueled with heavy fuel oil, generating steam to power the propulsion. All BOG from the tanks are used in the boilers, so no combustion unit is necessary. Another advantage is the low maintenance and operating cost due to the simplicity of the system. Still, the steam turbine has low thermal efficiency and large LNG carriers tend to require more power than what existing steam turbines can generate. In 2017, 63% of the LNG fleet was steam-based, however; these vessels are mostly smaller and older vessels (less than 150,000m^3 and over 10 years old).

Dual-Fuel Diesel Electric/Tri-Fuel Diesel Electric (DFDE/ TFDE)

DFDE systems can be fueled with both BOG and diesel oil and improve fuel efficiency by 25-30% compared to the traditional steam turbines. DFDE systems are electric propulsion systems powered by duel-fuel, medium-speed diesel generator sets. These diesel engines can run on either natural gas or marine diesel oil, and the engine operator can switch between the two fuel types. Unlike steam turbines, these propulsion systems need combustion units to burn excess BOG if necessary. This extra equipment requires extra maintenance. TFTE vessel is able to run on heavy fuel oil, diesel oil, and gas, thus further improving the ability to optimize propulsion efficiency. In 2017, 31% of the active world LNG fleet was equipped with DFDE or TFDE propulsion systems. So around 94% of the current fleet use either steam-based propulsion or DFDE/TFDE systems.

Slow-Speed Diesel (SSD) with a BOG Re-liquefaction Plant

This propulsion system consists of a conventional low-speed diesel engine generator sets fueled by heavy fuel oil or marine diesel oil. So instead of using the BOG as fuel, the gas is re-liquefied and led back into the cargo tanks. Vessels using this propulsion system also require a combustion unit to burn BOG when necessary. This propulsion system is advantageous when fuel oil is cheap compared to the BOG as it allows transportation of LNG without any loss of cargo. Today, the entire Q-class vessels use this propulsion system.

M-type, Electronically Controlled, Gas Injection (ME-GI)

The ME-GI utilizes high-pressure slow-speed gas injection engines that, unlike the Q-class propulsion system, can use BOG in the engine if necessary, instead of only re-liquefying the BOG. This allows better optimization of fuel consumption,

improving the fuel efficiency with around 15% over the TFDE engines. Only a small number of vessels in the current world fleet uses this propulsion system. However, around 42% of the vessels in the order book are equipped with this engine type.

Winterthur Gas & Diesel (WinGD) Low-Pressure Two Stroke Engine

This engine is an alternative to the DFDE propulsion system that reduces the capital cost of the propulsion system due to its lower cost gas handling system. Very few vessels are equipped with this engine type, and it accounted for around 2% of the 2017 order book.

2.2.2 Speed and Fuel Consumption

Changes in speed have a considerable impact on the voyage costs. Fuel consumption per distance unit can be estimated to be proportional to the second power of speed. Stopford (2013) uses the following formula to estimate fuel consumption

$$F = F^*(\frac{S}{S^*})^a \tag{2.1}$$

where F is fuel consumption in tons/day, F^* is fuel consumption when cruising speed is used, S is chosen speed, S^* is cruising speed, and a is usually 2 for steam turbines and 3 for diesel engines. Thus, increasing speed by 10 % may increase fuel consumption for a given leg by 20 %. Additionally, it is common to express a vessel's fuel consumption as a function of speed S and payload W, $F(S, W)$. Psaraftis and Kontovas (2013) presents for instance the following approximation

$$F(S, W) = k(p + S^q)(W + A)^{(2/3)}$$

where A is the lightship weight which is the weight of the vessel's structure, propulsion and engine, k, p and q are constants such that $k > 0$, $p \geq 0$ and $q \geq 3$.

3. Related Literature Review

In this Chapter, we present an overview of the relevant literature for our problem. In Section 3.1, an extensive survey on routing problems within the LNG industry is conducted. As the LNG literature is relatively young and limited in size, a survey on relevant literature that is closely related to the LNG routing and scheduling problem is presented in Section 3.2. The surveyed literature is summarized in Section 3.3.

3.1 LNG Delivery Planning

The LNG delivery planning problems are problems concerned with the routing and scheduling of transportation of LNG by vessels between liquefaction plants and regasification terminals. The majority of these problems are tactical planning problems with planning horizons between three months and one year. Some of these problems also include operational decision elements. We focus on this part of the LNG literature. A strategic LNG planning problem can be found in Koza et al. (2017), and a more general study of the LNG supply chain is presented in Andersson, Christiansen, and Fagerholt (2010). All the problem formulations reviewed in this section are modeled with discrete time.

3.1.1 LNG Inventory Routing Problems

The LNG inventory routing problem (LNG-IRP) can be considered a special case of the MIRP. To our knowledge, the LNG-IRP was first introduced in Grønhaug and Christiansen (2009). This problem seeks to maximize the profit of the operator by deciding the routing and scheduling of a heterogeneous fleet of LNG vessels, transporting one type of LNG between several liquefaction plants and regasification terminals, as well as determining the production quantity at the liquefaction plants and demand fulfillment at the regasification terminals. The problem also involves

inventory management at all plants and terminals, ensuring that the solution does not exceed the limits of the respective storages.

Moreover, each tank of the LNG vessels is modeled and managed. All vessels have to be fully loaded at a liquefaction plant, and partial unloading is allowed, as long as a whole number of tanks are completely unloaded. The problem adds complexity by considering boil-off, so completely unloaded means that some gas is still left in the tank to prevent the need for cool-down. The problem differs from pickup and delivery vehicle routing problems in that the number of visits to a port and the loading and unloading quantities are unknown. Also, there are no pickup and delivery pairs.

Grønhaug and Christiansen (2009) formulate both an arc-flow and a path-flow model to solve the LNG-IRP problem with a time horizon of two to four months. The path-flow model is studied further in Grønhaug, Christiansen, et al. (2010). A branch-and-price method is proposed to solve the problem. The solution method consists of a master problem that handles inventory management and ensures that the port capacity constraints are satisfied, while a set of subproblems generate columns representing vessel routes. Andersson, Christiansen, and Desaulniers (2016) reformulates the problem by splitting the routes and schedules into duties, where a duty is defined by the authors to be a sequence of ports starting in a loading port, visiting one or two unloading ports, before returning to a loading port. The authors propose a branch-and-bound algorithm where duties are generated a priori.

Fodstad et al. (2010) and Uggen et al. (2013) look at a richer LNG-IRP model, which includes a larger part of the LNG supply chain, both upstream and downstream, and considers contract management and sales in the spot market. The last-mentioned authors use a fix-and-relax time decomposition heuristic to construct a solution and then improve it. This is done by dividing the planning horizon into shorter intervals and solve a subproblem for each interval. This is done iteratively in a rolling horizon basis.

Goel et al. (2012) also add to the LNG-IRP introduced by Grønhaug and Christiansen (2009) by including the spot market. However, in this problem, the storage and contractual demand constraints are considered soft constraints, and the model seeks to minimize the total penalties from production loss, stockouts, and unsatisfied contractual demands at regasification terminals. Moreover, Goel et al. (2012) makes some simplifications by not allowing partial unloading and assuming that the boil-off quantities can be represented by the average boil-off loss on a typical voyage between two ports since the boil-off is usually a small fraction of the total vessel cargo. The authors propose a greedy construction heuristic and an

improvement heuristic based on a large neighborhood search method.

3.1.2 Annual Delivery Program

A second common type of LNG planning problem is the development of an annual delivery plan (ADP). Rakke, Stålhane, et al. (2011), Stålhane et al. (2012) and Rakke, Andersson, et al. (2014) all study essentially the same problem; developing an ADP for one of the largest LNG producers in the world. This producer operates only one liquefaction plant but can transport LNG to multiple regasification terminals, making this problem's structure simpler than the LNG-IRP. The producer must satisfy a set of long-term contracts but can also service LNG spot contracts. The producer is responsible for the LNG inventories at the single liquefaction plant, so the problem only includes inventory management at the liquefaction plant. It also manages a limited number of berths at the loading port and has to determine the routing and scheduling of a fleet of heterogeneous LNG vessels, with the possibility of chartering in extra vessel capacity. Vessels can also be unavailable due to certain pre-allocated activities such as maintenance. The vessels are assumed to always be fully loaded and unloaded, so boil-off does not need to be considered. The problem allows for the transportation of different types of LNG on distinct voyages. Another important distinction from the LNG-IRP is the longer planning horizon of one year and typically larger size of the problem. The objective of the ADP problem is to minimize the cost related to fulfilling the long-term contract and maximizing the revenue from spot sales.

In Rakke, Stålhane, et al. (2011), the authors use a RHH heuristic to solve an LNG inventory routing problem. This is done by dividing the planning horizon into shorter periods. In the proposed RHH, three periods are used, a frozen period, a central period and a forecast period. In iteration k of the procedure, the variables of the central period are determined, the variables of the frozen period are fixed and inherited from the central period in the previous iteration $k-1$, while the binary variables of the forecast period are relaxed and have a guiding role to avoid sub-optimal solutions. In the next iteration, the variables of the central period are frozen and the forecasting period becomes the new central period. Stålhane et al. (2012) propose a multi-start construction heuristic to generate a set of solutions. In order to improve these solutions, the search is intensified by solving a restricted version of the original mathematical model. In Rakke, Andersson, et al. (2014), the authors reformulate the same problem to a maritime inventory problem (MIRP) based on delivery patterns. The problem is then solved by an exact branch-price-and-cut algorithm.

Andersson, Christiansen, Desaulniers, and Rakke (2017) complements the previous

studies of the ADP problem by introducing four groups of inequalities to improve the lower bounds on the optimal value of the problem. The groups target the pricing of over- and under-deliveries, the quantities delivered, the timing of the deliveries and symmetry breaking.

Halvorsen-Weare and Fagerholt (2013) study a simplified version of the problem where the average cargo size is derived based on the loading capacities of the vessels, and time windows (both hard and soft) for the cargoes are assigned based on contractual agreements, without the possibility of optional (spot) cargoes. Therefore, the cargoes that should be delivered to specific regasification ports and time windows for deliveries will be known. Also, Halvorsen-Weare and Fagerholt (ibid.) assume that the fleet can be divided into disjoint groups. Despite the problem's complicating side constraints (inventory and berth capacity constraints), the authors solve the problem by separating the routing decisions from the scheduling decisions. In the first phase, the routing decisions are determined by local search heuristics. These decisions consist of deciding which cargoes should be served by which vessel and in what sequence. In the second phase, the scheduling problem is solved as a MILP and consists of deciding the start service time of each cargo such that the berth and inventory capacity constraints are satisfied. Halvorsen-Weare, Fagerholt, and Rönnqvist (2013) extends this problem by taking uncertainty in sailing times and daily LNG production into account. The paper develops robustness strategies and the resulting solutions are evaluated with a simulation model with a recourse optimization procedure.

Halvorsen-Weare and Fagerholt (2013) only consider time windows for delivery and penalty costs for deviations from these time windows. It also has bounds on delivery volumes but does not account for deviations in deliveries. Rakke, Stålhane, et al. (2011) include a penalty for both under and over delivery, while Stålhane et al. (2012) only have penalty for under delivery. Rakke, Stålhane, et al. (2011) and Stålhane et al. (2012) do not consider early or late delivery penalties. Furthermore, these formulations do not impose any bounds on the contract delivery volumes.

Mutlu et al. (2016) stands out as the most comprehensive among the existing formulations in terms of contract specifications. It includes both hard and soft time windows for delivery, lower and upper limits on delivery volumes, and penalty costs on deviations from expected delivery times and volumes. The paper is also, to our knowledge, the only one that allows split deliveries. Furthermore, this paper does not take the production rate at the loading port as a given parameter but models it as a decision variable that can be changed on a monthly basis. This allows the model to manage excess production after satisfying demand from long-term contracts, instead of assuming that the excess production is sold in the spot

market. The authors propose a vessel routing heuristic that iteratively constructs vessel routes such that the volume requirements and the delivery times of each contract are satisfied.

Al-Haidous et al. (2016) study a special case of the ADP problem where the fleet is homogeneous in capacity and speed but can have different cost structure, and where the objective is to minimize the number of vessels. This paper also introduces bunkering restrictions. The authors solve the problem to optimality by a MILP model and propose a fix-and-relax heuristic where the planning horizon is divided into four periods. For each period, all integrality constraints corresponding to future periods are relaxed while variables originating from previous periods are fixed.

3.2 Relevant Literature Related to the LNG Routing and Scheduling Problem

In this section, an extensive survey on related literature to the LNG routing and scheduling problem (LNG-RSP) is presented. The reasoning behind conducting this study is two-folded. First, the LNG literature is relatively young and is limited in size to the papers described in Section 3.1. Second, the LNG routing and scheduling problem has many structural elements in common with other well-studied routing problems like the multi vehicle pickup and delivery problem with time windows (m-VRPTW). Additionally, the LNG-RSP is tightly related to the maritime cargo routing and scheduling problem, which is extensively studied compared to the LNG-RSP. Since the literature on routing problems is vast in extent and scope, we consider only the part of the routing literature that utilizes solution methods that are variations of, or closely related to the solution methods applied to the LNG problems described above. Although we mainly focus on solution methods applied to maritime problems, other relevant applications with a related structure to our problem are included in the study.

As it may be observed by the reader, most of the solutions methods applied to the LNG problems described in Section 3.1 incorporate a MILP as part of the solution procedure. Such solution methods are called matheuristics.

Note that the only solution method discussed in Section 3.1 that does not incorporate a MILP in the search procedure is a large neighborhood search (LNS) proposed by Goel et al. (2012) to solve an LNG-IRP. However, many of the LNS algorithms in the routing literature use often a MILP as a local search operator. For this reason, LNS is discussed below in the context of matheuristics.

Boschetti et al. (2009) defines matheuristics as "heuristic algorithms made by the interoperation of metaheuristics and mathematical programming techniques". The improvement of the capabilities of general-purpose solvers in recent years has led to an increasing interest in designing algorithms where mathematical programs (MP) are incorporated in one or many phases of the solution procedure. Archetti and Speranza (2014) classify matheuristics used to solve routing problems into three classes; decomposition approaches, improvement heuristics, Branch-and-price approaches. The structure of the rest of this section is inspired by the same literature survey.

3.2.1 Decomposition-Based Matheuristics

In the decomposition-based matheuristics, the main problem is divided into smaller sub-problems, which then are solved independently to optimality or near-optimality by a MILP model.

Cluster first-route second approaches

In addition to application-specific decisions, decisions taken in routing problems can be divided into two categories: 1) clustering customers and assigning each cluster to a vehicle 2) deciding the sequence of customers to be visited in each route. This division of decisions makes decomposition-based matheuristics convenient to solve routing problems. The first paper that uses a cluster first - route second approach is written by Fisher and Jaikumar (1981). The authors solve a VRP by using a two-phase approach. In the first phase, "seed customers" are heuristically selected. Then, the problem of assigning the rest of customers to the seed customers is formulated and solved as a generalized assignment problem. In the second phase, the authors use TSP to determine the sequence of customers on each route. Bramel and Simchi-Levi (1995) solves the same problem in a similar approach, however, they determine the clustering of customers by solving a capacitated concentrator location problem (CCLP). In addition to VRP, the authors apply the same method to an IRP and show that it performs well on both problems.

Two-phase approaches

Cluster first - route second is not the only two-phase approach used to solve routing problems. For instance, a typical tramp routing and scheduling problem can be shown as a generalization of a traveling salesman problem which is NP-hard (Hemmati et al., 2014). For that reason, these problems are often reformulated as a set partitioning problem where columns represent feasible vessel schedules.

This strategy reduces the computation time considerably since all columns (or at least a subset if an approximation is needed) that satisfy a set of hard constraints can be generated a priori. This approach is followed by, among others, Brown et al. (1987) who generate all feasible schedules for a fleet of crude oil tankers and solve the problem with thousands of binary variables to optimality in less than a minute. Similarly, the set partitioning approach is also used by Bausch et al. (1998) to design an optimization-based decision support system for a multi-national company with a fleet of coastal tankers; Fagerholt (2001) for solving the aforementioned multi-ship pickup and delivery problem (m-PDP) with soft time windows and fixed cargo sizes; and recently Brønmo, Christiansen, et al. (2007) for solving a m-PDP with flexible cargo sizes. In order to generate feasible columns a priori, Fagerholt (2001) formulate the problem of finding the optimal schedule for each cargo-ship set as a Traveling Salesman Problem with Capacity constraints, Soft Time Windows, and Precedence constraints (TSP - CSTWPC). Similarly, Brønmo, Christiansen, et al. (2007) use the same approach, but modify Fagerholt (2001) to account for hard time windows and flexible cargoes.

Rolling Horizon

While all the approaches described above are known as "one-shot" approaches, referring to solving two phases one time, more advanced algorithms that apply the cluster first - route second idea in an iterative manner have been developed and often used to solve inventory routing problems (IRP). Such an iterative approach is used in two of the LNG problems described in Section 3.1; Rakke, Stålhane, et al. (2011), and Uggen et al. (2013). Agra et al. (2014) propose a similar approach to Rakke, Stålhane, et al. (2011) to solve a short sea routing and scheduling problem for an oil company where inventory constraints have to be satisfied at the demand side, however the authors combine RHH with local branching to efficiently solve the problem of each sub-horizon and feasibility pump to speed up the process of finding an initial feasible solution. In Campbell and Savelsbergh (2004), the authors solve an IRP with a long time horizon by decomposing the problem into two problems. In the first problem, the authors use an integer program to assign customer deliveries to days and determine how much to deliver to these customers. The output of the first problem is a delivery schedule for the next k days. In the second problem, the time horizon is shortened and only a few days of the original k days are considered. At this phase, the decision process is focused on constructing routes and schedules, which is done by insertion heuristics. The output of the second problem is then used to revise the input of the first problem. The algorithm is repeated on a rolling-horizon base.

Partial Optimization approaches

In certain routing problems, heuristic operators are used to solve a part of the problem. Then, all decision related to this part is fixed and one or more MILP models are used to solve the remaining part of the problem. Usually, a MILP is used to solve the part that does not include the routing decision in the problem as these are typically the most difficult for the heuristic operators to handle.

This approach is used in Coelho et al. (2011) to solve an IRP with transshipment. The authors propose an adaptive large neighborhood search (ALNS) comprised of several heuristics operators used to manipulate the vehicle routes, while a network flow algorithm is used to solve the remaining problem of determining transshipment moves and quantities delivered to customers.

In Coelho et al. (2012), the authors apply the same methodology as in Coelho et al. (2011) to solve an IRP with multiple vehicles and different consistency requirements. A similar methodology is applied in Adulyasak et al. (2012) for the solution of a production routing problem (PRP). The authors use heuristic operators to determine vehicle routes and delivery days, while a network flow model is optimally solved to determine production and delivery quantities. Demir et al. (2012) suggest an ALNS for the solution of the pollution routing problem (PLR). This is an extension of the VRPTW that includes the optimization of the speed on each route to minimize an objective function that takes fuel, emission and driver costs into account. Standard heuristic operators are used to search the solution space while a MILP model is solved to determine the optimal speed for each route segment.

The ALNS paradigm was introduced by Ropke and Pisinger (2006) as an extension of the principle of large neighborhood search (LNS) (Shaw, 1997) in which a solution is repeatedly destroyed and repaired through the application of various heuristic operators. The ALNS heuristic adds to this mechanism by selecting the operators in a probabilistic and adaptive fashion, based on the prior performance of the operators. This method was initially applied to the pickup and delivery problem with time windows (Ropke and Pisinger 2006) and to the vehicle routing problem (Pisinger and Ropke 2007), and has later been successfully adapted to a wide variety of routing problems, including inventory-routing (Coelho 2012a, Coelho 2012b), production-routing (Adulyasak 2014), stochastic arc routing (Laporte 2010) and cumulative vehicle routing (Ribeiro and Laporte, 2012).

Several hybrid extensions of the ALNS can be found in the literature. In Belhaiza (2019) a hybrid ALNS is used to solve a dial-a-ride problem with time windows (DARPTW). The heuristic combines an ALNS search procedure with a genetic

crossover of a pool of found solutions that take effect if the ALNS cannot improve the solution during a certain number of iterations. Gu (2018) propose a hybridization of ALNS and local search. In each iteration of the algorithm, a destroy and repair method is used to generate a new solution. Seven local search moves are applied iteratively on the solution until no improvement achieved. A similar method is used in Mahdi (2018) to solve a multi-depot, multi-compartment vehicle routing problem. After a large neighborhood search is used to find a new solution, a variable neighborhood search is applied if the new solutions is better than the current solution.

3.2.2 Improvement Based Matheuristics

In improvement based matheuristics, a MILP is utilized to improve a solution obtained by a heuristic. The purpose of using MILP as an improvement method is two-divided. First, MILP can be used once to improve the best solution found by another heuristic. Second, it can be employed as an intensification operator or a local optimizer inside a searching procedure. From the LNG literature, Stålhane et al. (2012) is classified as an improvement based matheuristic (Archetti and Speranza, 2014).

In some of the improvement based matheuristics, MILP is used for local optimization. In Yıldırım and Çatay (2014), the authors solve VRP by first generating a set of routes by using ant colony optimization (ACO) and second using a set partitioning program to choose the best routes. These routes are then fed back to ACO and used to update the pheromone trails. Another example of improvement based matheurisitc is presented in Rodrıguez-Martın and Salazar-González (2011). The authors solve the multi-Commodity one-to-one pickup-and-delivery traveling salesman problem by first constructing a feasible solution by a greedy randomized adaptive search heuristic (GRASP). The solution is then improved by fixing only a subset of the variables and then improved by using a branch-and-cut approach. The solution method is developed further by the same authors in Rodrıguez-Martın and Salazar-González (2012). The new solution method consists of a construction phase similar to the previous solution method, however, they propose a variable neighborhood Descent scheme (VND) consisting of among others a MILP operator similar to the one presented in Rodrıguez-Martın and Salazar-González (2011).

3.2.3 Branch-and-Price and Column Generation-Based Approaches

In column generation-based matheuristics, a problem is decomposed into a master problem and a sub-problem called the pricing problem. Reformulating a problem

to a column generation form (Dantzig-Wolfe decomposition) often incurs the generation of a huge number of columns. For this reason, a restricted master problem with only a subset of all columns is often used. The pricing problem has the role of generating columns that may improve the objective function value in the master problem. A branch-and-price is considered an exact solution algorithm only if the pricing problem is solved to optimality in each iteration. This is to ensure that all improving columns are included in the master problem. As described in Section 3.1, both Grønhaug, Christiansen, et al. (2010) and Rakke, Andersson, et al. (2014) apply a branch-and-price/column generation based approach to solve a LNG-IRP.

A similar approach is utilized in Brønmo, Nygreen, et al. (2010). The authors solve the same problem as Brønmo, Christiansen, et al. (2007) by using a Dantzig-Wolfe approach. The problem is decomposed into a master- and subproblem. The master problem is a set partitioning problem with the most promising schedules for all vessels and the subproblem that finds these schedules is formulated as a shortest path problem and solved by dynamic programming. When the algorithm converges and gives a fractional solution, a branch-and-price procedure is used to find the optimal solution. Wen et al. (2016) follow the same strategy but combines the aforementioned problem with speed optimization.

3.3 Summary of Literature Review

In this Chapter, a literature review on LNG problems with a focus on the applied solution methods is presented. Additionally, the literature review covers also a segment of the literature on routing problems that are considered tightly related to the LNG routing and scheduling problem.

The recent research on optimization of LNG transportation is two-fold and consists of LNG-IRP and the problem of developing an annual delivery plan. While the majority of papers studying LNG-IRP focus on operational decisions within a planning horizon of 2-4 months, many of the papers that study the development of an annual delivery plan focus on the longer planning horizon of one year and larger size of the problem. Moreover, the level of detail that is included in decision modeling is varying. For instance, Halvorsen-Weare and Fagerholt (2013), Goel et al. (2012) and Rakke, Stålhane, et al. (2011) make some simplifications in modelling quantity and BOG.

A general challenge with the LNG problems discussed in Section 3.1, compared to the standard maritime cargo routing problems and m-VRPTW, is the unknown loading and unloading cargo due to quantity flexibility in deliveries and its depen-

dency on factors like heel-out, cool-down, speed, boil-off, cool-down and sailing time. In addition to their combinatorial complexity, the dependency between a large number of different decisions that are continuous makes these problems challenging to model with a sufficient level of detail and solve within a reasonable time. Many of the surveyed papers address this problem by either decomposing the problem and solving it in two phases or on a rolling horizon basis. Our hypothesis is that utilizing a decomposition based approach might be essential to obtain high-quality solutions within a reasonable time. However, the success of such an approach is highly dependent on the ability to find the right decomposition for our problem. It is worth mentioning that the problem studied in this Thesis has a relatively short planning horizon and long sailing times, hence using rolling horizon based methods may result in a high degree of fragmentation with few decisions to be determined in each sub-horizon. When it comes to partial optimization approaches like ALNS, they are rarely used as solution methods in the LNG literature, despite their success on routing problems like VRP. Our hypothesis is that the ability of an algorithm like ALNS to move around in the search space makes it suitable for highly constrained problems like the one studied in this Thesis.

Another popular approach among the surveyed papers, especially inventory routing problems, is the branch-and-price and column generation-based approach. Although these approaches seem appropriate for the LNG-RSP at first glance, the restriction of allocated volume at the liquefaction ports and its impact on the feasibility of the vessels' routes make it difficult to efficiently separate the problem into a master problem and a sub-problem.

The aim of this thesis is to develop a solution method that is able to solve a large-scale LNG routing and scheduling problem as described in 4. It is crucial that the proposed solution method is able to find high-quality solutions with low computational time.

4. Problem Description

This report addresses a short-term planning problem for LNG producers. The problem includes routing and scheduling of a fleet of LNG vessels between liquefaction and regasification facilities.

The producer can produce multiple types of LNG in its liquefaction plants and is obliged to service a set of customers with long-term contracts. The amount to deliver and pick up, delivery and pickup locations, type of LNG and corresponding time windows for each cargo are known in advance, according to the annual delivery program. There is some flexibility concerning each load delivered to a customer; however, the accumulated volume depleted from a liquefaction plant in a period of time cannot exceed an allocated volume of LNG in that period. If a liquefaction plant's allocated volume exceeds the LNG amount depleted in a period of time, the deviant amount is either sold as spot cargoes on the market or transferred to the next period. A spot cargo may be delivered with the producer's own vessel if there is any idle (DES) or may be picked up by the customer if there is any empty pickup slot at the liquefaction plant (FOB). Furthermore, the producer has also the ability to charter vessels for one-off delivery if the producer does not have enough vessels to serve long-term contracts.

The producer operates a fleet of LNG vessels and is responsible for the routing and scheduling of the vessels. Each vessel can be owned by one or a group of customers, the producer, or a combination of both. As a result, some vessels are restricted to only visit ports agreed upon by the owners. The customers can also set such restrictions as they can require LNG from certain pickup ports. Moreover, vessels are heterogeneous in terms of capacity, speed, fuel and propulsion characteristics, boil-off technology and the different ports they can visit. Although a vessel can carry all types of LNG, it can only carry one type on board during a single voyage. Furthermore, the vessels are always filled close to their full capacities at the liquefaction plant.

The LNG vessels are allowed to visit one loading port and 1-2 unloading ports on the same voyage. On the return leg from unloading port to loading port, the vessel's tanks are kept cool by a small amount of LNG left in the tanks. For this reason, the tanks have to contain enough LNG to keep the boil-off process going until a new loading operation begins. The producer can also choose to empty the vessel's tanks at an unloading port completely. However, in that case, the vessel will be unavailable for some time to undergo a cool-down process at the next loading port before a loading operation can take place. Also, vessels that usually use BOG as fuel will have to switch to another fuel type. This might represent an economic loss or opportunity.

Consequently, the amount delivered to the customer depends on the duration of the voyage from loading to unloading port and the producer's decision to keep either the vessel's tanks cool on the return leg or perform a complete depletion of LNG.

Since the vessels do not have a depot they can return to, the availability of the vessels at the start of the planning horizon depends on the vessels' schedule at that time. If the planning horizon starts while a vessel performs a loading/unloading operation at a port, it first becomes available when it is done with the operation. However, one can choose to redirect a vessel that is en route to some destination if this is appropriate.

This problem aims to determine how customers should be serviced. That is, deciding which vessels that service which customers, in what sequence and when they are serviced. This provides the producer with information about the sequence of customers associated with each vessel, which spot customers to service and which customers with fixed contracts to satisfy through chartering. The decisions should also include how much to load/unload at pickup and delivery ports, sailing speed between ports, and how much to sell as FOB at the producer's ports.

These decisions should be determined based on the producer's objective. This objective consists of sales due to the LNG delivered to long-term contracts and the LNG sold to the spot market. The producer's costs in the short term are related to cool-down and transportation. Costs related to completely emptying the tanks include the cost difference between fuel mix with and without BOG and a one-time cool-down charge in the loading port. Additionally, transportation costs include variable costs like bunker oil, port, and canal fees. Both components are dependent on the vessel size, capacity, ports to be visited and the duration of the voyage. The producer may also charter vessels to serve long-term contracts. This may induce a financial profit or loss. Additionally, the producer is charged a penalty cost if a long-term contract is breached.

5. Mathematical Model

In the following we introduce an arc-flow model for the problem introduced, the short-term routing and scheduling of liquefied natural gas transportation. We also have dedicated sections for linearization of non-linear constraints as well as for initiatives relating to variable reduction. An important distinction in our models is that node and port have separate meanings (e.g. there are many nodes belonging to a single port).

Solution Requirements

The solution needs to satisfy (1) routing constraints, (2) stopover routing constraints (visiting two subsequent gasification ports where the LNG has to come from liquefaction ports that are allowed to deliver to both of the gasification ports, (3) time constraints, (4) quantity balance constraints for vessel legs, (5) heel management constraints, (6) quantity windows at both types of ports, (7) inventory constraints for liquefaction ports and FOB sale. Furthermore, the solution also need to satisfy constraints relating to vessel speeds.
The solution also needs to maximize the estimated profit for the LNG supplier, and it thus needs to sum all the relevant revenues and costs into the search for the optimal solution.

Solution Elements

The solution, or the model output, needs to provide information on the scheduling and routing of mandatory contracts and select the subset of optional contracts that should be delivered. More specifically, the solution needs to include where the vessels are sailing, how fast they will be sailing and when they are departing and arriving. The solution should also provide information on FOB sales.
The solution will automatically provide information on the cargoes which are executed with partial unloading (i.e. stopover), and it will also indirectly provide

information on when/if cool down is needed and breached contracts.

5.1 Mathematical Model

Sets

V : Vessels
N^P : Pickup nodes
N^D : Delivery nodes
N : $N^P \cup N^D$, all nodes
N^C : Fixed contracts
N^O : Optional contracts
N^D : $N^C \cup N^O$
N_v^P : Pickup nodes vessel v may visit
N_{iv}^P : Pickup nodes which may deliver gas to i and which may be visited by vessel v
N_i^P : Pickup nodes which may deliver gas to i
N_v^D : Delivery nodes vessel v may visit
N_v : $N_v^P \cup N_v^D \cup (o(v),d(v))$, where $o(v)$ and $d(v)$ is the artificial origin and destination
A : Arcs between nodes $N \times N$
A_v : Arcs for vessel v, corresponding to x-variables created in optimization engine for vessel v
P : Pickup ports
M : Periods
N_{pm}^{PO} : Pickup nodes corresponding to port p in month m
H : Speed modes

Parameters

R_i : Revenue per unit for contract i
B_v : Boil-off loss/consumption for vessel v per day
B_i^{CH} : Boil-off loss/consumption if contract i is serviced by a chartered vessel
T_{ijv}^S : Sailing time from node i to j using vessel v
T_{ij}^{CH} : Sailing time from node i to j using a chartered vessel

T_{iv}^{Q} : Loading/unloading quantity per time unit at node i using vessel v

D_v : Difference in cost for vessel v by using only bunker oil instead of the predetermined fuel m
(can be zero if the vessel never uses boil-off as fuel)

C_{ij}^{CH} : One-time cost if delivery node j is serviced from node i by a chartered vessel

C_{iv}^{CD} : One-time charge related to cool-down in loading port i for vessel v

C_{ijv}^{OP} : Operation cost from i to j using vessel v
(Could be any predetermined fuel mix of bunker oil and boil-off gas)

Π_i : Penalty if contract i is breached

Q_v^S : Required fixed buffer in order to avoid cool-down for vessel v

Q_v^F : Quantity factor (0-1) when leaving liquefaction port for vessel v

Q_i^{CH} : Quantity delivered to customer if contract i is serviced by a chartered vessel

T_{iv}^{CD} : Required cool-down time in loading port i for vessel v

Q_i^{MIN} : Lower quantity limit when servicing contract i

Q_i^{MAX} : Upper quantity limit when servicing contract i

T_i^{MIN} : Lower time limit for start of operation at node i

T_i^{MAX} : Upper time limit for start of operation at node i

V_v^{CAP} : Cargo capacity of vessel v

Q_{pm}^{A} : Allocated volume at port p in period m

Q_p^I : Volume transferred from last month in previous planning period at port p

Q_v^{CD} : Gas volume required to cool down vessel v

Q^{CDC} : Gas volume required to cool down chartered vessel

F_i^{MIN} : Minimum gas volume for FOB sale at pick-up node i

F_i^{MAX} : Maximum gas volume for FOB sale at pick-up node i

R_i^{FOB} : Revenue per unit for FOB sale at pick-up node i

T_{ijvh}^{S} : Sailing time from node i to j using vessel v with speed mode h

D_{vh} : Difference in cost for vessel v with speed h by using only bunker oil instead of the predetermined fuel mix (can be zero if the vessel never uses boil-off as fuel)

C_{ijvh}^{OP} : Operation cost from i to j using vessel v with speed mode h

Variables

x_{ijv} : Equal to 1 if vessel v sails from i to j, 0 otherwise

z_{ij} : Equal to 1 if a chartered vessel service delivery node j from pick-up node$i$

q_{iv} : Quantity loaded/unloaded at node i by vessel v

l_{iv} : Quantity on board when vessel v is leaving node i

y_{ijv} : Equal to 1 if vessel v sails from i to j with no LNG left,
0 otherwise or if the vessel does not sail the given route at all

t^S_{iv} : Service time start at node i for vessel v

t^E_{iv} : Service time end at node i for vessel v

s_i : Equal to 1 if contract i is breached, 0 otherwise

a_i : Equal to 1 if contract i is served directly from liquefaction port
or by charter (without stopover) at gasification port), 0 otherwise

q^{FOB}_i : FOB quantity sold at pick-up node i

f_i : 1 if FOB cargo is sold at node i, 0 otherwise

q^T_{pm} : Quantity transferred at port p from month m to month $m+1$

w_{ijvh} : Weight of speed mode h for vessel v sailing from node i to j

d_{ijvh} : Equal to 1 if speed is weighted by speed mode w_{ijvh} and $w_{ijv(h+1)}$, 0 otherwise

Objective Function

$$\max \sum_{v\in V}\sum_{i\in N_v^D} R_i\, q_{iv} - \sum_{v\in V}\sum_{(i,j)\in A_v}\sum_{h\in H_v} C^{OP}_{ijvh}\, w_{ijvh} + \sum_{v\in V}\sum_{(i,j)\in A_v|i\in N_v^D}\sum_{h\in H_v} D_{vh}\, T^S_{ijvh} w_{ijvh}\, y_{ijv}$$
$$- \sum_{v\in V}\sum_{(i,j)\in A_v|i\in N_v^D} C^{CD}_{jv}\, y_{ijv} - \sum_{i\in N^D}\Pi_i\, s_i + \sum_{i\in N^P} R^{FOB}_i\, q^{FOB}_i$$
$$+ \sum_{i\in N^P}\sum_{j\in N^D} (R_i\, Q^{CH}_i - C^{CH}_{ij})\, z_{ij} \tag{5.1}$$

The objective function (5.1) maximizes the profit gained from operating a fleet of LNG vessels. The first term is revenues from LNG sales to long term contracts and spot market. The second term is sailing costs according to the chosen speeds including port and canal fees. The third and fourth terms are fuel switching costs and one-time charge in case of cool-down in ports, respectively. The fifth term represents penalty/loss in case of breaching a long term contract. The sixth term is profit from FOB sales. The seventh term is the total profit from chartering activities. It consists of revenues from servicing delivery nodes by chartered vessels, and sailing and chartering costs for chartered vessels.

Constraints

Routing Constraints

$$\sum_{v\in V}\sum_{i\in N_v} x_{ijv} + \sum_{i\in N} z_{ij} \le 1 \qquad \forall\, j \in N^O \tag{5.2}$$

$$\sum_{v\in V}\sum_{i\in\{N_v^D, o(v)\}} x_{ijv} \le 1 \qquad \forall\, j \in N^P \tag{5.3}$$

$$\sum_{j\in N_v \setminus \{o(v)\}} x_{o(v)jv} = 1 \qquad \forall\, v \in V \tag{5.4}$$

$$\sum_{i\in N_v^D \cup \{o(v)\}} x_{id(v)v} = 1 \qquad \forall\, v \in V \tag{5.5}$$

$$\sum_{j \in N_v} x_{ijv} - \sum_{j \in N_v} x_{jiv} = 0 \qquad \forall\, v \in V,\, i \in N_v \setminus \{o(v), d(v)\} \tag{5.6}$$

Constraints (5.2) enable but don't oblige vessels to serve contracts in the spot market. In addition, they ensures also that each spot contract is served at most once. Constraints (5.3) ensures that each pickup node is departed from at most once. Constraints 5.4-5.6 represent the flow conservation constraints and describe the flow of vessels through each node. (5.4) and (5.5) ensure that each vessel leaves the artificial origin node and visits the artificial destination node once. In addition, constraints (5.6) make sure that all vessels that visit a node, also leave the node except the artificial origin and destination nodes.

Stopover Routing Constraints

$$\sum_{v \in V} \sum_{i \in N^P_{jv}} x_{ijv} + s_j + \sum_{i \in N^p_j} z_{ij} = a_j \qquad \forall\, j \in N^C \tag{5.7}$$

$$\sum_{v \in V} \sum_{i \in N^P_{jv}} \sum_{k \in N^D_v} x_{ikv}\, x_{kjv} = (1 - a_j) \qquad \forall\, j \in N^C \tag{5.8}$$

Constraints (5.7) and (5.8) require that fixed contracts are either serviced directly from a feasible pickup node (or the origin), or after a stopover at a delivery node. In the latter case, the vessel visiting the stopover node must leave from a pickup port which is feasible for both delivery nodes.

Time Constraints

$$t^E_{iv} = t^S_{iv} + \frac{q_{iv}}{T^Q_{iv}} \qquad \forall\, v \in V,\, i \in N_v \setminus \{o(v), d(v)\} \tag{5.9}$$

$$x_{ijv}(t^E_{iv} + \sum_{h \in H_v} T^S_{ijvh} w_{ijvh} + T^{CD}_{jv}\, y_{ijv}) \leq t^S_{jv} \qquad \forall\, v \in V, (i,j) \in A_v \mid j \in N^P_v \tag{5.10}$$

$$x_{ijv}(t^E_{iv} + \sum_{h \in H_v} T^S_{ijvh} w_{ijvh}) \leq t^S_{jv} \qquad \forall\, v \in V, (i,j) \in A_v \mid j \in N^D_v \tag{5.11}$$

$$T_i^{MIN} \sum_{j \in N_v \cup \{d(v)\}} x_{ijv} \leq t_{iv}^S \leq T_i^{MAX} \sum_{j \in N_v \cup \{d(v)\}} x_{ijv} \qquad \forall\, v \in V,\, i \in N_v \setminus \{o(v), d(v)\} \tag{5.12}$$

Constraints 5.9-5.11 ensure time compatibility at and between nodes. Constraints (5.9) defines the relationship between the time of service start time and service end time in each node. In constraints (5.10), the service start time at a loading node, node j, is greater than or equal to the sum of end service time at the unloading node, node i, sailing time from i to j, and time required to run a cool-down process if the vessel's tanks are totally emptied at delivery node. (5.11) ensure that the time of service start at an unloading node is at greater than or equal to the sum of the service end time at the previous node and sailing time between them. The time compatibility constraints in (5.10) and (5.11) are modelled as inequalities to allow for waiting time and enable vessels to arrive at port earlier than start of time window. However, vessels cannot dock at the assigned slot or start loading/unloading operation before the start of time window. Constraints (5.12) represent time windows at the pickup and delivery nodes.

Quantity Balance Constraints for Vessel Legs

$$x_{ijv}(l_{iv} - B_v\,(t_{jv}^S - t_{iv}^E)\,(1 - y_{ijv}) + q_{jv} - l_{jv}) = 0 \quad \forall\, v \in V, (i,j) \in A_v \mid j \in N_v^P \tag{5.13}$$

$$x_{ijv}(l_{iv} - B_v\,(t_{jv}^S - t_{iv}^E) - q_{jv} - l_{jv}) = 0 \qquad \forall\, v \in V, (i,j) \in A_v \mid j \in N_v^D \tag{5.14}$$

Constraints (5.13) and (5.14) represent the load management on board the vessel. (5.13) represent the situation when the end node is a pickup node. It ensures that quantity on board when leaving a pickup node is equal to the sum of the quantity on board when leaving previous node minus the loss due to boil-off (if the vessel is non-empty when leaving the delivery node) plus the loaded quantity at the delivery node. Constraints (5.14) represent the situation when the end node is a delivery node. It ensures that quantity on board when leaving a delivery node is equal to the sum of the quantity on board when leaving previous node minus the loss due to boil-off and unloaded quantity at the delivery node. Note that constraints (5.14) apply for vessels at delivery nodes that come from pick-up or delivery nodes. Additionally, if a vessel has a re-liquefaction technology on board, B_v can easily be set to zero to reflect the fact that no boil-off is produced

Heel Management

$$l_{iv} \sum_{j \in N_v^P} y_{ijv} \leq 0 \qquad \forall\, v \in V,\ i \in \{N_v^D, o(v)\} \tag{5.15}$$

$$y_{ijv} \leq x_{ijv} \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N_v^P \tag{5.16}$$

$$(1 - \sum_{j \in N_v^P} y_{ijv}) \sum_{j \in N_v^P} x_{ijv}((t_{jv}^S - t_{iv}^E)B_v + Q_v^S) \leq l_{iv} \quad \forall\, v \in V,\ i \in N_v^D \tag{5.17}$$

Constraints (5.15) make sure that the leaving quantity at a delivery port is zero if there is no heel on board.Constraints (5.16) make sure that if vessel v does not sail from i to j the heel variable is set to zero.

Constraints (5.17) ensure that the quantity of LNG on board before arriving at a pick-up node (leaving a delivery node) consists of a buffer and enough quantity to maintain the tanks cool by boil-off, if cool-down is to be avoid on the given sailing leg.

Other Quantity Constraints

$$Q_i^{MIN} \sum_{j \in N_v \cup \{d(v)\}} x_{ijv} \leq q_{iv} \leq Q_i^{MAX} \sum_{j \in N_v \cup \{d(v)\}} x_{ijv} \qquad \forall\, v \in V,\ i \in N_v^D \tag{5.18}$$

$$Q_v^F\, V_v^{CAP} \sum_{j \in N_v \cup \{d(v)\}} x_{ijv} \leq l_{iv} \leq V_v^{CAP} \sum_{j \in N_v \cup \{d(v)\}} x_{ijv} \qquad \forall\, v \in V,\ i \in N_v^P \tag{5.19}$$

$$q_{iv} \leq l_{iv} \qquad \forall\, v \in V,\ i \in N_v^P \tag{5.20}$$

$$l_{iv}(1 - \sum_{j \in N_v} x_{ijv}) \leq 0 \qquad \forall\, v \in V,\ i \in N_v \setminus \{d(v)\} \tag{5.21}$$

Constraints (5.18) ensure that the quantity unloaded is within the quantity window for delivery nodes, similarly, constraints (5.19) impose quantity windows at the pick-up nodes. These quantity windows are relating to vessel stability and capacity. In addition, we have quantity windows (or balance) constraint relating to a pickup port (consisting of many pickup nodes).

Constraints (5.20) make sure that the leaving quantity at a pickup port is greater than or equal to the pickup quantity.Constraints (5.21) make sure that l_{iv} is zero if vessel v does not sail from i.

Quantity Balance for Ports and FOB Sale

$$\sum_{i \in N_{pm}^{PO}} \sum_{v \in V} q_{iv} + \sum_{v \in V} \sum_{i \in N_v^D \cup \{o(v)\}} \sum_{j \in N_{pm}^{PO}} y_{ijv} Q_v^{CD} + \sum_{i \in N_{pm}^{PO}} \sum_{j \in N^D} (Q_j + B_j^{CH}\, T_{ij}^{CH} + Q^{CDC}) z_{ij}$$

$$+ \sum_{i \in N_{pm}^{PO}} q_i^{FOB} = Q_{pm}^A + q_{p,m-1}^T - q_{pm}^T \qquad \forall\, p \in P,\ m \in M \qquad (5.22)$$

$$F_i^{MIN} f_i \le q_i^{FOB} \le F_i^{MAX} f_i \qquad \forall\, i \in N^P \qquad (5.23)$$

$$\sum_{j \in N_v^D \cup \{d(v)\}} x_{ijv} \le 1 - f_i \qquad \forall\, i \in N^P \qquad (5.24)$$

$$Q_p^I \le q_{p,0}^T \le Q_p^I \qquad \forall\, p \in P \qquad (5.25)$$

Constraints (5.22) must hold for every port and every month. It states that the sum of pickup quantities picked up by own vessels and chartered ones, the quantities spent during cool-down processes and quantities sold as FOB must equal the net supply.

Constraints (5.23) must hold for every pickup node. It states that a spot sale must satisfy a given quantity window.

Constraints (5.24) must hold for every pickup node. It states that a spot sale can only take place if there are no outgoing arcs from the given node.

Constraints (5.25) assign the correct value for the input transfer quantity (from the last period of the previous planning horizon).

Speed Constraints and Speed Interpolating Constraints

$$\sum_{h \in H_v} w_{ijvh} = x_{ijv} \qquad \forall\, v \in V,\ (i,j) \in A_v \qquad (5.26)$$

$$w_{ijv1} \leq d_{ijv1} \qquad \forall\, v \in V,\ (i,j) \in A_v \qquad (5.27)$$

$$w_{ijvh} \leq d_{ijv(h-1)} + d_{ijvh} \qquad \forall\, v \in V,\ (i,j) \in A_v,\ h \in H_v \setminus \{1, |H_v|\} \qquad (5.28)$$

$$w_{ijv|H_v|} \leq d_{ijv(|H_v|-1)} \qquad \forall\, v \in V,\ (i,j) \in A_v \qquad (5.29)$$

$$\sum_{h \in H_v \setminus \{|H_v|\}} d_{ijvh} = 1 \qquad \forall\, v \in V,\ (i,j) \in A_v \qquad (5.30)$$

Constraints (5.26) - (5.30) makes sure that the model is able to choose speeds with corresponding fuel cost which are close to the actual speed-fuel curve. This is achieved by linear interpolation between actual points on the curve, these points correspond to the speed modes H and their corresponding sailing times T^S_{ijvh} and operating costs C^{OP}_{ijvh}. Furthermore, w_{ijvh} are the corresponding weights between to adjacent points. Lastly, using the variables d_{ijvh} we restrict the weights to in fact be a linear combination of no less and no more than two points and that the points need to be adjacent to each other.

Non-Negativity and Binary Constraints

$$x_{ijv} \in \{0,1\} \qquad \forall\, v \in V,\ (i,j) \in A_v \qquad (5.31)$$

$$y_{ijv} \in \{0,1\} \qquad \forall\, v \in V,\ i \in N^D_v,\ j \in N^P_{jv} \qquad (5.32)$$

$$s_i \in \{0,1\} \qquad \forall\, i \in N^C \qquad (5.33)$$

$$a_j \in \{0,1\} \qquad \forall\, j \in N^D_v \qquad (5.34)$$

$$t^S_{iv} \geq 0 \qquad \forall\, v \in V,\ i \in N_v \setminus \{o(v)\} \qquad (5.35)$$

$$t^E_{iv} \geq 0 \qquad \forall\, v \in V,\ i \in N_v \setminus \{d(v)\} \qquad (5.36)$$

$$q_{iv} \geq 0 \qquad \forall\, v \in V,\ i \in N_v \setminus \{o(v), d(v)\} \qquad (5.37)$$

$$l_{iv} \geq 0 \qquad \forall\, v \in V,\ i \in N_v \setminus \{d(v)\} \qquad (5.38)$$

$$q^{FOB}_i \geq 0 \qquad \forall\, v \in V,\ i \in N^P \qquad (5.39)$$

$$q^T_{pm} \geq 0 \qquad \forall\, p \in P,\ m \in M \qquad (5.40)$$

$$f_i \in \{0,1\} \qquad \forall\, i \in N^P \qquad (5.41)$$

$$z_{ij} \in \{0,1\} \qquad \forall\, i \in N^P,\ j \in N^D \tag{5.42}$$

$$d_{ijvh} \in \{0,1\} \qquad \forall\, v \in V,\ (i,j) \in A_v,\ h \in H_v \setminus \{|H_v|\} \tag{5.43}$$

$$w_{ijvh} \geq 0 \qquad \forall\, v \in V,\ (i,j) \in A_v,\ h \in H_v \tag{5.44}$$

Binary requirements on the flow variables are imposed by constraints 5.31-5.34. Constraints 5.35-5.40 impose non-negativity requirements on time and quantity variables. Constraints 5.41 puts binary restrictions on the binary FOB variable. Constraints (5.43) impose binary restrictions on the d_{ijvh} variables, while (5.44) impose non-negativity restrictions on the w_{ijvh}

5.2 Linearization

Linearization of objective function (5.1)

The following term in the objective function needs to be linearized:

$$\sum_{h \in H_v} D_{vh}\, T^S_{ijvh} w_{ijvh}\, y_{ijv}$$

This is done by introducing the variable

$$z_{ijvh} = w_{ijvh}\, y_{ijv} \qquad \forall\, v \in V,\ (i,j) \in A_v,\ h \in H_v$$

and adding the following constraints to the model:

$$z_{ijvh} \leq y_{ijv} \qquad \forall\, v \in V,\ (i,j) \in A_v,\ h \in H_v$$

$$0 \leq w_{ijvh} - z_{ijvh} \leq (1 - y_{ijv}) \qquad \forall\, v \in V,\ (i,j) \in A_v,\ h \in H_v$$

Linearization of time balance constraints where end node is a pickup node (5.10)

$$x_{ijv}(t^E_{iv} + \sum_{h \in H_v} T^S_{ijvh} w_{ijvh} + T^{CD}_{jv}\, y_{ijv}) \leq t^S_{jv} \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^P_v$$

The constraint can be linearized by using a big M coefficient as follows

$$t^E_{iv} + \sum_{h \in H_v} T^S_{ijvh} w_{ijvh} + T^{CD}_{jv}\, y_{ijv} \leq t^S_{jv} + M^{5.10}_{ijv}\,(1 - x_{ijv})\ \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^P_v \tag{5.45}$$

When x_{ijv} is equal to zero, y_{ijv} will be forced to be zero. t^E_{iv} is at most equal to the sum of the upper time limit for operation start at the delivery node and the time it takes to unload a quantity equal to the the upper limit for delivery at the node. Hence,

$$M^{5.10}_{ijv} = T^{MAX}_i + \frac{Q^{MAX}_i}{T^Q_{iv}} + \max_{h \in H_v}\{T^S_{ijvh}\}\ \forall\, v \in V,\ (i,j) \in A_v \mid\ i \in N^D_v \tag{5.46}$$
$$, \ j \in N^P_v$$

$$M^{5.10}_{ijv} = \max_{h \in H_v}\{T^S_{ijvh}\} \qquad \forall\, v \in V,\ (i,j) \in A_v \mid i \in \{o(v)\} \tag{5.47}$$
$$, \ j \in N^P_v$$

We add 5.45 to our model instead of 5.10

Linearization of time balance constraints where end node is a delivery node (5.11)

$$x_{ijv}(t^E_{iv} + \sum_{h \in H_v} T^S_{ijvh} w_{ijvh}) \leq t^S_{jv} \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^D_v$$

By using a big M coefficient, we can linearize (5.11) as follows

$$t^E_{iv} + \sum_{h \in H_v} T^S_{ijvh} w_{ijvh} \leq t^S_{jv} + M^{5.11}_{ijv}(1 - x_{ijv})\ \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^D_v \tag{5.48}$$

The time of operation end at node i is at most equal to the sum of the upper time limit of operation start and the time it takes to load/unload maximum allowed quantity at the same node.

$$M^{5.11}_{ijv} = T^{MAX}_i + Q^{MAX}_i T^Q_{iv} + \max_{h \in H_v}\{T^S_{ijvh}\} \quad \forall\, v \in V, \tag{5.49}$$
$$(i,j) \in A_v \mid j \in N^D_v,\ i \in N_v \setminus \{o(v)\}$$

$$M^{5.11}_{ijv} = \max_{h \in H_v}\{T^S_{ijvh}\} \qquad \forall\, v \in V, \tag{5.50}$$
$$(i,j) \in A_v \mid j \in N^D_v,\ i \in \{o(v)\}$$

We add 5.48 to our model instead of 5.11

Linearization of load management constraint where end node is a pickup node (5.13)

$$x_{ijv}(l_{iv} - B_v\,(t^S_{jv} - t^E_{iv})\,(1 - y_{ijv}) + q_{jv} - l_{jv}) = 0 \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^P_v$$

Implicitly we want that:

$$t^S_{jv}\,(1 - y_{ijv}) = c^S_{ijv} \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^P_v \tag{5.51}$$

$$t^E_{iv}\,(1 - y_{ijv}) = c^E_{ijv} \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^P_v \tag{5.52}$$

$$l_{iv} - B_v\,(c^S_{ijv} - c^E_{ijv}) + q_{jv} - l_{jv} \le M^{5.13,1}_{ijv}\,(1 - x_{ijv}) \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^P_v \tag{5.53}$$

$$-\,l_{iv} + B_v\,(c^S_{ijv} - c^E_{ijv}) - q_{jv} + l_{jv} \le M^{5.13,2}_{ijv}(1 - x_{ijv}) \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^P_v \tag{5.54}$$

We achieve this by adding 5.54 and 5.55 to the model, and by bounding the new variables c^S_{ijv} and c^E_{ijv} correctly by requiring that:

$$c^S_{ijv} \le T^{MAX}_j\,(1 - y_{ijv}) \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^P_v \tag{5.55}$$

$$0 \le t^S_{jv} - c^S_{ijv} \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^P_v \tag{5.56}$$

$$t^S_{jv} - c^S_{ijv} \le T^{MAX}_j\,y_{ijv} \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^P_v \tag{5.57}$$

$$c^E_{ijv} \le (T^{MAX}_i + \frac{Q^{MAX}_i}{T^Q_{iv}})\,(1 - y_{ijv}) \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^P_v \tag{5.58}$$

$$0 \le t^E_{iv} - c^E_{ijv} \qquad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N^P_v \tag{5.59}$$

$$t^E_{iv} - c^E_{ijv} \le (T^{MAX}_i + \frac{Q^{MAX}_i}{T^Q_{iv}})\,y_{ijv} \qquad \forall\, v \in V, \tag{5.60}$$

$$(i,j) \in A_v \quad | \, j \in N_v^P$$

Now, we need to set the values for big M coefficients:

For $M_{ijv}^{5.13,1}$ we have that $l_{iv} \leq max\{0, V_v^{CAP} - Q_i^{MIN}\}$ and the rest of the terms are non-positive as $q_{jv} \leq l_{jv}$. Thus $M_{ijv}^{5.13,1}$ becomes:

$$M_{ijv}^{5.13,1} = max\{0, V_v^{CAP} - Q_i^{MIN}\} + B_v(T_i^{MAX} + \frac{Q_i^{MAX}}{T_{iv}^{Q}}) \quad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N_v^P, i \in N_v \setminus \{o(v)\} \tag{5.61}$$

$$M_{ijv}^{5.13,1} = V_v^{CAP} \quad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N_v^P, i \in \{o(v)\} \tag{5.62}$$

For $M_{jv}^{5.13,2}$ we have that the second term is always less than $B_v\, T_j^{MAX}$, and we also have that $l_{jv} \leq V_v^{CAP}$. Thus $M_{jv}^{5.13,2}$ becomes:

$$M_{jv}^{5.13,2} = V_v^{CAP} + B_v\, T_j^{MAX} \quad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N_v^P \tag{5.63}$$

We add 5.54 - 5.60 to our model instead of 5.13

Linearization of load management constraint where end node is a delivery node (5.14)

$$x_{ijv}(l_{iv} - B_v\,(t_{jv}^S - t_{iv}^E) - q_{jv} - l_{jv}) = 0 \quad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N_v^D$$

This constraint is linearized by adding the following two constraints:

$$l_{iv} - B_v\,(t_{jv}^S - t_{iv}^E) - q_{jv} - l_{jv} \leq M_{iv}^{5.14,1}(1 - x_{ijv}) \quad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N_v^D \tag{5.64}$$

$$-l_{iv} + B_v\,(t_{jv}^S - t_{iv}^E) + q_{jv} + l_{jv} \leq M_{jv}^{5.14,2}(1 - x_{ijv}) \quad \forall\, v \in V,\ (i,j) \in A_v \mid j \in N_v^D \tag{5.65}$$

To determine $M_{iv}^{5.14,1}$ we use that $l_{iv} \leq V_v^{CAP}$ and $t_{iv}^E \leq T_i^{MAX} + \frac{Q_i^{MAX}}{T_{iv}^Q}$. The rest of the variables on the left side of (5.64) show up with negative signs and have 0 as lower bound.

The big M coefficients are calculated as:

$$M_{iv}^{5.14,1} = V_v^{CAP} + B_v(T_i^{MAX} + \frac{Q_i^{MAX}}{T_{iv}^Q}) \qquad \forall\, v \in V,\, (i,j) \in A_v \mid j \in N_v^D,\, i \in N_v \setminus \{o(v)\} \tag{5.66}$$

$$M_{iv}^{5.14,1} = V_v^{CAP} \qquad \forall\, v \in V,\, (i,j) \in A_v \mid j \in N_v^D,\, i \in \{o(v)\} \tag{5.67}$$

For (5.65) we have that the sum of the delivered quantity and the leaving quantity cannot exceed the vessel's capacity. Also, the vessel will have had to boil off some gas on its way to the delivery port. So $q_{jv} + l_{jv} \leq V_v^{CAP} - B_v \min_{i \in N_v, h \in H_v} \{T_{ijvh}^S\}$ and $t_{jv}^S \leq T_j^{MAX}$.

The big M coefficients are calculated as:

$$M_{jv}^{5.14,2} = V_v^{CAP} - B_v \min_{i \in N_v, h \in H_v} \{T_{ijvh}^S\} + B_v\, T_j^{MAX} \;\; \forall\, v \in V,\, j \in N_v^D \tag{5.68}$$

We add 5.64 and 5.65 to our model instead of 5.14

Linearization of stopover constraint (5.8)

$$\sum_{v \in V} \sum_{i \in N_{jv}^P} \sum_{k \in N_v^D} x_{ikv}\, x_{kjv} = (1 - a_j) \quad \forall\, j \in N^D$$

Implicitly we want that:

$$\sum_{i \in N_{jv}^P \cap N_{kv}^P} x_{ikv}\, x_{kjv} = b_{kjv} \qquad \forall\, v \in V,\, k \in N_v^D,\, j \in N_v^D \tag{5.69}$$

$$\sum_{v \in V} \sum_{k \in N_v^D} b_{kjv} = (1 - a_j) \qquad \forall\, j \in N^D \tag{5.70}$$

We accomplish this by by adding 5.70 to the model, and by bounding the new variables b_{kjv} correctly by requiring that:

$$b_{kjv} \leq \sum_{i \in N_{jv}^P \cap N_{kv}^P} x_{ikv} \qquad \forall\, v \in V,\, k \in N_v^D,\, j \in N_v^D \tag{5.71}$$

$$b_{kjv} \leq x_{kjv} \qquad \forall\, v \in V,\, k \in N_v^D,\, j \in N_v^D \tag{5.72}$$

$$b_{kjv} \geq \sum_{i \in N_{jv}^P \cap N_{kv}^P} x_{ikv} + x_{kjv} - 1 \qquad \forall\, v \in V,\, k \in N_v^D,\, j \in N_v^D \tag{5.73}$$

We add 5.70 - 5.73 to our model instead of 5.8

Linearization of the constraint making sure that leaving quantity is zero if there is no heel on-board (5.15)

$$l_{iv} \sum_{j \in N_v^P} y_{ijv} \leq 0 \qquad \forall\, v \in V,\, i \in \{N_v^D, o(v)\}$$

$$l_{iv} \leq M_{iv}^{5.15}(1 - \sum_{j \in N_v^P} y_{ijv}) \qquad \forall\, v \in V,\, i \in \{N_v^D, o(v)\} \tag{5.74}$$

The big M coefficient is chosen as small as possible without constraining l_{iv} if $\sum_{j \in N_v^P} y_{ijv}$ is equal to one. When leaving a delivery node i, a vessel v will at most contain maximum capacity minus unloaded quantity at the same node. However, vessel v does not have unloaded quantity at the origin node and the vessel can thus at most contain maximum vessel capacity. Hence, the big M coefficient can be calculated as,

$$M_{iv}^{5.15} = max\{0, V_v^{Cap} - Q_i^{Min}\} \quad \forall\, v \in V,\, i \in N_v^D \tag{5.75}$$

$$M_{iv}^{5.15} = V_v^{Cap} \qquad \forall\, v \in V, i \in \{o(v)\} \tag{5.76}$$

We add 5.74 to our model instead of 5.15

Linearization of the constraint making sure that leaving quantity is zero if there is no leg (5.21)

$$l_{iv}(1 - \sum_{j \in N_v} x_{ijv}) \leq 0 \qquad \forall\, v \in V,\, i \in N_v \setminus \{d(v), o(v)\}$$

$$l_{iv} \leq M_i^{5.21} \sum_{j \in N_v} x_{ijv} \qquad \forall\, v \in V,\, i \in N_v \setminus \{d(v), o(v)\} \tag{5.77}$$

The big M's become

$$M_i^{5.21} = max\{0, V_v^{Cap} - Q_i^{Min}\} \quad \forall\, v \in V,\, i \in N_v^D \tag{5.78}$$

$$M_i^{5.21} = V_v^{Cap} \qquad \forall\, v \in V,\, i \in N_v^P \tag{5.79}$$

We add 5.77 to our model instead of 5.21

Linearization of heel management constraint (5.17)

$$(1 - \sum_{j \in N_v^P} y_{ijv}) \sum_{j \in N_v^P} x_{ijv}((t_{jv}^S - t_{iv}^E)B_v + Q_v^S) \leq l_{iv} \quad \forall\, v \in V,\, i \in \{N_v^D, o(v)\}$$

We may reformulate this to:

$$B_v \sum_{j \in N_v^P} (t_{jv}^S x_{ijv} - t_{iv}^E x_{ijv}) + \sum_{j \in N_v^P} Q_v^S x_{ijv} \leq l_{iv} + M_i^{5.17} \sum_{j \in N_v^P} y_{ijv} \quad \forall\, v \in V,\, i \in \{N_v^D, o(v)\} \tag{5.80}$$

Implicitly we want that:

$$t_{jv}^S x_{ijv} = d_{ijv}^S \quad \forall\, v \in V,\, i \in \{N_v^D, o(v)\},\, j \in N_v^P \tag{5.81}$$

$$t_{iv}^E x_{ijv} = d_{ijv}^E \quad \forall\, v \in V,\, i \in \{N_v^D, o(v)\},\, j \in N_v^P \tag{5.82}$$

$$B_v \sum_{j \in N_v^P} (d_{ijv}^S - d_{ijv}^E) + \sum_{j \in N_v^P} Q_v^S x_{ijv} \leq l_{iv} + M_i^{5.17} \sum_{j \in N_v^P} y_{ijv} \quad \forall\, v \in V,\, i \in \{N_v^D, o(v)\} \tag{5.83}$$

We accomplish this by adding 5.83 to the model, and by bounding the new variables d_{ijv}^S and d_{ijv}^E correctly by requiring that:

$$d_{ijv}^S \leq T_j^{MAX} x_{ijv} \quad \forall\, v \in V,\, i \in \{N_v^D, o(v)\},\, j \in N_v^P \tag{5.84}$$

$$0 \leq t_{jv}^S - d_{ijv}^S \quad \forall\, v \in V,\, i \in \{N_v^D, o(v)\},\, j \in N_v^P \tag{5.85}$$

$$t_{jv}^S - d_{ijv}^S \leq T_j^{MAX}(1 - x_{ijv}) \quad \forall\, v \in V,\, i \in \{N_v^D, o(v)\},\, j \in N_v^P \tag{5.86}$$

$$d_{ijv}^E = 0 \quad \forall\, v \in V,\, i \in \{o(v)\},\, j \in N_v^P \tag{5.87}$$

$$d^E_{ijv} \leq (T^{MAX}_i + \frac{Q^{MAX}_i}{T^Q_{iv}})x_{ijv} \qquad \forall\, v \in V,\ i \in \{N^D_v, o(v)\},\ j \in N^P_v \tag{5.88}$$

$$0 \leq t^E_{iv} - d^E_{ijv} \qquad \forall\, v \in V,\ i \in \{N^D_v, o(v)\},\ j \in N^P_v \tag{5.89}$$

$$t^E_{iv} - d^E_{ijv} \leq (T^{MAX}_i + \frac{Q^{MAX}_i}{T^Q_{iv}})(1 - x_{ijv}) \qquad \forall\, v \in V,\ i \in \{N^D_v, o(v)\},\ j \in N^P_v \tag{5.90}$$

In order to determine $M^{5.17}_i$ we use that $t^S_{jv} \leq T^{MAX}_j$.

Hence, the big M coefficient becomes:

$$M^{5.17}_i = B_v \max_{j \in N^P_v}\{T^{MAX}_j\} + Q^S_v \qquad \forall\, v \in V,\ i \in \{N^D_v, o(v)\} \tag{5.91}$$

We add 5.83 - 5.90 to our model instead of 5.17

5.3 Variable Reduction

As the model's complexity increases, it is important to tighten and reduce the problem size. This can be done by omitting variables that, if they are assigned other values than zero, produce solutions that are either infeasible or dominated by other solutions. Some of these reductions are listed below.

- Arcs between i and j are not created if $i = j$, since it is never beneficial to pick up a cargo from one node and deliver it to the same node. A node here can be a pick-up or delivery node. This reduction applies for the variables x_{ijv}, w_{ijvh} and d_{ijvh}.
- Assume vessel v leaves pick-up node i at time T^{MIN}_i and sails to delivery node j at shortest possible sailing time, hence using the vessel's maximum speed mode. If vessel v cannot arrive at node j within T^{MAX}_j, it will never satisfy the time window for node j. Consequently, all arcs between these two nodes that belong to that vessel cannot be included in an optimal, neither feasible, solution. This applies for arcs from a pick-up node to delivery node, delivery node to pick-up node, and between two different delivery nodes. Hence, if either 5.92 or 5.94 is satisfied, the associated arcs represented by the variables x_{ijv}, q_{iv}, l_{iv}, t^S_{iv}, t^E_{iv} and $\forall h \in H$ w_{ijvh} and d_{ijvh} can be omitted. However, if 5.93 is satisfied, one has to include y_{ijv} in addition to the aforementioned

variables.

$$T_i^{MIN} + \min_{h \in H}\{T_{ijvh}^S\} > T_j^{MAX} \quad \forall \quad i \in N^P, j \in N^D, v \in V \tag{5.92}$$

$$T_i^{MIN} + \min_{h \in H}\{T_{ijvh}^S\} > T_j^{MAX} \quad \forall \quad i \in N^D, j \in N^P, v \in V \tag{5.93}$$

$$T_i^{MIN} + \min_{h \in H}\{T_{ijvh}^S\} > T_j^{MAX} \quad \forall \quad i \in N^D, j \in N^D, v \in V \tag{5.94}$$

- Assume a vessel v with capacity V_v^{CAP}, pick-up node i and delivery node j with quantity window $[Q_j^{MIN}$,$Q_j^{MAX}]$. In order to satisfy the quantity constraints for node j, represented in Constraints 5.18, the vessel must have at least Q_j^{MIN} and enough quantity to be boiled off during the voyage from i to j. Hence, variables representing the arcs between i and j by vessel v that satisfy 5.95, can be discarded. These variables are x_{ijv}, q_{iv}, l_{iv}, t_{iv}^S, t_{iv}^E and $\forall h \in H$ w_{ijvh} and d_{ijvh}

$$V_{cap} - B_v \min_{h \in H}\{T_{ijvh}^S\} < Q_j^{MIN} \tag{5.95}$$

- The same capacity condition as in 5.95 applies also between two delivery nodes; however, the condition has to be modified to account for the quantity window of two delivery nodes instead of only one. The condition for discarding arcs between delivery node j and k for vessel v due to the constraint of quantity windows becomes

$$V_{cap} - (Q_j^{MIN} + Q_k^{MIN} + B_v \min_{h \in H}\{T_{ijvh}^S\} + B_v \min_{h \in H}\{T_{jkvh}^S\}) < Q_j^{MIN} \tag{5.96}$$

- Let j ands k denote two delivery nodes. A vessel v can only deliver a quantity to j and k on the same voyage (stopover) if this quantity is picked up from a node and by a vessel that is approved by both customers in j and k. Hence, if

$$N_{jv}^P \cap N_{kv}^P = \emptyset \tag{5.97}$$

  all arcs between node j and k for vessel v can be discarded as well.

- Assume that vessel v has five speed modes and it has to sail at least at the second highest speed mode from node i to j in order to satisfy the time window of node j, i.e. arrive between T_j^{MIN} and T_j^{MAX}. In this case,

variables representing the three lowest speed modes, $w_{ijvh} \forall h \in 1, 2, 3$ are redundant and can be discarded since they will never be part of a feasible solution. Additionally, the variables which connect the first three speed modes, d_{ijv1} and d_{ijv2}, become unnecessary. If the speed needed to satisfy the time window of j is a linear combination of, for instance, w_{ijv3} and w_{ijv4}, only w_{ijv1}, w_{ijv2} and d_{ijv1} can be discarded.

- The precedence requirement between node i and j is usually taken care of by the time constraints 5.10 and 5.11. However, the variable z_{ij} is not covered by these constraints, hence z_{ij} should not be created if the precedence constraint between the associated pair of nodes is violated. This is handled by setting the sailing time T_{ij}^{CH} of a chartered vessel to zero where the precedence constraint is violated. When the variable is created, it is then only created if $T_{ij}^{CH} > 0$.

6. Adaptive Large Neighborhood Search

This chapter introduces an adaptive large neighborhood search heuristic applied to the short term LNG routing and scheduling problem. The ALNS methodology was first presented by Ropke and Pisinger (2006), extending the large neighborhood search framework proposed by Shaw (1997). The ALNS uses a set of destroy and repair operators to repeatedly break down parts of a solution and re-build them. The method is adaptive in the sense that the probability of selecting an operator is based on its prior performance. ALNS has proven to be successful for a wide range of routing problems.

The solutions method described in this chapter is a *partial optimization method* that uses an ALNS framework in the first part and solves a MIP in the second part with the routes from a solution in the first part fixed. In the first part, decisions related to heel-out, FOB-sale, vessel speed and delivered quantity to customers are fixed. These are the decisions that are not directly related to the routing of the vessels, but to the delivered quantity and speed decisions along the routes, which are difficult to handle in a neighborhood search based heuristic. This approach is inspired by Coelho et al. (2012), Korsvik and Fagerholt (2010) and Demir et al. (2012). The assumptions in the first part of the solution method about the decisions that are postponed to the second part are listed below:

- Never do a heel-out
- No FOB-sale
- Always deliver the minimum possible quantity (lower limit of quantity window) to customers
- Always sail with maximum speed

Computational testing and rationale behind these assumptions are described in

Section 10.1.

Figure 6.1 shows a simple flow chart of the ALNS heuristic. The first part of the heuristic consist of the ALNS framework that stores all new best feasible solutions found. When a stopping criteria is met, the routes of each of the best feasible solutions from the first part are fixed, and the MIP model in Section 5.1 is solved with fixed x_{ij} variables corresponding to the routes from the solutions from the first part.

Since the second part of the ALNS heuristic is straight forward, the following sections are devoted to explaining the first part. Section 6.1 describes the framework used in the first part of the heuristic, and section 6.2 - 6.11 go into more detail on each element of the framework.

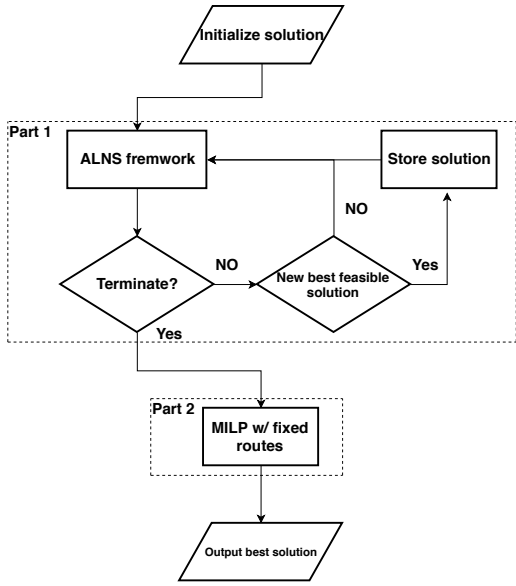


Figure 6.1: A schematic overview of the ALNS heuristic

6.1 The Adaptive Large Neighborhood Search Framework

An algorithmic overview of the ALNS framework is given in Algorithm 1. The input to the heuristic is an initial feasible solution. The representation of the

solution is described in Section 6.2, and the two methods used to find an initial feasible solution are presented in Section 6.3. The iterative search loop in the ALNS is defined in lines 13-45. The search continues until either the number of performed iterations reaches the iteration limit, I^{MAX}, or the running time of the heuristic exceeds the time limit, T^{MAX}. The algorithm iterates between performing local search (LS) operations and large neighborhood search (LNS) operations on the current solution. The algorithm starts by doing a local search on the initial feasible solution to quickly probe if there are better solutions close to the initial solution. The LS component is described in detail in Section 6.8. If the LS component performs I^N iterations without improving the best solution, the algorithm switches to LNS. This components serves to diversify the search, ensuring that the heuristic readily explores the solution space, even if the problem is tightly constrained. If the LNS finds a new best solutions, it is likely that the algorithm has reached a promising region of the solution space, so local search is performed on the solution to ensure an intensified search around the found solution. Moreover, if no improvement is made on the best solution during I^S iterations of LNS, the algorithm switches to doing local search.

The LNS component finds new solutions by exploring large neighborhoods around the current solution through repeatedly destroying and repairing the solution. The destruction and repairing process, and therefore the choice of neighborhood, is controlled dynamically based on the prior performance of the neighborhood, making the search process adaptive. First, a destroy operator, d, is selected from the set of destroy operators, Q^-, and a repair operator, r, is selected from the set of repair operators, Q^+, based on the weights $w_{d,j}$ and $w_{r,j}$, and applied to the current solution S. A detailed explanation of the different destroy and repair methods is given in Section 6.6 and 6.7 respectively, and how the weights are used to select operators is described in Section 6.9. The weights are updated in lines 36-42 based on scores awarded to the operators used in each iteration (line 35) depending on the performance of the operators. The weight adjustment mechanisms are detailed in Section 6.10. The evaluation of a temporary solution, S^t, is done in lines 18-24 for the LS component and in lines 28-34 for the LNS component. Here, $\mathrm{F}(\cdot)$ denotes the evaluation function, which is elaborated in Section 6.5. If a new best solution is found, the solution is added to the list of best solutions. The heuristic can accept non-improving solutions (lines 22 and 32) based on a simulated annealing scheme presented in Section 6.11.

Algorithm 1 Adaptive Large Neighborhood Search

Input: initial solution, S^0
Output: final solution, S^f

Initialize: $w_{d,0}, w_{r,0}$ ▷ Weights
Assign: I^{MAX}, T^{MAX} ▷ Max number of iterations, max running time
Assign: M ▷ Segment size
Assign: I^R ▷ Number of iterations before weights are reset
Assign: U^{INIT}, c ▷ Initial temperature, cooling factor in simulated annealing

$i = 0$, $t = 0$ ▷ Iteration counter, timer
$U = U^{INIT}$ ▷ Temperature in simulated annealing
$S = S^0$, $S^b = S$ ▷ Current solution, best solution

while $i < I^{MAX}, t < T^{MAX}$ **do**
    Select operator type
    if Local Search **then**
        Randomly select local search operator L
        $S^t \leftarrow L(S)$
        if $F(S^t) < F(S^b)$ and S^t is feasible **then**
            $S^b = S^t$
            $S = S^t$
            Add S^t to list of best solutions
        else if $accept(S^t, S)$ **then**
            $S = S^t$
        end if
    else if Large Neighborhood Search **then**
        Select destroy and repair operators $d \in Q^-, r \in Q^+$ using $w_{d,j}, w_{r,j}$
        $S^t = r(d(S))$
        if $F(S^t) < F(S^b)$ and S^t is feasible **then**
            $S^b = S^t$
            $S = S^t$
            Add S^t to list of best solutions
        else if $accept(S^t, S)$ **then**
            $S = S^t$
        end if
        Award scores $\sigma_1, \sigma_2, \sigma_3, \sigma_4$
        if $i\%M = 0$ **then** $j = j + 1$
            if $i\%I^R = 0$ **then**
                Reset weights $w_{d,j}, w_{r,j}$
            else
                Update weights $w_{d,j}, w_{r,j}$ ▷ Section 6.10
            end if
        end if
    end if
    $U = cU, i = i + 1$, update t
end while
return S^b

6.2 Solution Representation

This section introduces the solution representation used in the ALNS heuristic. Notation from Chapter 5 is used.

A solution consists of two parts: a list of routes, r, and a list of unserviced nodes, u. The list of routes contains routes r_v for each vessel $v \in V$. Each route r_v is comprised of an chronologically ordered sequence of nodes, n_i, visited by vessel v. As in Chapter 5, the nodes are divided into pickup nodes, N^P, and delivery nodes, N^D, and the delivery nodes are further categorized as either fixed contract nodes, N^C, or optional contract nodes, N^O. The list of unserviced nodes contains all nodes, both pickup and delivery nodes, that are not included in any of the vessels routes r_v. The first node in each route r_v is an artificial origin node, n_v^0, corresponding to the origin position of vessel v. The artificial destination node is not included in the node sequence. An illustration of the solution representation is presented in Figure 6.2. .

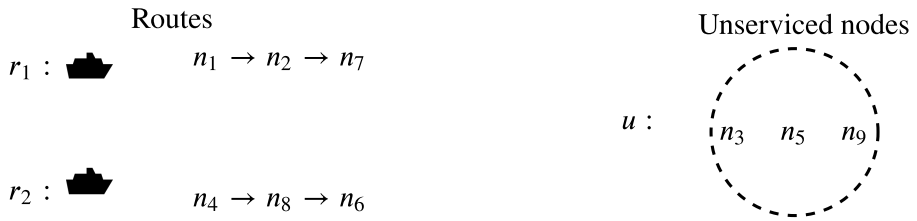


Figure 6.2: Solution representation

6.3 Initial solution

Two different methods of generating an initial solution is used in this thesis. One is based on the original ADP plan, while the other create the initial solution from scratch using a construction heuristic. Both of these methods are explained in detail in Section 7.1.

6.4 Search Space and Feasibility

The ALNS is allowed to restrictively search in the infeasible space. Although this expands the search space and entails the risk of wasting resources on blind alleys or being trapped in an infeasible optimum, there are some theoretical and empirical

studies that indicate improvement in the search performance, e.g., Yan et al. (2005) show that combining feasible solutions and a restricted set of infeasible solutions offers a significant improvement in the performance of a traditional local search algorithm solving a transportation project evaluation problem. In (Glover and Hao, 2011), the authors study a hard 1-0 optimization problem and show that a search strategy that alternates between feasible and infeasible solutions performs better than neighborhoods that don't. Generally, the global optima lies on the edge of the feasible space, or nearby in case of integer linear problems. Allowing search in the infeasible space enables the algorithm to attack the target region (where the global optima lies) both from the inside and the outside of feasibility. Additionally, this might improve the algorithm's diversification capabilities in case of disjoint feasible regions where infeasible solutions might act like a bridging mechanism that allows jumps and shortcuts to the target region.

There are two feasibility criteria which are allowed to be broken during the search. In the model in Section 5.1 constraints (5.7) and (5.8) allows a vessel to visit two consecutive delivery nodes if is has sufficient capacity to serve both nodes, and if both nodes are allowed to receive gas from the preceding pickup node. In the ALNS, a vessel route r_v is allowed to contain consecutive delivery nodes even if the vessel cannot serve both nodes consecutively and regardless of the origin of the gas. Moreover, a route is allowed to contain consecutive pickup nodes. Consecutive delivery nodes that violate constraints (5.7) and (5.8) and consecutive pickup nodes are penalized as described in Section 6.5.

Initial testing indicated that allowing the vessels to pick up more gas than the allocated volume or allowing vessels to arrive after the time window of a node were detrimental to the performance of the ALNS as it had trouble finding feasible solutions, even when adaptively adjusting the infeasibility penalties.

6.5 Evaluation Function

This section describes the evaluation function $F(S)$ used to assess the quality of a solution S. Some notation from Chapter 5 is used. The evaluation function used in the ALNS is different from the objective function in Section 5.1 with regards to a couple of aspects. First of all, the evaluation function is a cost function, so all revenue related terms have negative values, while cost and penalty terms have positive values. Thus, the lower the value of the evaluation function, the better the solution. Second, some changes occur due to the assumptions outlined in Section 6.1; the terms related to FOB-sale and heel-out are omitted, and the terms related to speed and quantity are simplified as all vessels sail at maximum speed and deliver quantities equal to the lower bound of the quantity window in each node.

Moreover, a penalty term is added to the evaluation function to penalize infeasible solutions, and a scaling factor is included in the penalty for breaching contracts to better control the trade-off between inserting nodes that causes the solution to become infeasible and breaching contracts.

6.5.1 Route Cost

The cost C_v^R related to a vessel route r_v is the difference between the operating costs and the total revenue of the route. Let n_i be the node at position i in the node sequence r_v, where $i \in \{1, \ldots, l_v\}$ and l_v is the number of nodes in route r_v. Vessel v is sailing in the maximum speed mode denoted $\overline{h}$. The route cost is then defined as

$$C_v^R = \sum_{i=1}^{l_v-1} C^{OP}_{n_i n_{i+1} \overline{h}} - \sum_{i|n_i \in N^D \cap r_v} R_{n_i} Q_{n_i}^{MIN} \tag{6.1}$$

where $C^{OP}_{n_i n_{i+1} \overline{h}}$ is the operating cost of sailing from n_i to n_{i+1} with speed mode $\overline{h}$, R_{n_i} is the revenue per unit for node n_i and $Q_{n_i}^{MIN}$ is the lower bound for the quantity window of n_i.

6.5.2 Cost of Unserviced Nodes

Fixed contract nodes that are in the list of unused nodes u can either be chartered or breached. A fixed contract node $u_f inu$ can only be chartered if there exists a pickup node in u that is a feasible pickup node $u_p inu$ for u_f. Let P^{CH} be the set of pairs (u_f, u_p), where $u_f \in N^C \cap u$ and $u_p \in N^P \cap u$, that are used in chartering. The chartering cost, C^H, can then be calculated as

$$C^H = \sum_{(u_f, u_p) \in P^{CH}} C^{CH}_{u_f, u_p} - R_{u_f} Q^{CH}_{u_f} \tag{6.2}$$

where $C^{CH}_{u_f, u_p}$ is the cost of servicing u_f from node u_p by a chartered vessel, R_{u_f} and $Q^{CH}_{u_f}$ is respectively the unit revenue and the the chartered volume for node u_f.

If a fixed contract node cannot be serviced by any pickup node in the unused node list by a charter vessel, the contract corresponding to the node is breached. Let B be the set of all nodes $u_i \in u$ that cannot be chartered. The total breaching cost is then

$$C^B = \sum_{u_i \in B} \beta \Pi_{u_i} \tag{6.3}$$

where Π_{u_i} is the penalty cost of breaching node u_i, introduced in Section 5.1, and $\beta > 0$ is the scaling factor mentioned in the introduction of this section.

6.5.3 Penalty Cost

As stated in Section 6.4, a solution having routes with consecutive pickup nodes or an infeasble consecution of delivery nodes is penalized. Let k_v^P and k_v^D be the number of consecutive pickup nodes and infeasble consecutions of delivery nodes, respectively, in the route r_v for vessel v. The penalty cost for vessel v, C_v^P is defined as

$$C_v^P = \alpha \, \overline{R} \, \overline{V^{CAP}} \, (k_v^P + k_v^D) \tag{6.4}$$

where $\overline{R}$ is the average unit revenue of all fixed contracts, $\overline{V^{CAP}}$ is the average vessel capacity of vessels $v \in V$, and $\alpha > 0$ is a scaling factor.

6.5.4 Total Evaluation Function

Based on Equation 6.1 - 6.4, the total value of the evaluation function $F(S)$ of a solution S containing routes r_v for vessels $v \in V$ and list of unserviced nodes u is calculated as

$$F(S) = C^H + C^B + \sum_{v \in V} C_v^R + C_v^P \tag{6.5}$$

6.5.5 Feasibility

The evaluation function will also check the following feasibility criteria:

1. All nodes n_i in route r_v must be in the set of nodes that can be visited by vessel v, N_v, for all vessels v
2. All vessels v must have sufficient capacity to deliver the minimum quantity to all nodes n_i in route r_v, when boil-off is accounted for.
3. All vessels must be able to arrive before the end of the time window of all nodes n_i in route r_v.

4. The total volume picked up by all vessels in each port in each period must be less than or equal to the corresponding allocated volume, taken into account that allocated volume at a port can be transferred from one period to the subsequent one.

If any of the criteria above is broken in a solution S, the value of the evaluation function is set to ∞. A solution is feasible if it satisfies (1-4) above and has a penalty cost $\sum_{v \in V} C_v^P = 0$

6.6 Destroy Operators

This section presents the different destroy operators used in the ALNS heuristic. All the destroy operators take a solution as input, remove nodes from the routes in the list of routes and adds them to the unserviced nodes list. The main purpose of including several different destroy methods is to be able to destroy a solution in many different ways, thereby potentially explore a variety of large neighborhoods around the solution.

The destroy operators are controlled by the degree of destruction, γ, representing the share of nodes in the solution which is removed in an iteration of the ALNS heuristic. Every iteration y is selected randomly between a minimum and a maximum value, denoted Γ^{MIN} and Γ^{MAX} respectively. As explained in Ropke and Pisinger (2006), the degree of destruction will intuitively have a large impact on the performance of the ALNS. A too small γ can make the ALNS lose the advantage of the large neighborhood search and might render the heuristic unable to sufficiently explore the solution space. An excessively large γ effectively reduce the repair operators to re-construct the solution from scratch in each iteration, which may be time consuming and produce poor results. The number of nodes, q, to be removed by the destroy operators are calculated from the degree of destruction and passed to the operator as input.

Four different destroy operators are presented. The first three are generic destroy operators inspired by Ropke and Pisinger (ibid.). The last one is a problem specific destroy operator that seeks to exploit general knowledge of the problem structure to find promising neighborhoods.

6.6.1 Random Removal

The Random Removal operator removes q randomly selected nodes from routes in the solution and adds them to the list of unserviced nodes. This operator is a special case of the Shaw Removal operator (Section 6.6.2) with $p = 1$, but is

implemented as a separate operator for speed purposes.

6.6.2 Shaw Removal

This removal operator was first introduced by Shaw (1997). The operator is here slightly modified to fit the problem in this thesis. The idea behind the operator is to remove nodes that are related to each other, as this makes it more likely that the nodes can be inserted in new positions when the solution is repaired, thus constructing new, possibly better, solutions. To quantify the similarity between two nodes, n and m, a *relatedness measure* $R(n, m)$ is defined:

$$\begin{aligned} R(n,m) = \nu_1 & \left(\frac{|T_n^{MIN} - T_m^{MIN}|}{\max\limits_{i \in N} T_i^{MIN} - \min\limits_{i \in N} T_i^{MIN}} + \frac{|T_n^{MAX} - T_m^{MAX}|}{\max\limits_{i \in N} T_i^{MAX} - \min\limits_{i \in N} T_i^{MAX}} \right) \\ & + \nu_2 \frac{D_{nm}}{\max\limits_{i,j \in N} D_{nm}} + \nu_3 \left(1 - \frac{|V_n \cap V_m|}{\min\{|V_n|, |V_m|\}} \right) \end{aligned} \tag{6.6}$$

where T_i^{MIN} and T_i^{MAX} is the lower and upper limit of the time window for node i, D_{ij} is the distance between node i and j, V_i is the set of vessel that can visit node i and N is the set of all nodes in the solution. The relatedness measure consist of three normalized terms, weighted by $\nu_1, \nu_2, \nu_3 \in (0, 1]$. The first term measures the relatedness in time, the second term measures the distance between the nodes and the third term ensures that nodes get a high relatedness of only a few vessels can visit them. A lower value of $R(n, m)$ means that two nodes are more related.

Algorithm 2 shows the removal procedure of the Shaw Removal operator. q nodes are removed from the list of routes r in solution S based in their relatedness measure. A determinism parameter $p \geq 1$ is used to introduce randomness into the removal process. A larger value of p means less randomization.

Algorithm 2 Shaw Removal

Input: solution S with routes r and unserviced nodes list u, q nodes to remove, parameter $p \geq 1$

1: n = randomly selected node from r
2: $D = \{r\}$
3: **while** $|D| < q$ **do**
4: m = randomly selected node from D
5: L = set of nodes $\in r \setminus D$
6: sort L such that $i < j \Rightarrow R(n, L[i]) < R(n, L[j])$
7: y = random number $\in [0, 1)$
8: $D = D \cap \{L[y^p|L|]\}$
9: **end while**
10: remove all nodes in D from r

6.6.3 Worst Removal

The Worst Removal operator removes the nodes that appears to be the worst parts of the solutions, seemingly placed in the wrong position. The *cost* $C(n, S)$ of a node n in a solution S is defined as:

$$C(n, S) = F(S) - F(S^{-n}) \tag{6.7}$$

where F is the evaluation function (Section 6.5) and S^{-n} is defined as the solution S without node n. Note that in S^{-n}, the node is not removed to the unserviced nodes list, so the node is not breached if it is a fixed contract node. The operator seeks to remove the nodes with the highest cost $C(n, S)$. As in the Shaw Removal, a parameter $p \geq 1$ introduces randomness to the removal procedure, as shown in Algorithm 3

Algorithm 3 Worst Removal

Input: solution S with routes r and unserviced nodes list u, q nodes to remove, parameter $p \geq 1$

```
i = 0                                                    ▷ counter
while i < q do
    L = set of nodes ∈ r
    sort L such that i < j ⇒ C(L[i], S) > C(L[j], S)
    y = random number ∈ [0, 1)
    n = L[y^p |L|]
    remove node n from r
    i = i + 1
end while
```

6.6.4 Vessel Removal

The Vessel Removal operator removes nodes in similar positions in the routes of vessels with similar capacities. This removal operator is motivated by the observation that the difference between the routes of good and excellent solutions are often shuffles of segments of routes between vessels. Given the degree of destruction *Gamma* and number of nodes to be destroyed q, the operators starts by selecting a random vessel v (with corresponding route r_v with l_v number of nodes) and removes a segment consisting of $m = \text{ceil}(l_v\Gamma)$ nodes from a random position in the node sequence r_v before position $l_v - m$. The vessel not yet selected with the closest capacity to vessel v is then selected and a segment of up to (less if the end of the route is reached) m nodes are removed from the vessel at the same position as in r_v. This process is repeated until q nodes are removed from the solution.

6.7 Repair Operators

In this section, the repair heuristics used to repair a destroyed solution in the ALNS are presented. These heuristics try to insert nodes from the unserviced nodes list into the routes, r_v. Insertion heuristics in vehicle routing related problems are typically categorized as either *sequential* or *parallel* (Ropke and Pisinger, 2006). Sequential insertion heuristics construct one route at a time, while parallel insertion heuristics build several routes simultaneously. The insertion heuristics presented in this section are inspired by Ropke and Pisinger (ibid.) and are classified as parallel. The repair operators in them self are relatively simple and fast heuristics, and as shown in Ropke and Pisinger (ibid.), they do not necessarily produce good solutions when used to construct a solution from scratch. Still, the results in

Ropke and Pisinger (ibid.) indicate that these inherently imprecise heuristics can find very promising solutions when incorporated in an ALNS framework.

The possibility of serving contracts with chartered vessels is handled by the repair operators. All fixed contract nodes in the list of unserviced nodes, u, that can be serviced by a chartered vessel from any of the pickup nodes in u are paired up. If more than one pairing is possible for a fixed node, the one with the lowest chartering cost is chosen. The repair heuristics insert all nodes in such pairs last.

6.7.1 Basic Greedy Insertion

The simplest repair operator is the Basic Greedy Insertion. This heuristic iterates over all nodes in the list of unserviced nodes and tries to insert them at the cheapest possible position in any vessel route. For diversification, the list of unserviced nodes is shuffled before the start of the insertion process. The best insertion of an unserviced node is determined as follows. Let ΔF_{iv} be the cost added to the evaluation function value of inserting unserviced node i in the set of unserviced nodes u at the position in the route of vessel v that increases the cost the least. If no insertion is possible, set $\Delta F_{iv} = \infty$. Iterating through all vessel routes, the best insertion is determined by

$$\underset{v \in V}{\operatorname{argmin}} \Delta F_{iv} \qquad \forall i \in u \tag{6.8}$$

If the best possible insertion cost is ∞, no insertion is possible and the node remains in the unserviced nodes list until the next iteration of the ALNS.

6.7.2 Deep Greedy Insertion

The Deep Greedy Insertion is an extension of the Basic Greedy Insertion. Instead of trying to insert each unserviced node before moving on to the next, the Deep Greedy Insertion evaluates the insertion cost of every unserviced node in each iteration and inserts the unserviced node with the best insertion cost. After an insertion is performed, all remaining nodes in the unserviced nodes list is looped through again to find the next insertion. This makes the search deeper, which should lead to better insertions, but also makes the algorithm more computationally heavy than the Basic Greedy Insertion.

Let ΔF_{iv} be the cost added to the evaluation function value of inserting unserviced node i in the set of unserviced nodes u at the position in the route of vessel v that

increases the cost the least. If no insertion is possible, set $\Delta F_{iv} = \infty$. Iterating through all unserviced nodes and vessel routes, the best insertion is determined by

$$\underset{i \in u,\, v \in V}{\operatorname{argmin}} \Delta F_{iv} \tag{6.9}$$

If no feasible solution is found in an iteration of the operator, we break out of the algorithm and start a new iteration of destroy and repair.

6.7.3 Regret-k Insertion

The previously described repair operators insert the node with the lowest insertion cost first, which means that the insertion of more troublesome nodes, i.e nodes with high insertion costs, are postponed. However, when the high cost nodes are to be inserted, most of the routes are filled up so possible insertions are far less likely. Regret heuristics aim improve on the greedy insertion operators by considering the effect of not inserting a node at the position with the lowest insertion cost when selecting which node to insert first. Let ΔF_{iv}^k denote the change in the objective value by inserting unserviced node i in the set of unserviced nodes u at its k-th cheapest possible insertion in the vessel v's route. ΔF_{iv}^2 is by this notation the second cheapest possible insertion of node i in vessel route v. If no possible insertions are found, $\Delta F_{iv}^k = \infty$. For $k = 2$, the node to be inserted is decided based on

$$\underset{i \in u,\, v \in V}{\operatorname{argmax}} \left(\Delta F_{iv}^2 - \Delta F_{iv}^1\right) \tag{6.10}$$

Equation 6.10 calculates the difference between the cost of inserting a node at the position with the second lowest insertion cost and the lowest insertion cost. For a node with several viable insertion possibilities, this value is low, so the node is inserted later. A node with a large value of this difference have limited insertion possibilities, so it is better to insert the node early. The generalization of the regret heuristic to consider the k lowest cost insertions of node i is shown below:

$$\underset{i \in u,\, v \in V}{\operatorname{argmax}} \sum_{c=2}^{k} \Delta F_{iv}^c - \Delta F_{iv}^1 \tag{6.11}$$

Compared to a regret heuristic with $k = 2$, Regret-k heuristics take more look-ahead information into account and are therefore better at avoiding problems with

inserting high cost nodes, or ending up with no possible insertions a node. However, for large ks, the heuristic can become too conservative and overlook good insertion possibilities. A larger k will also make the operator more computationally heavy. The regret operators used in this thesis is Regret-2 and Regret-3.

6.8 Local Search Operators

This section presents the local search operators used in the LS component in the ALNS framework. The component uses tabu seach as its overall search strategy, with full enumeration of the neighborhood of one LS operator in each iteration. The list of unused nodes is shuffled before the use of an LS operator.

6.8.1 Ejection Swap

The Ejection Swap intends to swap a node from the unserviced nodes list with a node in a vessel route. Pickup nodes and delivery nodes can only be swapped with a node of similar type, as this was proven most effective during testing.

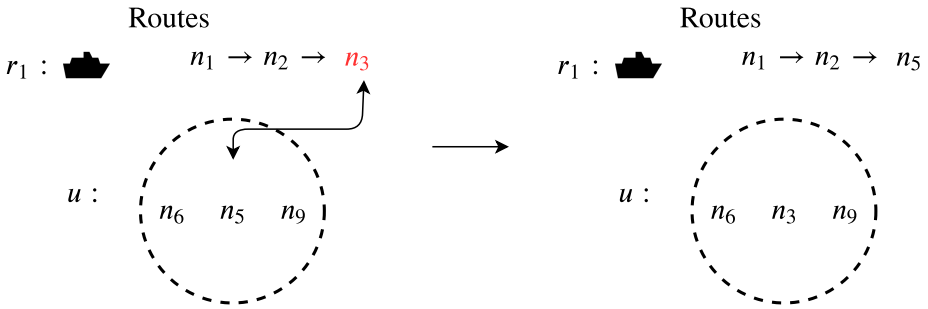


Figure 6.3: Illustration of Ejection Swap operator

6.8.2 Inter Swap

The Inter Swap operator tries to swap two nodes between the routes of two randomly selected vessels. Pickup nodes and delivery nodes can only be swapped with a node of similar type, as this was proven most effective during initial testing.

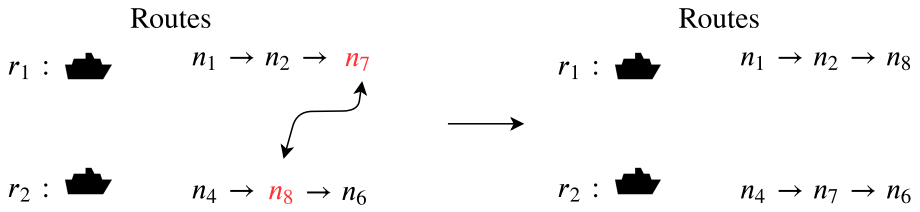


Figure 6.4: An illustration of Inter Swap operator

6.8.3 Re-assign

The Reassign operator is inspired by a local search operator used in Korsvik, Fagerholt, and Laporte (2011). The Reassign algorithm consists of two steps for each vessel route. The first step is to try to swap any of the nodes in the vessel route with any of the nodes in the unserviced nodes list. If such a swap is possible, the algorithm performs the best swap and tries to insert the node that was removed from the vessel route into a route of another vessel, i.e re-assigning the node. If an insertion is possible, the node is inserted into the best possible position in any of the other vessel's routes. If no insertion is possible, but the solution was improved by doing the swap in the first step, this swap is done so that the node is added to the unserviced nodes list. This case is the same as the Ejection Swap.

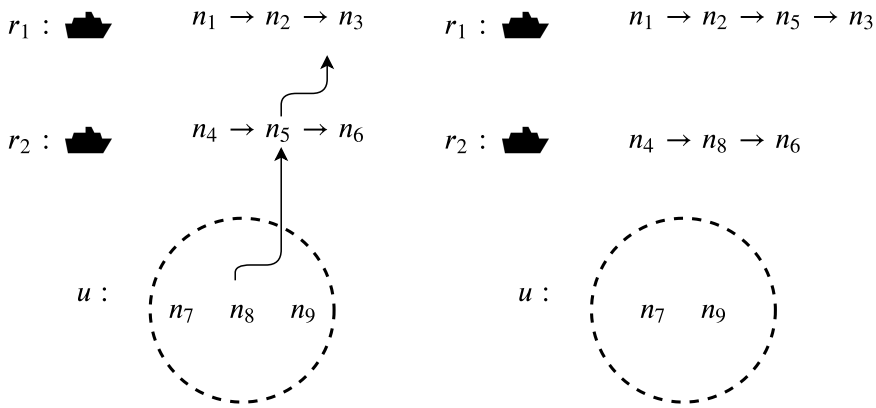


Figure 6.5: An illustration of Re-assign operator

6.8.4 Tabu list

To prevent the local search from cycling and getting trapped in local minima, a tabu list included in the search. The tabu list includes the L last swap moves done by the local search component. The tabu list only applies to the Ejection Swap and the Inter Swap operators, or to the Re-assign operator if it ends up doing an ejection swap. A neighbor that is reached during the use of any of these operators by a move in the tabu list, is considered infeasible.

6.8.5 Neighbor Selection Strategy

Two strategies for selecting a neighbor is considered. The *best improvement* strategy select the neighbor that improved the evaluation function of a solution the most. If no neighbor improves the solution, the one that worsen the evaluation function the least is chosen. The other selection strategy is the *first improvement* strategy. With this strategy the first neighbor that is found to improve the evaluation function is selected. Again, if no neighbor improves the solution, the one that worsen the evaluation function the least is chosen.

6.9 Selecting a Destroy and a Repair Operator

To choose which repair and destroy operator to use in each iteration of the ALNS, the roulette wheel selection principle is used, based on weights assigned to each operator. As proposed by Ropke and Pisinger (2006), Equation 6.12 is used to make the selection. By alternating between different destroy and repair methods, a more extensive search of the solution space might be explored, and the heuristic is believed to be more robust overall. The probability of selection a specific operator in a given segment is calculated as:

$$\frac{w_{q,j}}{\sum_{\hat{q} \in Q} w_{\hat{q},j}} \tag{6.12}$$

Where $w_{q,j}$ is the weight assigned to operator q in the set of operators Q in segment j. A destroy and repair operator is selected independently of each other. The process of assigning weights to the operators is described in the following section.

6.10 Adaptive Weight Adjustment

The weight assigned to each operator is adjusted during the run of the ALNS based in the recent performance of the operator. The entire search is divided into segments, where one segment is a number of iterations of the overall ALNS heuristic. To measure how well an operator has performed in a given segment, a score is given to the operator. The higher the score, the better the operator has performed. The score of all operators are set to zero at the beginning of each segment and is subsequently increased by a score adjustment parameter for each iteration in which a specific operator is used. Table 6.1 describes the scoring criteria used to evaluate an operator. These criteria are inspired by Ropke and Pisinger (2006). The values of the score adjustment parameters satisfy the inequalities $\sigma_1 > \sigma_2 > \sigma_3 > \sigma_4 \geq 0$. In each iteration both a destroy and repair operator is used. As there is no way to determine whether the cause of the performance in a single iteration was the destroy operator, the repair operator or the combination of the two, both operators are given the same score adjustment parameter. The score adjustment parameter is then normalized based on the computational time of the operator, before it is added to the overall segment score. The weights of all

Table 6.1: Score adjustment parameters

Score adjustment parameter	Description
σ_1	New global best solution is obtained during the iteration
σ_2	New solution is better than current solution
σ_3	New solution is worse than current, but accepted by SA
σ_4	New solution is worse than current and not accepted by SA

operators are initialized to 1. After each segment of iterations, the weigh of each operator for the next segment is calculated as a weighted average of the weight for the previous segment and the average normalized score in that segment:

$$w_{q,j+1} = w_{q,j}(1-\lambda) + \lambda\frac{\overline{\pi_{q,j}}}{a_{q,j}} \tag{6.13}$$

$w_{q,j+1}$ is the weight in segment $j+1$, $w_{q,j}$ is the weight in segment j, $\pi_{q,j}$ is the accumulated normalized score for segment j, $a_{q,j}$ is the number of times operator q was used in segment j and λ is the decay parameter, $\lambda \in [0,1]$. The decay parameter controls how sensitive the weights are to variations in performance from segment to segment. If λ is 0, the weights are not affected by the performance of the operators. If λ is 1, the weights for the next segment is only affected by the score in the previous one.

The weights are reset to their initial value every I^R iterations to make sure that operators that perform poorly in early segments are not given too small probability of being selected later in the search, where they might perform better.

6.11 Acceptance Criteria - Simulated Annealing

The acceptance criteria used in the ALNS are from simulated annealing. This allows the algorithm to accept non-improving solutions, which can help the ALNS escape local optima. There are numerous different acceptance criteria with this property, but simulated annealing based criteria has been widely used in the ALNS literature, showing good results. In simulated annealing, a solution S^t is accepted with probability

$$e^{-(F(S^t)-F(S))/T_i} \tag{6.14}$$

where S is the current solution and $T_i > 0$ is the temperature in iteration i. An improving solution is always accepted with this criterion. The probability of accepting a non-improving solution is controlled by the temperature parameter, where a higher value of T_i increases the chance of acceptance. The temperature is initially set to U^{INIT} and is decreased in each iteration following an *exponential cooling scheme*, i.e $T_{i+1} = c \cdot T_i$, where $0 < c < 1$ is the *cooling rate*. This way the acceptance criteria get stricter as the search progresses. Setting the initial temperature too high, will help the heuristic with early exploration of the solution space, but if the cool down is too slow, the heuristic might not get a sufficient amount of time to intensify the search in promising areas of the search space, harming the performance. Having a too small initial temperature or cooling down too fast will essentially reduce the heuristic to a descent heuristic, increasing the chance of getting trapped in a local optima.

7. Fix and Optimize Large Neighborhood Search

A natural way of searching for solutions to a problem is to start with an initial solution before improving it iteratively. As discussed earlier, this can generally be achieved by searching through a neighborhood around the current solution, and it is known in the literature as a Large Neighborhood Search (LNS). When such a method is applied using a MIP-solver it is categorized as a matheuristic.
The main purpose of this chapter is to introduce a MIP-based Fix and Optimize (hereby FO) matheuristic, which works by guiding a MIP towards good, feasible solutions through destroying and repairing limited areas of the solution space. First, in Section 7.1 two options for providing heuristics with initial solutions are presented. Second, Section 7.2 starts off with a general overview of the Fix and Optimize heuristic. Third, a detailed description of the heuristic is presented in section 7.2.3. A summary of the related literature within this field of research which can be found in Chapter 3.

7.1 Constructing Initial Solutions

Many heuristics require an initial solution in order to function properly. In this thesis, two heuristics are presented. One of them need to start from an initial solution (the Fix and Optimize, introduced later in this chapter). Later, in Chapter 6 an ALNS will be presented, and for this heuristic starting from an initial solution is optional. In this thesis when the term initial solution is used, it only concerns vessel legs, or routes. Other variable values are not considered important, as long as there exists values for these which are feasible for the complete problem.

Fix and Optimize LNS and the ALNS may start by having a feasible solution as input. Then, the heuristics continue by iterating their way to better solutions. Below, different alternatives of creating feasible initial solutions are described.

7.1.1 Destroying Infeasibilities From ADP Plan

The first alternative is to create an initial solution from an ADP plan, after running a specialised destroy heuristic in order to make it feasible, while changing the input solution as little as possible. The destroy heuristic works by evaluating the sequence of nodes in each route and correcting infeasibilities when they occur. If a given leg is infeasible, the "to-node" is removed from the ADP-plan. If the "to-node" is a pickup node, then the delivery node following this pickup node is also removed to ensure feasibility, as there may not be any other alternative of serving this customer if the pickup node before it is infeasible. In other words, the incoming arc to this delivery node is already removed when the pickup node was removed, but in addition, the heuristic must remove the outgoing arc from the delivery node, making it possible for the vessel to not visit that delivery node. The main benefits of this starting alternative is that it makes it possible to make use of the ADP plan which is believed to have a high probability of being close to the optimal solution. At the same time the heuristic accommodate any disruption in input data by removing legs leading to the infeasibilities. This is believed to be the most time-efficient alternative (i.e. finding good solutions fast). This first approach is quite similar to the second approach in how feasibility calculations are made, but in the second approach the solutions are constructed from scratch. Therefore, only the pseudo-code for the second alternative is included here.

7.1.2 Constructing an Initial Solution

The second alternative is to create an initial solution using a construction heuristic. The construction heuristic is greedy in the sense that it tries to serve the nodes which has the earliest time windows first, without considering those that come later. Also, it chooses a random vessel and constructs a route for the given vessel without considering other vessels which are yet to be given a route, and in this sense it is greedy. It starts by considering the initial state of the first vessel: if it is able to go to a delivery node, the heuristic add the delivery node to the vessel route which has the earliest time window and which is also a feasible destination for the given vessel. The heuristic ignores stopover possibilities (i.e. visiting two delivery nodes in a row) and continues by looking for a new feasible pickup node destination, and so on. The heuristic assumes minimum delivery quantities, maximum sailing speed and no FOB sale and makes exact feasibility feasibility calculations thereafter. Considerations with respect to boil-off calculations, quantity balance at ports, time constraints etc. are included. It is thus easy to satisfy allocated volume constraints at liquefaction ports and heel-out decisions.

The pseudo-code for this heuristic is presented in Algorithm 4. In line 1-3, global

lists and sets are initialized. In line 4, the potential allocated volume is calculated for each port-period combination. Here, "potential" mean that it is assumed that for every period, all quantity left in the given port at an earlier period is transferred to the current period (i.e. an element in this list is a cumulative sum of the volumes in previous periods for the given port). In line 5-30, a vessel is chosen and the route for the given vessel is constructed. Line 6-8 initialize the vessel state at the origin. Line 9 initiates a search for new voyages which continues until a stopping criteria is met. In lines 11-20 and 21-30, the heuristic search for feasible voyages to delivery nodes and pickup nodes, respectively. The search consists of making a list of feasible voyages (lines 12-13 and 22-23). Here, the heuristic takes in an external function called "checkFeas", and the pseudo-code is found in Algorithm 5), before choosing the voyage closest w.r.t time windows (lines 16 and 26). In lines 17 and 27, then decision has been made and the state variables are updated. If there are no feasible voyages (lines 19 and 29), then the while-loop is exited and the current route is added to the solution (line 33). After this has been performed for every vessel, the complete solution is returned (line 35).

The pseudo-code for the "checkFeas" function is found in Algorithm 5. In lines 1-7 the function receives input variables. Lines 8-11 represent the function output, which is later used in the ADP-heuristic in order to destroy infeasible vessel legs, or used in the Construction Heuristic in order to add nodes to a vessel route and in order to make quantity/time calculations. Note that if the output is used in the ADP-heuristic then only the value of B^F is relevant. The rest of the heuristic (lines 12-29) is self-explaining. Here, various feasibility calculations are made in correspondence to the mathematical model in Chapter 5.

7.1.3 Ignoring Infeasibilities

The third alternative is to start from an ADP plan, and ignore any infeasibilities. In other words, the heuristic commences as normal and hopefully a future destruction will lead to a feasible solution. For instance, if an iteration fails to produce a feasible solution after destroying the ADP-plan, then the next iteration will start over by destroying the ADP-plan again.

Testing of these alternatives is included in section 10.4.3

Algorithm 4 Construction heuristic

Input: Problem data
Output: Feasible initial solution (S^0)

$L^v = [\,]$ ▷ List of nodes visited - empty
$S^0 = \{\,\}$ ▷ Solution, set of vessel routes - empty
V ▷ List of vessels - shuffled

Calculate $V_{p,m}$ ▷ Volume left in all port-period combinations
for v in V **do**
    Initialize: $R = [0]$ ▷ Current route for vessel
    Initialize: T^c, Q^c and N^c ▷ Current time, cargo and node for vessel
    Calculate B^d ▷ Boolean - True if next node is a delivery node
    while True **do**
        $D^{feasibleDest} = \{\,\}$ ▷ Set of current feasible destinations for vessel

        if B^d **then** ▷ Going to a delivery node
            for N^d in delivery nodes **do**
                $D^{feasibleDest} \leftarrow$ checkFeas(B^d, N^c, T^c, Q^c, N^d) ▷ See Alg. 5
            end for
            if $D^{feasibleDest}$ is not empty **then**
                $N^c \leftarrow$ Find closest node w.r.t time windows in $D^{feasibleDest}$
                Update: $L^v, V_{p,m}, R, B^d, T^c, Q^c, N^c$
            else
                Exit loop (and remove previous pickup-node)
            end if
        else ▷ Going to a pickup node
            for N^d in delivery nodes **do**
                $D^{feasibleDest} \leftarrow$ checkFeas(B^d, N^c, T^c, Q^c, N^d) ▷ See Alg. 5
            end for
            if $D^{feasibleDest}$ is not empty **then**
                $N^c \leftarrow$ Find closest node w.r.t time windows in $D^{feasibleDest}$
                Update: $L^v, V_{p,m}, R, B^d, T^c, Q^c, N^c$
            else
                Exit loop
            end if
        end if

    end while
    $S^0 \leftarrow R$ ▷ Add vessel route to solution
end for
return S^0

Algorithm 5 Check feasibility of a specific vessel leg (checkFeas)

Input:

B^d ▷ Boolean - True if next node is a delivery node
N^c ▷ Current vessel position
T^c ▷ Current time for vessel at current position
Q^c ▷ Current load on board vessel
N^d ▷ Potential vessel destination
v ▷ Vessel
V^{Port} ▷ Volume left in port for a given month

Output: ▷ Output from this function are needed if it is later decided that the vessel should go to the given node. If node is infeasible, output is "False"

B^F ▷ Boolean - True if next node is feasible
T^d ▷ Current time for vessel after visiting N^t
Q^d ▷ Current cargo for vessel after visiting N^t
V^d ▷ Volume depleted from port - only relevant for pickup port visits

if B^d **then** ▷ Going to a delivery node
    Calculate vessel-port compatibility
    Calculate time compatibility w.r.t time windows
    Calculate vessel unloading quantity-port compatibility
    Calculate T^d and Q^d
else ▷ Going to a pickup node
    if Heel-out required **then**
        Calculate vessel-port compatibility
        Calculate time compatibility w.r.t time windows
        Calculate vessel loading quantity-port compatibility
    else
        Calculate vessel-port compatibility
        Calculate time compatibility w.r.t time windows
        Calculate vessel loading quantity-port compatibility
        Calculate T^d, Q^d and V^d
    end if
end if
return T^d, Q^d, V^d

7.2 Model and Framework

The remaining part of this chapter is concerned with the formulation of a Fix and Optimize (FO) matheuristic. This heuristic works by guiding a MIP-solver in an iterative manner in order to find feasible solutions in a large neighborhood. The neighborhood is defined by fixing a subset of the variables. Parts of the solution space are then restricted while the remaining unfixed variables opens up for exploring a neighborhood. The heuristic repeats these steps in an iterative manner, and seeks to include the benefits of both low computational times and the ability to find new, improving, solutions frequently.

The FO needs to be both fast and effective in searching the solution space. Similar to many other heuristics, the model should have the ability to search large parts of the solution space fast. As pointed out in Lindahl et al. (2018), a crucial aspect of the implementation phase of the FO is that the base model has to be built from scratch only once by the MIP-solver. It is common sense that any changes needed later on during the iterative phase need to be added or removed much more efficiently. One possible way of achieving this is mentioned in Chapter 10. Furthermore, it is important that the repair phase in each iteration is warm-started from the previously found solution. Often, this is automatically being taken care of by the MIP-solver if the solver is not called from an external program.

7.2.1 Model Overview

Before decision variables can be fixed, an initial feasible solution is needed. Therefore, the FO assumes a feasible solution populated with vessel legs at hand. It then searches the solution space by destroying parts of the vessel routes (how much is controlled by a parameter specifying the degree of destruction). It then applies a MIP-solver, attempting to repair the solution for a pre-specified amount of time. During the repair phase, the model works by fixing the remaining vessel legs that were not destroyed. If a new improving solution is found, then this solution is accepted as the current solution. These steps are then repeated iteratively, while destroy and repair parameters are being dynamically adjusted. This continues until a stopping criteria is met. In the following sections, the details of this heuristic is presented. A simple pseudo-code of the start phase of the heuristic is presented in Algorithm 6, while a description of the proposed pseudo-code for the FO is shown in Algorithm 7.

7.2.2 Modeling Decisions and Similarities with an ALNS

Although the Fix and Optimize LNS shares some ideas and methodology with the ALNS, it is named a "Fix and Optimize Large Neighbourhood Search" as there is only a single repair method, the MIP. In the development of a Fix and Optimize LNS model, the work on the ALNS and the work on the extensive MIP is combined into a new model, and the result corresponds to a large neighbourhood search algorithm which works by simply using the MIP as the only repair method. Multiple options for destroy methods are available, but in this thesis, only the Shaw destruction method introduced in section 6.6.2 is included. There are two reasons for this. First, initial testing indicates that the Shaw destruction method is the most suitable destruction method when it comes to choosing a single destroy method. Second, in general, other fix and optimize heuristics are characterized by longer iteration times than for ALNS-methods; therefore, the statistical effect of including multiple destroy methods is expected to be small compared to including it in an ALNS, as the number of iterations for the FO is expected to be much lower. Therefore, introducing an adaptive framework seems to not make sense in this case.

7.2.3 Model Details

The pseudo-code of the FO is presented in Algorithm 7. It starts with a feasible initial solution where all the decision variables are fixed. In lines 1-5 static parameters are assigned values and dynamic parameters are initialized. Then, in lines 6-13 the heuristic initializes internal internal variables, sets and counters. The destroy and repair phases repeats at lines 14-37 in a while-loop as long as a stopping criteria is not met. First, the current solution is destroyed in line 15, before being repaired in line 16. If the new candidate solution is strictly better than the existing solution, the candidate solution is accepted (line 18-20). If the solution is not better, and criteria for tuning the parameters are met, then the parameters are changed in lines 23-28. In line 31, if criteria for guiding the model is met, then the node in the current set of routes which has been destroyed least frequent is chosen. Potential tuning and/or guiding actions are implemented when the while loop returns to line 15. When a stopping criteria is met then the heuristic finishes of and returns a solution in line 39.

For a given iteration, fixation of legs shrinks down the solution space, but it may also speed up the MIP-solving procedure significantly. Whether the the FO is allowed to move relatively freely around in the solution space while still being able to close iteration gaps fast are key to its success. As each iteration corresponds to fixing parts of the previously found solution, each iteration will not produce

a worse solution than the one already found, as long as the solver is allowed to run until optimality. The time budget allocated for each model iteration will not always suffice in order for the MIP to close the gap at each iteration; however, only strictly improving candidate solutions are accepted (for FO) throughout this thesis.

Destroy Method

Although the destroy procedure is not adaptive in the sense of learning which destroy methods to use, it is still possible to apply smart destroy procedures. In the model presented here, the FO periodically deviate from the Shaw destruction method by guiding the model to start the destroy procedure by destroying a node which is both included in the current solution and which also has a low rate of destruction. After destroying the root node, the Shaw destruction method destroys other parts of the solution space as normal, before repairing the solution again.

Parameters

The Fix and Optimize LNS has two main parameters in addition to some indirect parameters (which may be thought of as parameters guiding the two main parameters). The two parameters are the *degree of destruction* and the *max iteration time*. As the names suggest, they control the degree of destruction at each iteration and the time allocated for repairing the destroyed solutions, respectively. By varying these parameters, the intensification and the diversification of the search is controlled. For instance, by increasing the destruction rate the diversification is increased, allowing the model to access larger areas of the solution space in a given iteration. These parameters are adjusted dynamically. The parameters may or may not be monotonously increasing (e.g fluctuating up and down from iteration to iteration). In the model presented here, the degree of destruction and the max iteration time are monotonously increasing. Testing related to tuning of the FO are found in Section 10.4.2.

Algorithm 6 Initial solution for FO or ALNS

Input: Problem instance
Output: Feasible initial solution, S^0

$S^0 = None$ ▷ initial solution (vessel legs)
if startup using ADP-plan is preferred **then**
    if S^{ADP} feasible **then**
        $S^0 = S_{ADP}$
    else
        $S^0 \leftarrow Destroy\_infeasibility(S^{ADP})$
    end if
else
    $S^0 \leftarrow Construction\_heuristic()$
end if
return S^0

Algorithm 7 Fix and Optimize

Input: initial feasible solution, S^0 (vessel legs only)
Output: final solution, S^f (complete solution)

assign: p^{max_iter} ▷ max # of non-improving iterations before self-tuning
assign: $p^{max\_iter\_gap}$ ▷ max gap: tuning based on iteration gaps
assign: p^{freq} ▷ guiding frequency
initialize: p^{d_rate} ▷ current destruction rate
initialize: p^{t_max} ▷ current max time budget for each iteration

$S^d = [\,]$ ▷ destroyed solution (vessel legs only)
$S^t = None$ ▷ candidate solution
$S^c = S^0$ ▷ current solution
$n^{iter} = 0$ ▷ current iteration number
$n^{non} = 0$ ▷ current number of subsequent non-improving iterations
$d^{guide} = None$ ▷ guide / start node for destruction method
$d^{freq} = [\,]$ ▷ list of destroy frequency for each node in data set
$g^{avg} = None$ ▷ mean gap for recent iterations

while time < max running time **do**
    $S^d \leftarrow Destroy\_solution(S^c, p^{d\_rate}, d^{guide})$
    $S^t \leftarrow Solve(S^d, p^{t\_max})$ ▷ solve a restricted MIP imposing S^d
    update g^{avg}
    if $eval(S^t) > eval(S^c)$ **then** ▷ compare with prev. best solution
        $S^c = S^t$
        $n^{non} = 0$
    else
        $n^{non} = n^{non} + 1$
        if $n^{non} > p^{max\_iter}$ **then** ▷ if solution is not improving, tune model
            if $g^{avg} < p^{max\_iter\_gap}$ **then**
                increase p^{d_rate} ▷ destroy more
            else
                increase p^{t_max} ▷ solve longer
            end if
        end if
    end if
    if $mod(n^{iter}, p^{freq}) = 0$ **then** ▷ guide algorithm every d^{freq} iteration
        $d^{guide} \leftarrow Find\_node(S^c, d^{freq})$ ▷ node with low destroy frequency
    else
        $d^{guide} = None$
    end if
    $n^{iter} = n^{iter} + 1$
end while
$S^f = S^c$
return S^f

7.3 Robustness Strategies

In this section, we propose three robustness strategies to guide the proposed solution methods to find more robust and reliable solutions. The proposed strategies are chosen based on considerations of what is realistic to implement in a planning horizon of 45-90 days. The three strategies are 1) penalizing arrival after the start of time windows 2) increasing buffer quantity 3) increasing sailing time between ports.

The proposed strategies are motivated by the work of Halvorsen-Weare, Fagerholt, and Rönnqvist (2013) and Fischer et al. (2016). In the first paper, the authors propose three robustness strategies for a routing and scheduling problem belonging to a producer with one production port and a planning horizon of 3-12 months. The three proposed strategies are increasing sailing time, targeting inventory to a certain level to avoid hitting a minimum or maximum level and penalizing accumulated berth use if it deviates from a target level. Due to a shorter planning horizon, a lower degree of flexibility, and berth control exercised by the producer in the problem studies in this thesis, the last two strategies are considered unrealistic. In the second paper, the authors study disruption management in roll-on roll-off liner shipping. In addition to adding extra sailing time, the authors propose rewarding early arrivals and penalizing start times that are close to the end of time windows.

7.3.1 Penalizing Late Arrivals

To penalize late arrival after the start of time windows, the following term is added to the objective function.

$$-\alpha \sum_{v \in V} \sum_{i \in N} (t^s_{iv} - \sum_{j \in N} T_i^{MIN}) \tag{7.1}$$

where α is the penalty parameter. One drawback of this strategy is estimating/tuning the parameter α. On one hand, A small penalty will barely have an impact on an objective function consisting of, among others, revenues worth many million dollars. On the other hand, a very big α might have an opposite impact and render an optimal and reliable solution sub-optimal. For these reasons, α should be deliberately chosen.

7.3.2 Increasing Sailing Time

As proposed by Halvorsen-Weare, Fagerholt, and Rönnqvist (2013), adding extra time to the actual sailing time is easy to implement and a straight-forward way of adding slack in vessel schedules. Embedding this strategy is usually done by manipulating the input to the solution model or by multiplying the sailing time parameter T^S_{ijv} with a scaling parameter ω. Similarly to the penalty α, the amount of extra time added to sailing time has to be carefully chosen. For a vessel with a tight and unrealistic schedule, an increase of ω might prevent the vessel from sailing the same node sequence as in the original solution. However, this strategy has a drawback as pointed out by Halvorsen-Weare, Fagerholt, and Rönnqvist (ibid.). This drawback becomes visible when a fleet is fully utilized. Increasing sailing times will render some customers unvisitable with own fleet and have to be either breached or chartered which incur higher costs. For this reason, it is recommended to show caution when using this strategy. This can be done by comparing the solutions obtained by using this strategy with a solution obtained by other strategies or the original solution.

7.3.3 Increasing Buffer Quantity

Buffer quantity is often used to avoid cool-down, and it constitutes a tiny amount of the LNG on board (typically 3% of the vessel's capacity). Whenever the amount of LNG on board is below this quantity, the vessel has to be cooled down which often takes around 24 hours. This usually happens when a vessel uses BOG as fuel between a delivery and production port and gets delayed during sailing. For an LNG vessel with a tight schedule, violating the buffer quantity constraint may render the rest of the route infeasible as it has to undergo an emergency cool-down which may cause an unbearable delay in.

Similar to the strategy of increasing sailing time, this strategy is straight-forward and easy to implement. This is done by either adjusting the buffer quantity in the input or by multiplying the buffer quantity parameter with a scaling parameter, β.

8. Simulation Model

This chapter presents a stochastic dynamic discrete-event simulation framework developed to 1) evaluate the robustness of a solution in a realistic setting 2) evaluate robustness strategies that help to guide a solution method to more robust solutions. First, the aim and motivation behind implementing this framework are presented in Section 8.1. Second, a general overview is presented in Section 8.2. Third, all the components of the framework are described in detail in Section 8.3. Fourth, three robustness strategies are presented in Section 7.3.

8.1 Aim and Motivation

As described in 1, the LNG supply chain sets apart from other industries with its highly integrated supply chain both legally and economically through long-term contracts. However, its logistic part consists of an expensive and relatively small fleet compared to other industries and is subject to strict operational constraints like tight time windows, low flexibility in rerouting options and high penalties in case of violating these restrictions. All these factors place a massive demand for delivery reliability and robustness.

As the model presented in 5, it assumes deterministic input, solving it to optimality or near optimality does not stress the impact of the real-world uncertainties on the solution's delivery reliability. Hence, an optimal solution for a given input is only optimal for that input, and a small deviation from that input may render it to be infeasible in the real world. This fact is known as the knife-edge property in Stochastic Optimization describing how optimal solutions based on deterministic input often fail to generalize to small changes in the input data.

The purpose of this simulation model is to evaluate solutions in an environment that mimics a real-world setting. They may be obtained by solving a MIP to optimality or near-optimality by FO or ALNS. A solution in this context contains all the decisions made in the chosen solution method and refers to 1) which loading

and unloading nodes to be visited by each vessel 2) in which sequence and speed 3) at what time 4)how much to load/unload at each node 4) which fixed contracts to satisfy through chartering 5) how much LNG to sell as FOB at which node and period. Note that instance hereby is used to refer to all information needed to describe entities like nodes, ports, vessels and to solve the routing and scheduling problem, while a configuration is used to refer to a combination of an instance and solution which means all required input to define an instance, route, and schedule for each vessel and all other decisions related to the instance.

8.2 General Overview

Figure 8.1 shows a flow chart of the simulation framework. The arrows indicate the information flow between its components. The framework starts by reading input data about a given instance. The instance data is passed to a solution method that produces a set of solutions. For each solution, an environment where a simulation is processed and tracked is initialized.

The simulation starts at time $t = 0$ and iterates through all events that are planned during the planning horizon. Different types of disruptions are introduced and contingency strategies are performed if possible. At each event, global variables like profit, quantity in each vessel, and few others are updated. As an example, a vessel v starts sailing from the origin at $t = 0$ to the next port in the planned route. Let's assume that the next port is a loading port. Now, an event representing the arrival of the vessel at the next port is scheduled and added to a queue of events. The time of arrival at the next port is determined by the distance between ports, speed and unforeseen disruptions like bad weather conditions which might increase the sailing time. Since sailing is an event that incurs an operational cost, profit is now updated. Right before arrival, an event representing loading is scheduled. Once the vessel arrives at a port, the simulator activates the loading event and schedules a new sailing event to the next port in the route. In the same fashion, the simulator continues to track the planned route by generating and processing events. Note that vessels are not simulated independently since they are connected together by the total amount of LNG available at loading ports. When the simulator is done with iterating through the planning horizon, and all events are processed, the simulation is completed, and output and statistical measures are generated. Simulating one configuration one time represents only one possible outcome of the disruptions. Hence, it is crucial to simulate each configuration many times to be able to make reasonable conclusions when comparing different solutions.

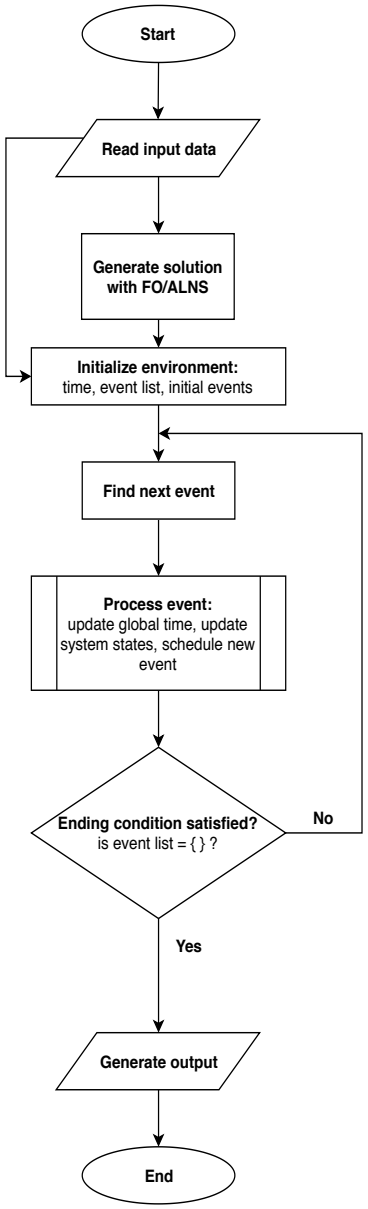


Figure 8.1: Flow chart of the simulation model

8.3 Simulation Components

In the section, the components of the simulation framework are presented.

8.3.1 System States

System states in this context refer to all variables needed to describe the system at any time. The system states used in this framework are the profit of the producer, total allocated volume at each loading port and period, and LNG quantity at each vessel.

Profit

Profit, u, is calculated sequentially and updated whenever an event has an impact on costs or revenues. In general, an increase in profit is caused by either delivering LNG to a customer or selling excessive LNG as FOB at loading port. Contrary, profit is reduced due to operational costs, cost of breaching fixed contracts, and costs related to chartering vessels.

Allocated Volume

Allocated volume, q_p, represents the available LNG at port p. q_p is updated when LNG is loaded into a vessel or used to cool down a vessel at a port which happens whenever the quantity level on board is less than a required buffer to avoid cool-down.

Vessel Quantities

Vessel quantity, q_v describes the amount of LNG on board vessel v. It is updated when LNG is loaded or unloaded into the vessel. Furthermore, a small amount of the LNG quantity on board fades out during sailing between ports due to boil-off. For this reason, q_v is also updated right before arriving a port where an amount equivalent to the vessel's BOG is subtracted from q_v.

8.3.2 Entities

Entities are all the objects that are explicitly represented in the simulation. Each entity usually has attributes that make it different from its counterparts. Vessel, node, and port are the different entities used in this model.

Vessels

The entity vessel is described by the attributes capacity, boil-off rate, the required amount of LNG used in cooling down the vessel, speed modes, a buffer quantity required to avoid cool-down at the next loading port.

Nodes

A node is described by which port it belongs to, its time window and (un)loading rate. If a node is a delivery node, it has an attribute to describe its quantity window and potential revenue per unit LNG.

Ports

The port entity is an umbrella for all the nodes that belong to it. It is essential to track the change in the system state allocated volume q_p.

8.3.3 Environment

Environment refers to the world where events are created, processed and stored. It also keeps track of time, current event, and events to be processed.

Clock

The environment contains the simulator's clock is represented by a global and dynamic variable in T^{Global} which keeps track of time during the simulation. T^{Global} is event-based and is updated whenever a new event is processed. The time variable is initially set to 0.

Tracking Events

The environment keeps also track of the state of each event. An event has three possible states in the simulation, planned, triggered and processed. Apparently, an event is planned when it is decided on by the solution method, yet, it is not known for the event queue, which we denote by Events List. An event is triggered when it is scheduled and is added to the events list. Finally, an event gets the state processed when it has happened and all the relevant system states are updated.

Scheduling and Processing an Event

When an event is triggered, the event, its ID and occurrence time, (time, ID, event), are added to the events list. When processing an event, the event with the

earliest occurrence time is removed from the events list and processed further in the simulation. If two events have equal occurrence time, the simulator chooses the event which was added first to the events list.

In order to schedule and process events efficiently, the events list is stored as a heap or a binary tree where each parent node has a value less or equal than its child nodes. This data structure allows using the heap queue algorithm with two operators INSERT and EXTRACT-MIN. The operator EXTRACT-MIN is called to find the next event to simulate, while the operator INSERT is called when a new event is scheduled. The running time of both operators is $O(\lg n)$.

8.4 Process-Related Events

In this section, the process events are described in detail. The flow chart in figure 8.2 shows which the sequence of events a vessel undergoes during one route. The main events are sailing, arrival, loading, unloading, cool-down. Also, an instance may include FOB-sale and chartering. The latter one is considered as a vessel with a route consisting of two nodes; origin and destination.

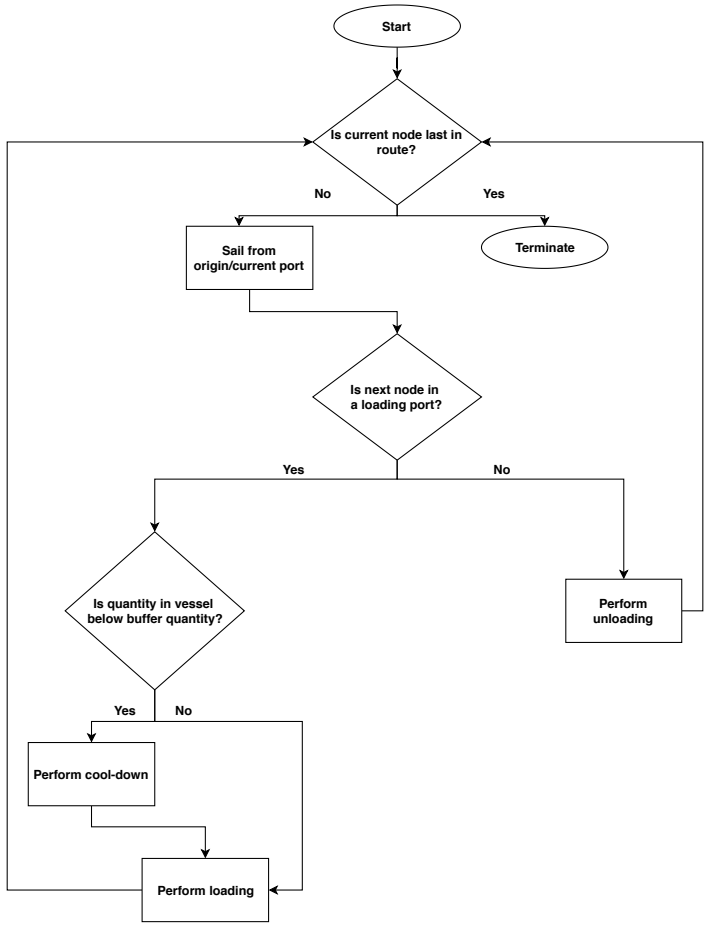


Figure 8.2: Flow chart of a vessel's journey in the simulator

8.4.1 Sailing

Sailing is an activity that constitutes a large part of a vessel's life and is the part where disruptions may cause significant delays and accelerate costs. Rough weather conditions, mechanical failure during sailing in open water or congestion in passageways are not unusual disruptions that a vessel's operating crew has to handle very often. As an illustration, figure 8.3 shows a realization of a probability density function fitted to sailing times between Rome (Italy) and Bergen (Norway) by Halvorsen-Weare, Fagerholt, and Rönnqvist (2013). Even though the voyage Rome - Bergen is considered relatively short with an average sailing time of 148 hours, the figure shows how sailing time could be spread out. Assuming the same probability density function as the authors, the theoretical probability of using more than 148 hours is calculated to be around 32%. With this in mind, a tight schedule of an LNG vessel is likely vulnerable to the uncertainty in sailing time.

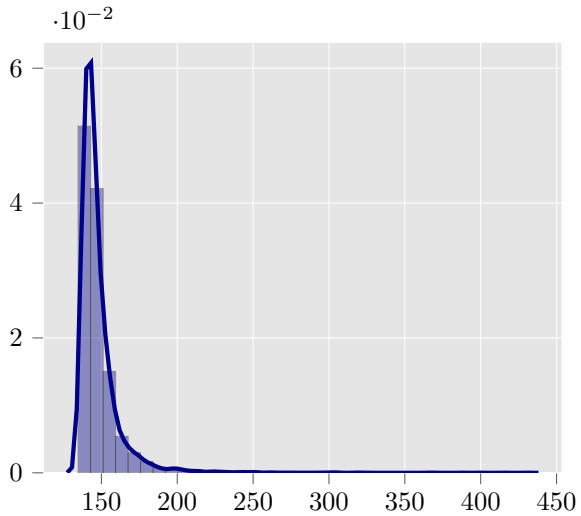


Figure 8.3: A realization of fitted probability density function to sailing time data between Rome (Italy) and Bergen (Norway). Source: Halvorsen-Weare, Fagerholt, and Rönnqvist (2013).

Modeling Weather States and Their Impact on Vessel Speed

To mimic the real world, four different weather states are used. Each state causes a certain speed reduction proportional to the prevailing wave height in the state.

Even though this assumption is a simplification of the real world, it resembles the relationship between wave height, the efficiency of a propeller and a vessel's forward thrust. This relationship can be shown in table 8.1 with four states representing different weather conditions, different wave height intervals and possible speed reduction for supply vessel operating in the North Sea as presented in Halvorsen-Weare and Fagerholt (2011).

Table 8.1: Weather states with the associated wave height interval and speed reduction. Source: Halvorsen-Weare and Fagerholt (2011)

Weather state	Wave heigh [m]	Sailing Reduction
1	<2.5	0 kn
2	<2.5,3.5]	0 kn
3	<3.5, 4.5]	-2 kn
4	≥4.5	-3 kn

In order to simulate the four states, the Markov chain is used with starting state probabilities for each state and a transition matrix that describes the transition probability from one state to another. Weather conditions in a period of time are simulated by first dividing the period into small sub-periods. In the first sub-period, an initial state is drawn based on the start probabilities. The weather state in the next sub-period is then dependent only on the weather state in the current sub-period and can be drawn based on the probabilities of transition from the current state to the others. Table 8.4 and 8.3 show starting state probabilities and transition matrix for the four states in table 8.1.

Table 8.2: Starting state probabilities. Source: Halvorsen-Weare and Fagerholt (2011)

State	1	2	3	4
Probability	22.7%	27.1%	28.2%	22.0%

Table 8.3: Transition probability matrix. Source: Halvorsen-Weare and Fagerholt (2011)

State	1	2	3	4
1	82.5%	16.9%	0.6%	0.0%
2	14.0%	60.6%	20.7%	4.7%
3	0.5%	23.9%	57.7%	17.9%
4	0.0%	0.6%	27.9%	71.5%

Implementation

In the simulation framework, the event sailing is generated whenever a vessel is done with a loading or unloading operation. Sailing distance is estimated based on the location of the current port and the location of the next port in the route. Once the sailing distance is found, the sailing time is estimated based on planned speed and distance, sailing disruption and contingency measures which are triggered whenever the vessel risks violating time window constraints in the remaining part of the route. The planned speed, h, is determined by a linear combination of speed modes and the binary variables w_{ijvh} as described in Chapter 5.

Algorithm 0 describes how sailing is simulated between a pair of nodes (i, j). The algorithm is inspired by the paper of Halvorsen-Weare, Fagerholt, and Rönnqvist (2013) which focuses on routing and scheduling of supply vessels in the North Sea. The states, start probabilities and transition matrix are also reused from the same paper due to lack of weather and vessel data. Note that the vessels studied in the thesis operate in a higher speed range than supply vessels and in different areas in the world, yet the weather states and the impact of wave heights on speed are assumed equally applicable on LNG vessels.

The input of the algorithm of simulating a voyage is the voyage origin i, voyage destination j, distance d_{ij}, the transition matrix P and the probabilities of starting states P_0, the duration of period with fixed weather condition Δ, maximum speed h_v^{MAX} and speed h. The output is a simulated sailing time T_{ijv}^S.

The algorithm starts by calculating planned sailing time T_{ijv}, the number of periods with fixed weather state in the planned sailing time N, the distance that could be sailed in one period $d_{ij\Delta}$ and initializing a starting state ξ_0 by Generate-Start-State. Sailing is then simulated for that period by Calculate-Simulated-Period. This is done by sailing the distance $d_{ij\Delta}$ with a speed reduction equivalent to the

current speed. If the cumulative sailing time so far deviates from the planned sailing time, the contingency measure Speed-Up is carried out, the distance $d_{ij\Delta}$ is then sailed with the speed $h_v^{MAX} -$ speed_reduction(xi). The cumulative sailing time so far is then updated. A new weather state is generated, and the same procedure is carried out. The contingency measure Speed-Up($d_{ij\Delta}$, h_v^{MAX}) is implemented to reflect the fact that the vessel's manager/crew is likely to speed up in case of delay.

Algorithm 8 Simulating sailing between node i and j by vessel v

Input: v, i, j, h_v, h_v^{MAX}, d_{ij}, P, P_0, Δ
Output: T_{ijv}^S

$T_{ijv} = \frac{d_{ij}}{h_v}$ ▷ Calculate planned sailing time
$N = \frac{T_{ijv}}{\Delta}$ ▷ nT is number of periods in T_{ijv}
$d_{ij\Delta} = h_v\Delta$ ▷ Distance sailed in period Δ with speed h_v
$T_{ijv}^S \leftarrow 0$
$\xi_0 \leftarrow$ Generate-Start-State(P_0)
$n \leftarrow 1$
for each t in 0 **to** N **do**
    $T_t =$ Calculate-Simulated-Period(ξ_t, T)
    if $T_{ijv}^S + T_t < n\Delta$ **then** ▷ If T_{ijv}^S so far deviate from plan
        $T_t =$ Speed-Up($d_{ij\Delta}$, h_v^{MAX})
        $T_{ijv}^S = T_{ijv}^S + T_t$
    else
        $T_{ijv}^S = T_{ijv}^S + T_t$
    end if
    $\xi_{t+1} \leftarrow$ Generate-Next-State(ξ_t, P)
    $n = n + 1$
end for
return T_{ijv}^S

8.4.2 Arrival

The event arrival is scheduled whenever a sailing event is triggered. The arrival time is determined based on T^{Global} and the time of simulated sailing before arrival. The event is responsible for updating global variables affected during sailing. Additionally, it generates the next event occurring in port, which typically is loading, unloading, or cool-down.

8.4.3 Cool-Down

The event cool-down is triggered whenever a vessel arrives at a pickup node j and the variable y_{ijv} is fixed to 1 by the solution method. The event has an impact on global time since a cool-down operation requires a period of time equal to T_{jv}^{CD}. Additionally, it incurs a cost of C_{jv}^{CD}.

8.4.4 Loading

The event Loading is generated by either the event arrival or cool-down. Since a vessel may speed up during a leg if it risks delays in the future, it may use less sailing time than planned. Hence it may waste less BOG and contain more LNG than what is planned. For this reason, one has to check if the sum of the planned quantity to load at port, q_{iv}, and the total quantity at the vessel prior to arrival, γ_{iv}^{S}, does not exceed the vessel's capacity, V_v^{CAP}. If so, a contingency measure is carried out as seen in Algorithm 9. The new quantity to be loaded is then reduced to (l_{iv} - γ_{iv}^{S}).

Algorithm 9 Simulating the Event Loading

Input: Planned quantity to be loaded at node i q_{iv}, leaving quantity l_{iv}, simulated incoming quantity γ_{iv}^{S}, vessel capacity V_v^{CAP}
Output: Simulated quantity to load q_{iv}^{S}

```
if γ^S_iv + q_iv > V^CAP_v then ▷ If loading quantity violated capacity constraint
    q^S_iv ← (l_iv - γ^S_iv)
else
    q^S_iv ← q_iv
end if
return q^S_iv
```

8.4.5 Unloading

Similarly to the event loading, contingency measures have to be carried out in case of deviance from the plan obtained from the used solution method. This concerns the case where the quantity at a vessel prior to arrival is less than what is planned. There are three scenarios:

- Scenario 1: The vessel contains less LNG than an adequate quantity to cover the planned quantity to unload and heel to avoid cool-down at next port (if heel-out is not planned)

- Scenario 2: The vessel contains less than what is planned to unload, but it is possible to unload more than the minimum quantity required by the customer.
- Scenario 3: The vessel contains less than the minimum quantity required by the customer.

The inputs of the algorithm are Planned quantity to unload at node i, q_iv, minimum and maximum quantity window of node i, (Q_i^{MIN}, Q_i^{MAX}), planned and simulated quantity at the vessel before arrival, γ_{iv} and γ_{iv}^S, estimated time of service start at next node t_{jv}^{SS}, estimated time of service end at current node t_{iv}^{ES}, heel-out variable y_ijv, and finally the vessel's boil-off rate B_v. The output is the simulated quantity to unload after performed contingency measures, q_{iv}^S

Algorithm 10 Simulating the event unloading

Input: q_iv, Q_i^{MIN}, Q_i^{MAX}, γ_{iv}, γ_{iv}^S, t_{jv}^{SS}, t_{iv}^{ES}, y_ijv, B_v
Output: q_{iv}^S
  if $\gamma_{iv}^S < \gamma_{iv}$ **then** ▷ Quantity on board less than planned
    if $\gamma_{iv}^S > q_iv + y_ijv\ B_v(t_{jv}^{SS}$ - $t_{iv}^{ES})$ **then**
      $q_{iv}^S \leftarrow q_iv$
    else
      if $\gamma_{iv}^S > q_iv$ **then** ▷ Scenario 1
        $q_{iv}^S \leftarrow \min(\gamma_{iv}^S, Q_i^{MAX}$
      end if
      if $\gamma_{iv}^S < q_iv \wedge \gamma_{iv}^S > Q_i^{MIN}$ **then** ▷ Scenario 2
        $q_{iv}^S \leftarrow \gamma_{iv}^S$
      end if
      if $\gamma_{iv}^S < Q_i^{MIN}$ **then** ▷ Scenario 3
        $q_{iv}^S \leftarrow \gamma_{iv}^S$
      end if
    end if
  end if
  return q_{iv}^S

8.4.6 FOB Sale

FOB sale is an event that is not connected to any of the producer's vessels. Hence it is scheduled during the initialization stage of the simulation. Note that the quantity to be sold is initially determined by the solution method. In many cases, the planned quantity to sell is more than what is available of inventory at the

event time. This happens as a result of longer voyages between ports which cause vessels to load more than what's planned and in many cases perform emergency cool-downs. For this reason, the quantity sold as FOB is limited to available inventory.

8.4.7 Disruption-Related events

In addition to longer sail times due to rough weather conditions, loading/unloading less than what is planned, the following events are included in the framework.

Emergency Cool-Down

This event is similar to the event cool-down, however, it is considered disruptive as it incurs unintended delay and costs. The event is triggered whenever a vessel arrives at a loading port and the quantity on board is less than a buffer quantity Q_v^S. It incurs a delay equivalent to $T_{iv}^C D$, an additional cost of C_{iv}^{CD} and reduces the LNG inventory at port with an amount of Q_v^{CD} which is used to cool down the vessel.

Disruptive Events at Ports

Port congestion was the main contributor to schedule unreliability in line shipping, according to Notteboom (2006). In another paper by Berle et al. (2013), the authors identify the most critical sources of disruption in the LNG transportation in collaboration with practitioners from the LNG industry. Seven of the nine identified sources are related to port disruption. Four of the seven events are found relevant for our problem and are included in the simulation framework. The events and their probabilities, as identified by the authors are shown below.

8.4.8 Output

When a simulation is done, the output is generated and statistical measures are collected. This includes an estimate of the objective function and the total time violation in all nodes visited by the fleet. Additionally, all events are printed out for a quality check.

Table 8.4: Disruptive events related to ports. Source: Berle et al., 2013

Events	Description	Probability per day
Unavailable loading port	Loading port is unavailable for 48 hours	0.002
Unavailable unloading port	Unloading port is unavailable for 96 hours	0.002
Loading rate down	Loading rate is reduced by 50% for seven days	0.001
Unloading rate down	Unloading rate is reduced by 50% for 14 days	0.001

9. Data Research & Generation

This Chapter serves to give a walk-through on how data instances are created. In order to test and to keep developing the model, it is vital to have input data sets which represent the problem environment well. In the absence of real data from the industry, various techniques have been used in order to mimic a real data set. The data sets are used in subsequent chapters to evaluate and to test the different versions (e.g. fixed vs. variable speed) of the model described in previous chapters. In an attempt to create realistic input instances, input data, although non-verified, is based on written research and expert interviews (more specifically, university professors within operations research at NTNU and industry experts from Tieto Oil & Gas).

9.1 Input Data

In this Section, we present the information used as input to the instance generator.

9.1.1 Case Description

The case studied in this thesis is a set of fictitious agreements of LNG transportation between a producer and several customers located in Australia and East Asia, based on long-term contracts. The producer operates three liquefaction ports in Australia and one in Indonesia. The producer is responsible for delivering LNG to customers at five different ports located in Japan, China and South Korea, connected to regasification plants operated by the respective customers. The ports used in the case are inspired by real LNG projects found in GIIGNL (2018). The contract/project specifications for the long-term contract between the producer and each customer will vary for each test instance since the size of the instances changes number of deliveries, as described in Section 9.5.

The distances between each port is shown in Table 9.1 and 9.2. *Sea routes and*

distances (n.d.) was used to find the distances. Compared to other free online tools for finding sailing distances between ports, it includes a large number of ports, however, the accuracy is lower than for other tools studied. Still, the accuracy was assumed sufficient for the test case.

Table 9.1: Distances in nautical miles between liquefaction and regasification ports

	Sodegura	**Hibiki**	**Senboku II**	**Hainan**	**Pyeong-Taek**
Withnell Bay	3682	3905	3554	3301	4035
Barrow Island	3730	4027	3602	3271	4159
Curtis Island	4248	4614	4844	4878	4974
Bontang	3141	2695	3393	1897	2790

Table 9.2: Distances in nautical miles between regasification ports

	Sodegura	**Hibiki**	**Senboku II**	**Hainan**	**Pyeong-Taek**
Sodegura	-	664	777	2475	1149
Hibiki	664	-	715	1947	497
Senboku II	777	715	-	2646	1195
Hainan	2475	1947	2646	-	1939
Pyeong-Taek	1149	497	1195	1939	-

9.1.2 Vessel Characteristics

The producer operates a fleet of heterogeneous vessels, however, for simplicity the vessels are assumed to be of one of three vessel types. Characteristics for each vessel type is summarized in Table 9.3.

Table 9.3: Vessel type characteristics

Vessel Type	Capacity [m^3]	Loading Rate [m^3/h]	Design Speed [kn]	Engine Type
A	120.000	11.000	18	Dual-Fuel
B	150.000	13.000	18	Dual-Fuel
C	260.000	11.000	18	Slow-Speed Diesel

Vessel type A and B have sizes within the standard size of vessel before the introduction of the Q-class vessels. Both of these vessel types have dual-fuel engines that

can run on both BOG and fuel oil. The boil-off rate is assumed to be 0.15% of the total vessel capacity each day (see 2.1.1). Vessel C resembles the Q-max vessel type, and although none of these vessels currently operates between Australia/Indonesia and East Asia, this vessel type is included to open up the possibility of a vessel doing two consecutive deliveries that are not planned in the ADP. This vessel type is assumed to be equipped with a reliquefaction system, so the boil-off rate is set to 0. The loading rates for all the vessel type are selected within the normal range (for more details about loading rates, see Section 2.1.1), and the unloading rate is assumed to be equal to the loading rate. In reality, the loading and unloading rates may vary from port to port, depending on the facilities and equipment in the ports, but they are here assumed to be independent of the port visited. Each vessel is only loaded to 95% of its capacity to prevent liquid from entering into the ventilation pipeline and spilling into the hull structure of the vessels. If one of the vessels has to undergo a cool down process, the required gas volume to cool down the tanks are assumed to be 3% of the the vessel's tank capacity, based on the studies in Moon et al. (2005) and Iversen and Sørensen (2005). The cool down time is 24 hours, as described in Section 2.1.1.

Speed modes and fuel consumption

LNG vessels usually operate between 15 and 22 knots (GIIGNL, 2018). Vessel type A-C are all assumed to have similar speed characteristics, with 18 kn as the design speed (used in fixed speed model) and five speed modes corresponding to 15 kn, 18 kn, 18 kn, 20 kn and 22 kn respectively (see Section A.2.1 for the reasoning behind duplicate speed mode at design speed).

Vessel type A and B are assumed to be able to run completely on BOG at vessel speeds up to the design speed. At higher speeds, a switch to fuel oil is required. The consumption of fuel oil at the design speed for each vessel type is estimated based on Toth and Vigo (2014) and IGU (2018) as 95.3 ton/day, 104.8 ton/day and 115.3 ton/day for vessel type A, B and C respectively. The fuel consumption at the other speed modes are calculated using Equation 2.1. We neglect the difference in fuel consumption on laden and ballast sailing legs.

The fuel oil used in the transportation case is IFO 380 for all vessel types. The fuel price was set equal to the average price of IFO 380 in Singapore in 2017 - 329 USD/ton (Ship & Bunker, 2018).

9.1.3 Revenue and Costs

The LNG contracts that we have based our case on were negotiated in 2013-2014. The average long-term contract price of LNG in those years were around 15 USD/MMBtu in Asia (*Drawn-out ball game: Asian spot LNG prices to stay below long-term* n.d.). The LNG price has dropped significantly since then; the Ministry of Economy, Trade and Industry in Japan report a spot price (pure spot contracts) in Japan in November 2018 of 11.5 USD/MMBtu (*Spot LNG Price Statistics* 2018). As indicated by *Drawn-out ball game: Asian spot LNG prices to stay below long-term* (n.d.), it is likely that the contracts initiated around 2014 have been renegotiatied to reflect the current price trends in the Asian LNG market. We therefore assume that the price of LNG in the long-term contracts are 10 UDS/MMBtu, equaling 240 USD/m^3 when applying a conversion factor of 24 MMBtu/m^3 LNG (IGU, 2012). The spot price is set to 11.5 UDS/MMBtu. Due to lack of available data on LNG FOB prices in Asia, the FOB price is assumed to be 5.7 UDS/MMBtu, based on a 15% increase of the Henry Hub spot price and a liquefaction charge of USD 2.25 (Pedersen, 2017).

Operating costs are omitted in the model as it is assumed fixed for the planning horizon. For simplicity we have also neglected port costs, as these is likely to not vary significantly for port to port and therefore not have a large effect on the solution. The only component included in the cost parameter is therefore the fuel cost, discussed above. The cost of undergoing a cool down process is estimated to be USD 55,000 from DESFA price indications (DESFA, n.d.). The cost of cool down might depend on port and vessel, but is here assumed to be the same for all.

9.2 ADP Generation

The model requires input on reserved slot times in delivery and pickup ports, in the form of time windows, as well as contract specification on quantities and cargo flexibility. This would ideally come from an ADP or another planning problem. Since this data was not available, time and quantity windows were generated by giving each vessel a predefined path. Moreover, information on port limitations and contract requirement of LNG origin is needed. We assume that all vessels can visit all ports, and that each regasification terminal is indifferent to the origin of the LNG delivered.

For simplicity the origin of each vessel is set to a random port. Each vessel can either be empty or (almost) full, which decides if the first port in the path of a vessel is a liquefaction or regasification port. A disruption is added to the origin of the vessel to reflect delays and updated information, as discussed in Section 9.4.3.

For each node in the path of a vessel, the sailing time at design speed is used to calculate the expected arrival time of the vessel. A time window of 48 hours is added to the expected arrival time. For delivery ports, the contract volume is calculated from the maximum volume the assigned vessel can carry, subtracted the expected boil-off from the previous node in the path. A quantity window of ±15% of this value is then allowed. If the number of pickup nodes is greater than the number of delivery nodes, the time windows for the excess pickup nodes are evenly distributed over the planning horizon, again with a length of 48 hours.

9.2.1 Allocated Volume

The planning period is the divided into four periods (where the last period represents the first period of the next planning period), and the three first periods are given a "planned" allocated volume based on the volume necessary to serve all planned pickups with the beginning of its time window in that time period. This allocated volume is then disrupted as explained in Section 9.4.2 to reflect production changes and updated production data from the ADP.

All LNG not committed to long-term or spot contracts are either transferred to the next time period or sold as FOB. The available vessel capacities for FOB pickup are assumed to be between 100,000m^3 and 150,000m^3, giving the boundaries for the FOB sale quantity at a specific node. The volume transferred from last month in previous planning period is set to 10,000m^3 for all liquefaction ports.

9.3 Chartering

The vessels that can be chartered are not specified directly, but there is a cost related to servicing a given long-term contract by a chartered vessel, picking up LNG at a specific pickup node. The volume delivered by the chartered vessel is assumed to be the middle of the quantity window for a contract, and the vessel capacity for the chartered ship is assumed to be 5% larger than this volume. The boil-off volume and volume required for cool down are then calculated as in Section 9.1.2. Charter rates are for simplicity assumed to be equal for all vessel capacities, using the average charter rate in 2017 for a 160,000m^3 LNG carrier according to GIIGNL (2018) of 46,058 USD/day. Even though the chartering can be both time charter or voyage charter, we assume that the chartering cost is the product of the charter rate and the charter time. The charter time is an estimation of the time it takes to load and unload the chartered volume (added expected boil-off to the loading volume), sail from pickup to delivery port at an assumed design speed of 18 knots, and potentially wait outside a regasification terminal due to

time windows. The time for pickup and delivery for a chartered vessel is so that the charter time is as low as possible.

9.4 Disruption and Randomization

9.4.1 Spot

Spot contracts that were not planed in the ADP are added to the problem. These are assumed to be ”pure spot”-contracts, as defined in Section 2.1.2. We set the number of spot contracts to be 20% of the number of fixed contracts, as this somewhat reflects that approximately 20% of the total LNG volumes delivered in 2017 was on pure spot contracts (GIIGNL, 2018). The time windows for the spot contracts is assumed to be 48 hours, randomly distributed over the planning horizon.

9.4.2 Randomization of Allocated Volumes

The ”planned” allocated volume described in Section 9.2.1 is disrupted to take production changes and updated production data from the ADP into account. We have assumed that the allocated volume is normally distributed around its ”planned volume”, with a standard deviation of σ of this volume. Values tested for σ are specified in Chapter 10.

9.4.3 Uncertainty in Origin

To account for delays and changes in the previous planning period, the sailing time from origin to its planned destination according to the ADP is disrupted. This is here done by assuming that the delay from origin to all nodes is exponentially distributed with an expected value, μ hours. Values tested for μ are specified in Chapter 10. However, in reality this would be done more accurately by assuming a new position for the origin and recalculating all sailing times to the different nodes.

9.5 Instance Sizes and Combination of Vessels and Nodes

When making several input data sets it is either for the purpose of increasing the statistical strength of the test results (sampling) or for the purpose of testing different sizes of problem instances. When the term “problem size” is used in this

thesis, it refers to the combination of vessels and pickup and delivery nodes. We mainly talk about fixed contract delivery nodes, even though the number of spot contracts also increase with problem size. For the test instances the ratio of pickup and delivery nodes is not 1:1. This is due to the assumption of a greater number of pickup nodes than delivery nodes, and also for making it possible for the model to choose FOB solutions. Table 9.4 shows the different problem instances. It is based on the assumption of each vessel completing on average 8 trips during a 90-day period (including the artificial origin but excluding the artificial destination). For each number of vessels, there are 4 different problem sizes (e.g. for one vessel there are problem instances with 6, 8, 10 and 12 nodes); however, it is also possible to make as many problem instances for a given problem size as needed because of the randomization when the input file is created. These vessel-node combinations may need to be updated when dealing with a specific case where for instance the sailing times are different from the ones chosen in the examples used in this thesis.

Table 9.4: Problem sizes

Instance name	# of vessels	# of pickup nodes	# of spot contracts	# of fixed contracts	Total # of delivery nodes	Total # of nodes
1_V1P5F3	1	5	1	3	4	9
2_V1P6F4	1	6	1	4	5	11
3_V1P8F5	1	8	1	5	6	14
4_V1P9F6	1	9	2	6	8	17
5_V2P10F7	2	10	2	7	9	19
6_V2P11F8	2	11	2	8	10	21
7_V2P12F9	2	12	2	9	11	23
8_V2P14F10	2	14	2	10	12	26
9_V3P15F11	3	15	3	11	14	29
10_V3P16F12	3	16	3	12	15	31
11_V3P17F13	3	17	3	13	16	33
12_V3P18F14	3	18	3	14	17	35
13_V4P20F15	4	20	3	15	18	38
14_V4P21F16	4	21	4	16	20	41
15_V4P22F17	4	22	4	17	21	43
16_V4P23F18	4	23	4	18	22	45
17_V5P24F19	5	24	4	19	23	47
18_V5P25F20	5	25	4	20	24	49
19_V5P27F22	5	27	5	22	27	52
20_V6P28F23	6	28	5	23	28	56
21_V6P29F24	6	29	5	24	29	58
22_V6P30F25	6	30	5	25	30	60
23_V6P31F26	6	31	6	26	32	63
24_V7P32F27	7	32	6	27	33	65

10. Computational Study

In this Chapter a computational study for the proposed solutions methods in 6 andn 7 is presented. Addtionally, we present a computational study of the simulation framework and the robustness strategies proposed in Section 7.3.

The models are written in Python[1] programming language on a standard computer with Intel Core i7-7700 3.6 GHz processor and RAM of 32 GB.

In Section 10.1 tests regarding opportunities of postponing certain decisions are presented. Section 10.2 introduces a set of selected problem instances of different sizes used for further testing. Here, problem instances are divided into blocks of which different realistic disruptions are applied. In Sections 10.3 and 10.4 a computational study of the two solution methods introduced in Chapter 6 and in Chapter 7 are presented. Finally, we present a computational study of the robustness strategies proposed in Section 7.3.

10.1 Sequential Decision Making

In this section, different configurations of the MIP program introduced in Chapter 5 are presented and tested. First, the concept of and motivation for postponing decisions is discussed briefly. Second, different options for excluding decision variables and potential values they can take on are discussed and tested. Third, selected configurations are presented (a configuration corresponds to a set of decisions to postpone). Lastly, the different configurations are tested using a MIP.

The tests performed in this section consists of two steps: For the first step, the MIP is simplified by fixing a subset of the variables, and the goal is to find vessel

[1]The methods presented in this Chapter are written in Python and the MIP program is written using the Pyomo package. Various MIP-solvers are supported, and Gurobi was chosen since it supports a so called "persistent solver". A persistent solver serves the purpose of efficiently informing the solver of incremental changes to a MIP model.

routes. The second step type corresponds to solving the complete problem while imposing the routes already found. The goal is to analyze whether it is possible to decrease the computational time while still being able to find good solutions (vessel routes). Initial testing has shown that after fixing vessel legs, the rest of the problem is easy to solve, and the computational time for this step is thus not included. The test procedure is fairly straightforward and details will not be discussed in further detail.

The problem considered in this thesis is a variant of the pickup and delivery problem; however, the problem considered here contains a number of complicating side-constraints (e.g. boil-off, heel-out, quantity dependent loading time). The rationale for postponing decisions is to discover side-constraints which show little importance in simultaneous decision making. These decision components might at the same time add significant complexity in the model. If this is the case, then it may be a good idea to make these decisions after other decisions have already been made.

10.1.1 Variables Considered for Sequential Decision Making

In the following, various simplifications, as well as potential benefits and limitations, are described.

Vessel Speed

To the best of our knowledge, simplifying the problem by setting vessel speed to maximum does not affect the solution quality. Initial testing with generated data sets show that the optimal solution are found in every case when using maximum speed for vessels when searching for the vessel legs, and then solving again later on with variable speed and while imposing the path already found. It is believed that this is a safe assumption for this specific problem type, as sailing costs are very small compared to revenue for this problem type, as opposed to many other maritime routing and scheduling problems.

Delivery quantity

Next, it is possible to require that vessels deliver only the minimum quantity required by customers. As discussed earlier in section 6.1 this is an important assumption for the ALNS to work efficiently. By choosing the minimum quantity, the feasible region should allow finding paths corresponding to any feasible solution in step 2.
Furthermore, imposing anything more than the lowest possible quantity is prob-

lematic, as it may exclude certain solutions completely. Enforcing delivery of max quantity resulted in a large increase in chartered cargoes and breach quantities for most of the test instances, which yielded very poor objective values. Similar although not as severe results were obtained when forcing a delivery in the middle of the quantity window. This highlights the importance of quantity flexibility when modelling allocated volume, boil-off, cool-down, FOB and other quantity and time related decisions with the detail used in this thesis. This also argues that fixing the quantity to any other value than the minimum of the quantity window in the first part of the ALNS is unfavorable.
However, a vessel route found using a method assuming minimum delivery quantity may end up finding a route which is heavily constrained with respect to time. While the route may be feasible with respect to arrival times using the lowest possible delivery quantities, this does not need to hold for other delivery quantities, as larger delivery quantities lead to longer loading/unloading duration at ports. This scenario is depicted in figure 10.1. Here one can observe that it is not possible to increase the quantity delivered to D1 because this will lead to longer unloading/loading times at D1 and at P1, which again leads to the vessel arriving after the time window at P2. As a side comment, it should be noted that such solutions with tight constraints with respect to arrival times are often not robust. In Chapter 8 different robustness strategies aiming to avoid such routes av those described above are discussed.
Another less frequent observed shortcoming of the minimum delivery quantity requirement is that it might lead to stopover decisions being made in step one which is far away from being optimal in step 2. In step 2 the minimum quantity requirement is relaxed again, but a potential issue is that stopover decisions automatically puts strict restrictions on the range of delivery quantities it is possible to achieve. For these reasons, the only quantity-related configurations included in the tests presented here are for minimum delivery quantity. Numerical results for other quantity configurations are not included.

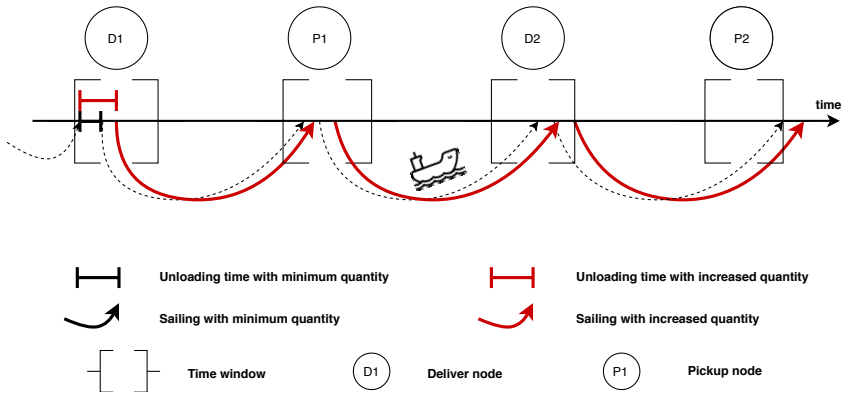


Figure 10.1: Illustratation of how tight time-windows can constrain possible delivery quantities

FOB Sale

Another possible simplification is to require that no FOB sale is allowed during step one in order to decrease computational times. This may be problematic as FOB opportunities may be limited in a real life setting and revenue per unit may also vary. Therefore, ignoring FOB early on and then later impose strict requirement on vessel legs may make it impossible to reach the optimal solution; however, it might still serve as a method to find decent paths for the vessels which can then later be improved by local search or destroy + MIP repair methods. As for all simplifications considered, there is a speed/quality trade-off.

Heel-out

Finally, it is possible to postpone the decisions concerning vessel heel-out. This may lead to sub-optimal paths in some cases (e.g. a vessel with fill capacity just large enough to serve a customer, but only if the tank is emptied completely).

To sum up, several simplifications have been considered: constraining the vessel speed, FOB sale, delivery quantity and heel-out decisions. Testing of these simplifications are presented in section 10.1.3. These tests are the basis for some of the modelling decisions made for the ALNS and for the FO.

Table 10.1: Sequential decision making - Assessing different options for simplifying decisions in step one

Configurations	Maximum speed	Minimum quantity	No heel-out	No FOB sale
Configuration 0	-	-	-	-
Configuration 1	Yes	-	-	-
Configuration 2	Yes	Yes	-	-
Configuration 3	Yes	-	Yes	-
Configuration 4	Yes	-	Yes	Yes
Configuration 5	Yes	Yes	Yes	Yes

10.1.2 Configurations of Decisions

The different configurations are listed in Table 10.1. Configuration 1 corresponds to the assumptions made for the FO and Configuration 5 corresponds to the assumptions made for the ALNS.

In Table 10.1 the different configurations of the aforementioned alternatives are presented. As already have been discussed, only the options of "maximum speed" and "minimum delivery quantity" are included for vessel speed and cargo quantity variables. Testing shows that using a lower value for vessel speed gives poor solutions, while when using maximum speed the MIP is able to find the optimal solution for all problem instances tested.

10.1.3 Results and Conclusion

In Table 10.2 computational tests with respect to running time performance and solution quality for different configurations of the MIP is presented. In essence, the purpose of this section is to analyze trade-off between objective value and computational time for the different configurations.

Making decisions sequentially could potentially be a stand-alone model, but, as can be observed from Table 10.2, using a only MIP to solve decisions sequentially is not fast or accurate enough. It still serves as an important study of which decisions which may be postponed, and which decisions may not be postponed. For instance, we observe that by finding vessel routes while assuming maximum vessel speed (Configuration 1) makes it possible to solve the problem a little bit faster while not sacrificing solution quality at all. Therefore, in the rest of this thesis, speed decisions have been postponed until after finding vessel routes. From the Table 10.2 one can observe that the minimum quantity requirement is somewhat problematic, and even more so when combined with the no heel-out and no FOB sale restrictions

Table 10.2: Sequential decision making - effect of postponing decisions

Problem instance	Configurations											
	C0		C1		C2		C3		C4		C5	
	Gap [%]	Comp. time [s]	Gap [%]	Comp. time [s]	Gap [%]	Comp. time [s]	Gap [%]	Comp. time [s]	Gap [%]	Comp. time [s]	Gap [%]	Comp. time [s]
N19V2	0.0	0.5	0.0	0.3	6.0	0.2	0.0	0.3	0.0	0.3	6.0	0.2
N21V2	0.0	0.6	0.0	0.3	0.0	0.3	0.0	0.3	0.0	0.3	0.0	0.2
N24V2	0.0	0.7	0.0	0.4	0.5	0.4	0.5	0.4	0.5	0.4	0.5	0.3
N26V3	0.0	2.1	0.0	1.2	8.5	0.8	0.0	1.0	1.2	1.1	8.5	0.7
N28V3	0.0	1.9	0.0	1.3	2.1	1.0	0.0	1.4	0.2	1.7	6.0	1.1
N30V3	0.0	2.0	0.0	1.4	3.0	1.2	0.0	1.7	1.5	1.3	2.1	1.0
N32V3	0.0	3.1	0.0	1.5	0.9	1.3	0.0	1.7	1.2	1.9	3.0	1.1
N35V4	0.0	6.6	0.0	4.8	5.7	2.5	0.1	4.4	1.6	3.3	5.3	2.4
N37V4	0.0	7.0	0.0	4.4	2.7	2.9	0.1	5.9	0.1	3.8	2.8	2.9
N39V4	0.0	7.9	0.0	7.0	4.5	3.4	0.1	3.7	1.3	3.8	5.6	3.2
N41V4	0.0	24	0.0	14	8.9	5.7	0.0	10	0.0	8.9	7.3	4.6
N43V5	0.0	128	0.0	12	0.2	5.8	0.2	86	0.7	198	4.4	4.9
N45V5	0.0	224	0.0	90	5.9	18	0.1	192	1.3	179	9.7	22
N47V5	1.8	3600	1.8	3600	7.0	115	0.3	3600	2.6	2964	9.1	89
N49V5	0.1	3600	0.1	104	7.8	96	0.1	990	0.1	1364	7.7	26
N51V6	1.2	3600	1.2	129	2.9	52	1.7	1145	1.8	884	5.4	45
N53V6	1.0	3600	1.0	3600	3.3	126	1.0	3600	1.4	3600	7.3	81
N55V6	0.4	3600	0.4	3600	10.2	306	0.5	3600	1.5	3600	15.1	75
N57V6	2.4	3600	2.4	3600	5.6	3600	2.5	3600	3.1	3600	8.0	34

from configuration 4. However, this does not rule out the ALNS as a method for this problem type. Again, these constraints speed up the computational time and might serve as a method to find good paths, although sub-optimal when applied in a MIP setting.

10.2 Description of Problem Instances

For the remaining computational study presented in the following sections, the problem instances presented in table 10.3 will be used.

Table 10.3: Problem instances used for testing

Problem instance	Nodes			Vessels
	Pickup nodes	Fixed nodes	Spot nodes	
N32_V3	18	14	3	3
N41_V4	23	18	4	4
N71_V8	38	33	7	8
N87_V10	46	41	9	10
N105_V12	55	50	10	12

10.2.1 Disruption of Problem Instances

Disruptions are categorized into blocks and are applied to all problem instances introduced in table 10.3. That is, each study in the following sections corresponds to one or several model runs for each of the five problem instance sizes, and for each block included. Disruptions considered in this thesis are starting positions for vessel and/or allocated volume at liquefaction ports. Disruptions are classified as either low, medium or high. For initial vessel position, low, medium and high disruption corresponds to μ-values of 48 hours, 96 hours and 144 hours, respectively. For allocated volume, low, medium and high disruption corresponds to σ-values of 5%, 10% and 15%, respectively. The disruption characteristics for different blocks are shown in table 10.4.

Table 10.4: Disruption categorization - initial position and allocated volume (AV)

Instances in block X	Block 1 (B1)		Block 2 (B2)		Block 3 (B3)		Block 4 (B4)		Block 5 (B5)	
	Disruption		Disruption		Disruption		Disruption		Disruption	
	Init. pos.	AV	Init. pos.	AV	Init. pos.	AV	Init. pos.	AV	Init. pos.	AV
BX_N32_V3 BX_N41_V4 BX_N71_V8 BX_N87_V10 BX_N105_V12	MED	-	-	MED	LOW	LOW	MED	MED	HIGH	HIGH

10.3 ALNS

10.3.1 Tuning of the ALNS Parameters

The ALNS heuristic includes a considerable amount of parameters that might affect the performance of the algorithm. Testing all the possible parameter configurations in order to determine the optimal parameter values is not practicable. Instead, a sequential calibration strategy of the most important parameters is used; first, initial values for the parameters are decided based on Ropke and Pisinger (2006) and preliminary testing during the development phase of the algorithm. The preliminary testing indicated that some parameters affected solution quality and computational time more than others. These parameters are tuned sequentially, in pairs using grid search, in order of their deemed relative importance. For each parameter, the value is varied within a predefined range, while all other parameters are kept fixed. Once a parameters is tuned, its value is fixed for the rest

of the calibration procedure. The tuning is deliberately not performed on the same instances used in the study in Section 10.3.2 to avoid over-fitting the parameters to the instances used in these studies. All problem instances were solved five times for each variation of the parameters, with a maximum running time, T^{MAX}, of 3600 seconds and maximum number of iterations, I^{MAX}, of 10 000. The criterion for deciding the best parameter setting is primarily the average objective value gap. This gap is calculated based on the optimal objective value found by solving the MIP model for instances where the MIP is able to find optimal values within 3600 seconds. For problem instances where the MIP is unable to find optimal solutions, the gap is calculated based on the best upper bound from the MIP after an hour of running. This way, the larger instances are likely to be credited the most, as their gaps are likely to be larger than the gaps of the smaller instances. If a parameter setting has a substantial effect on the computational time of the ALNS, a trade-off evaluation of solution quality and computational time is used as decision criterion.

As described in Section 6.1, the ALNS contains a local search component. Since the LS components can have various configurations, the inclusion of this components adds several parameters to the overall ALNS algorithm. The ALNS mechanism is considered the main component and the LS is an improvement procedure. Therefore, the ALNS is first calibrated without the LS component. Next, testing is done on the ALNS with local search to determine the best configuration of the local search parameters. The ALNS is initialized using the construction heuristic described in Section 6.3.

The initial values of the parameters are presented in Table 10.5. λ, M, I^R, σ_1, σ_2, σ_3 and σ_4 are based on values used in Ropke and Pisinger (2006). These values were suitable for the heuristic, and preliminary testing indicated that changing them did not significantly impact the performance of the ALNS. The initial weights of the destroy and repair operators, $\omega_{d,0}$ and $\omega_{r,0}$, were all set to 1. With these initial weights, in combination with the score and decay parameters chosen above, the ALNS was able to adequately adjust the probability of choosing an operator according to its prior performance. ρ, ν_1, ν_2 and ν_3 are set based on preliminary testing. α, β, Γ^{MIN}, Γ^{MAX}, U^{INIT} and c, as well as the local search configuration, are calibrated using the strategy described above. The initial values of α and β are derived from preliminary testing and general knowledge about the problem. The initial values of the simulated annealing parameters, U^{INIT} and c, are obtained from Ropke and Pisinger (ibid.). Γ^{MIN} and Γ^{MAX} are the first parameters to be calibrated, so no initial values are set.

Table 10.5: Overview of ALNS parameters and their initial values

Parameter	Description	Value	Subject for tuning
I^{max}	Maximum number of iterations	10 000	-
T^{max}	Maximum running time [s]	3 600	-
α	Scaling factor for cost of infeasible node pair	1.5	Yes
β	Breach cost scaling factor	1	Yes
Γ^{max}	Upper limit of degree of destruction	-	Yes
Γ^{min}	Lower limit of degree of destruction	-	Yes
p	Determinism parameter	3	-
ν_1, ν_2, ν_3	Weights used in Shaw Removal	0.3, 0.4, 0.3	-
λ	Decay parameter for operator weight adjustment	0,1	-
w_o	Initial weights for destory operators	{1,1,1,1}	-
w_o	Initial weights for repair operators	{1,1,1,1}	-
M	Segment size	100	-
I^{reset}	Number of iterations before weights are reset	1000	-
σ_1	Score when new global best solution is found	33	-
σ_2	Score when new solution is better than current solution	13	-
σ_3	Score when new solution is worse than current, but accepted by SA	9	-
σ_4	Score when solution is worse than current and not accepted by SA	0	-
U^{init}	Initial temperature in simulated annealing	0.05	Yes
c	Cooling factor in simulated annealing	0.99975	Yes

Degree of Destruction

The degree of destruction will intuitively have a large impact on the performance of the ALNS. If it is too small, the ALNS will lose the advantage of the large neighborhood search and might not be able to sufficiently explore the solution space. An excessively large degree of destruction will more or less be a re-construction of the solution from scratch in each iteration, which may be time consuming and produce poor results (depending on the repair operator used to re-build the solution). As described in Section 6.6, the degree of destruction is chosen randomly within a range between Γ^{MIN} and Γ^{MAX} in each iteration. The results from the tuning of these parameters are presented in Table 10.6. As the degree of destruction is likely have a significant impact on the computational time of the ALNS, both average gap and average computational time is evaluated.

As expected, the degree of destruction has a large impact on the performance of the algorithm. The results indicate that both choosing between too large values and too small values yield poor results in terms of optimality gap. Having a large range of the degree of destruction is neither favorable. A clear trend of decreasing computational times when either bound of the degree of destruction is decreased

Table 10.6: Simulated annealing - overview of parameter combinations subject for tuning

	$\Gamma^{min} = 0.1$		$\Gamma^{min} = 0.15$		$\Gamma^{min} = 0.2$		$\Gamma^{min} = 0.25$	
	Gap [%]	Comp. time	Gap [%]	Comp. time	Gap [%]	Comp. time	Gap [%]	Comp. time
$\Gamma^{max} = 0.2$	8.8	413.3	7.3	421.7	5.7	427.3	-	-
$\Gamma^{max} = 0.3$	9.0	484.5	5.3	450.1	6.4	490.1	6.2	474
$\Gamma^{max} = 0.4$	5.9	457.1	10	428.9	6.5	454.5	11.6	502.2
$\Gamma^{max} = 0.5$	5.9	712.6	8.9	678.7	11.2	690	14.4	703.7

is also revealed. The exceptions to this trend can be explained by the fact that more feasible solutions are found during the ALNS when Γ^{MIN} and Γ^{MAX} are incrementally decreased, which means that the MIP in the second part of the ALNS is run more times, resulting in higher computational time. The best gap is found for $\Gamma^{MIN} = 0.15$ and $\Gamma^{MAX} = 0.3$. $\Gamma^{MIN} = 0.2$ and $\Gamma^{MAX} = 0.2$ give a slightly larger gap, but lower computational time. Still, the difference in computational time is not large, and having a varying degree of destruction is considered favorable, so the former parameter setting is selected.

Penalty Costs

Next, the penalty cost scaling factors, α and β, are tuned. The trade-off between these parameters will have a large influence on how the search is guided, which might result in a substantial impact on the performance in the second part of the ALNS. If for instance the breach cost is too high compared to the cost of having an infeasible pair of nodes, the algorithm might have trouble with guiding the search back to the feasible region. A too large α compared to β restricts the search too much, which can make it difficult for the repair methods to reach a wide variety of solutions and sufficiently explore the solution space. Therefore, these parameters are tuned using a grid search. The absolute values of the two parameters are also important as they determine the trade-off between the penalty costs and the non-penalty costs in the ALNS. These parameters do not have a noteworthy impact on computational time, so only the average gap is used as criterion in the calibration. Table 10.7 shows that no single combinations of the parameters is an evident selection, but indicates that setting α to 1/10 of β could be a reasonable choice. α is set to 0.1 and β to 1, as this parameter combination yielded, however slightly, the lowest average gap in the tuning study.

Table 10.7: Penalty costs - overview of parameter combinations subject for tuning

	Gap [%]					
	$\alpha = 0.1$	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 1.5$	$\alpha = 2.5$	$\alpha = 4$
$\beta = 1$	4.6	5.2	4.8	5.7	6.8	6.3
$\beta = 10$	6.5	6.6	4.8	6.4	6.7	7.3

Simulated Annealing

The acceptance criterion in the ALNS is calibrated by doing a grid search for different values of the simulated annealing parameters U^{INIT} and c. Setting the U^{INIT} too high, will help the ALNS with diversifying the search, but if the cool down is too slow, the heuristic may have trouble intensifying the search, harming the performance. Having a too small initial temperature or cooling down too fast will essentially reduce the heuristic to a descent heuristic, increasing the chance of getting trapped in a local optima. Note that $U^{INIT} = 0$ means that only improving solutions are accepted in the ALNS. This setting of U^{INIT} is obviously independent of the cooling factor, c. The results presented in Table 10.8 shos that having no simulated annealing ($U^{INIT} = 0$) or rapid cooling ($c = 0.996$) gives a higher average gap than the other parameter combinations. This is likely because the ALNS gets stuck in local optima. Based on Table 10.8, $U^{INIT} = 0.01$ and $c = 0.99975$, giving the lowest average gap of 3.6, is selected.

Table 10.8: Simulated annealing - overview of parameter combinations subject for tuning

	$c = 0.996$		$c = 0.99975$		$c = 0.9999$	
	Gap [%]	Comp. time	Gap [%]	Comp. time	Gap [%]	Comp. time
$U^{init} = 0$	5.9	417.7	5.9	417.7	5.9	417.7
$U^{init} = 0.01$	8.4	467.3	3.6	429.7	3.7	412.4
$U^{init} = 0.025$	6.6	421.8	3.7	428.4	4	397.3
$U^{init} = 0.05$	6.5	427.1	3.9	425	5.1	394

Local Search

Finally, the local seach component of the ALNS is configured. The different configuration settings for the local search involve neighbor selection, as described in

Section 6.8, as well as the parameters I^N (maximum number of iterations without improving the best solution in local search before switching to ALNS), I^S (maximum number of iterations without improving the best solution in the ALNS before switching to local search) and the length of the tabu list, L. Initial testing indicated that the performance of the ALNS was not significantly dependent on L, as long as it was not too small, so $L = 8$ is selected. Table 10.9 shows that the first improvement strategy performs slightly better than the best improvement. $I^N = 50$ and $I^S = 400$ gave the lowest average gap using the first improvement strategy, so this parameter setting is chosen.

Table 10.9: Local search - tuning of maximum number of iterations before recourse

	Gap [%]					
	Best improvement			First improvement		
	$I^N = 50$	$I^N = 100$	$I^N = 200$	$I^N = 50$	$I^N = 100$	$I^N = 200$
$I^S = 100$	3.5	3.5	3.9	3.1	3.5	4.5
$I^S = 200$	3.7	3.7	3.5	3.6	3.7	3.5
$I^S = 400$	4.0	3.3	3.7	3.0	3.5	3.7

Final ALNS Parameter Configuration

The final setting for the ALNS parameters, including the parameters related to the local search component, after the tuning is completed is presented in Table 10.10. The local search component uses the first improvement strategy. For the remaining parts of the computational study, the ALNS is run in this configuration.

Table 10.10: Final ALNS parameter values

Parameter	Description	Value
I^{max}	Maximum number of iterations	10 000
T^{max}	Maximum running time [s]	3 600
α	Scaling factor for cost of infeasible node pair	1.5
β	Breach cost scaling factor	1
Γ^{max}	Upper limit of degree of destruction	-
Γ^{min}	Lower limit of degree of destruction	-
p	Determinism parameter	3
ν_1, ν_2, ν_3	Weights used in Shaw Removal	0.3, 0.4, 0.3
λ	Decay parameter for operator weight adjustment	0,1
w_o	Initial weights for destroy operators	{1,1,1,1}
w_o	Initial weights for repair operators	{1,1,1,1}
M	Segment size	100
I^{reset}	Number of iterations before weights are reset	1000
σ_1	Score when new global best solution is found	33
σ_2	Score when new solution is better than current solution	13
σ_3	Score when new solution is worse than current, but accepted by SA	9
σ_4	Score when solution is worse than current and not accepted by SA	0
U^{init}	Initial temperature in simulated annealing	0.05
c	Cooling factor in simulated annealing	0.99975
I^N	Max # of non-improving iterations in LS before switching to ALNS	400
I^S	Max # of non-improving iterations in ALNS before switching to LS	100
L	Length of tabu list	8

10.3.2 Results and Discussion

The results from solving the problem instances described in Section 10.2 is presented in this section. All problem instances were solved five times with a time limit of 3600 s and an iteration limit of 10 000. Table 10.11 reports the best and average optimality gaps, as well as the average computational time, for both of the initialization methods described in Section 6.3. The ALNS heuristic initialized with the ADP and the construction heurisitc will be denoted ALNS-ADP and ALNS-CH, respectively, for the rest of this chapter. The first two columns of Table 10.11 show the optimality gap and computational time of solving the MIP model, as a reference. The gaps are calculated as in Section 10.3.1. Table 10.12 shows the average gap of the MIP and the ALNS with both initialization methods after 250

s and 1000 s, as well as the final gap as a reference.

Table 10.11: Overall ALNS Performance after 1 hour

Problem instances	MIP		ALNS					
			Initialization method					
			ADP			Construction heuristic		
	Gap (MIP) [%]	Comp. time [s]	Gap (avg.) [%]	Gap (best) [%]	Comp. time [s]	Gap (avg.) [%]	Gap (best) [%]	Comp. time [s]
B1_N32_V3	0.0	1	0.1	0.1	194	0.6	0.0	200
B1_N41_V4	0.0	3	2.0	2.0	330	1.7	1.0	344
B1_N71_V8	7.6	3600	9.1	8.3	982	11.7	10.1	1214
B1_N87_V10	17.2	3600	17.0	16.1	1830	25.6	16.2	2022
B1_N105_V12	11.4	3600	17.9	17.2	1919	19.0	18.6	2007
B2_N32_V3	0.0	1	1.8	1.0	215	0.9	0.3	220
B2_N41_V4	0.0	5	1.1	0.0	351	0.8	0.0	368
B2_N71_V8	12.3	3600	11.8	10.2	1152	19.4	12.1	1348
B2_N87_V10	19.3	3600	21.9	13.5	2080	26.1	22.0	2222
B2_N105_V12	8.0	3600	9.4	9.3	2011	14.3	13.5	2128
B3_N32_V3	0.0	1	1.2	0.5	214	1.4	0.5	235
B3_N41_V4	0.0	5	1.9	0.6	343	1.4	0.3	381
B3_N71_V8	2.7	3600	23.5	19.8	1324	18.9	9.2	1350
B3_N87_V10	15.8	3600	37.6	25.6	1980	27.8	25.5	2324
B3_N105_V12	13.0	3600	17.1	16.4	2014	28.1	23.5	1873
B4_N32_V3	0.0	1	1.3	0.2	224	1.6	0.4	243
B4_N41_V4	0.0	4	0.0	0.0	341	0.1	0.0	380
B4_N71_V8	8.8	3600	27.3	24.8	1232	20.3	17.2	1210
B4_N87_V10	6.6	3600	24	16.1	2015	29.4	21.6	2268
B4_N105_V12	21.4	3600	19.9	19.9	1916	24.6	23.7	2035
B5_N32_V3	0.0	1	0.0	0.0	240	0.0	0.0	244
B5_N41_V4	0.0	5	0.9	0.5	345	0.8	0.0	361
B5_N71_V8	5.6	3600	14.9	10.7	1345	13.1	9.1	1380
B5_N87_V10	16.1	3600	31.1	24.2	2118	33.3	26.7	2113
B5_N105_V12	14.2	3600	10.9	9.8	2222	16.7	14.5	2099

Table 10.11 indicates that using the original ADP plan to initialize the first solution yields better results in terms of optimality gap than using the construction heuristic in most of the large problem instances, both considering the average and best solutions. The ADP-CH gave the best results on the smaller instances, although the difference in gaps between the ALNS-ADP and ALNS-CH are not large. The ALNS-ADP and ALNS-CH are fairly consistent, without large differences between the best and average solutions in many of the problem instances. Still, in some cases the difference is large, as for in instance B2_N87_V10 with an average gap of

21.9 and best gap of 13.5. The ALNS-ADP seems to be more consistent than the ALNS-CH. Table 10.12 shows that the average gap after 250 s is usually lower for the ALNS-ADP than for the ALNS-CH, especially for the larger instances. The difference is not as striking after 1000 s and at the end. Both the results concerning consistency and time to find decent solutions are not that surprising considering that the original ADP is likely to be close to a fairly good solution, with only a few modifications needed to handle the disruptions.

Table 10.12: ALNS performance for selected comp. times

Problem	Gap after 250s [%]			Gap after 1000s [%]			Final gap [%]		
instances	MIP	ALNS-ADP	ALNS-CH	MIP	ALNS-ADP	ALNS-CH	MIP	ALNS-ADP	ALNS-CH
B1_N32_V3	0.0	0.1	0.6	0.0	0.1	0.6	0.0	0.1	0.6
B1_N41_V4	0.0	2.0	1.7	0.0	2.0	1.7	0.0	2.0	1.7
B1_N71_V8	10.4	9.1	15.7	10.4	9.1	11.7	7.6	9.1	11.7
B1_N87_V10	76.9	22.5	36.4	17.2	18.8	26.4	17.2	17.0	25.6
B1_N105_V12	n/a	23.2	19.0	17.3	17.9	19.0	11.4	17.9	19.0
B2_N32_V3	0.0	1.8	0.9	0.0	1.8	0.9	0.0	1.8	0.9
B2_N41_V4	0.0	1.1	0.8	0.0	1.1	0.8	0.0	1.1	0.8
B2_N71_V8	12.3	18.3	23.4	12.3	11.8	19.7	12.3	11.8	19.4
B2_N87_V10	318	32.2	37.6	27.0	26.7	28.1	19.3	21.9	26.1
B2_N105_V12	165	9.4	14.9	32.0	9.4	14.3	8.0	9.4	14.3
B3_N32_V3	0.0	1.2	1.4	0.0	1.2	1.4	0.0	1.2	1.4
B3_N41_V4	0.0	1.9	1.4	0.0	1.9	1.4	0.0	1.9	1.4
B3_N71_V8	4.2	34.9	21.7	4.2	23.6	19.8	2.7	23.5	18.9
B3_N87_V10	n/a	41.2	48.6	15.7	39.5	32.9	15.8	37.6	27.8
B3_N105_V12	158	17.1	28.2	31.6	17.1	28.2	13.0	17.1	28.1
B4_N32_V3	0.0	1.3	1.6	0.0	1.3	1.6	0.0	1.3	1.6
B4_N41_V4	0.0	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.1
B4_N71_V8	8.8	38.8	25.6	8.8	27.4	20.4	8.8	27.3	20.3
B4_N87_V10	184	36.3	39.3	6.8	24.5	30.6	6.6	24.0	29.4
B4_N105_V12	173	19.9	30.9	801.2	19.9	24.6	21.4	19.9	24.6
B5_N32_V3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B5_N41_V4	0.0	0.9	0.8	0.0	0.9	0.8	0.0	0.9	0.8
B5_N71_V8	6.5	17.0	15.5	6.5	15.2	13.3	5.6	14.9	13.1
B5_N87_V10	284	52.7	45.6	17.4	32.7	33.5	16.1	31.1	33.3
B5_N105_V12	142	15.8	21.9	19.2	10.9	16.8	14.2	10.9	16.7

Computational time for the ALNS-ADP and ALNS-CH are fairly similar, however, the using the construction heuristic as initialization method gives slightly higher computational times a almost all cases. This might be because the ALNS with this initialization will find more feasible solutions, thus increasing the time used in the second part of the heuristic, solving a MIP model with fixed routes. Computational time increases drastically with increasing size of the problem instance. This is

mostly due to the repair methods in the ALNS increasing the time per iteration.

The last three columns of Table 10.12 show that the ALNS has trouble finding better solutions than the MIP. For the smaller instances, where the MIP model is solved to optimality, both versions of the ALNS are close to optimal, but not consistently able to find the optimal solutions. The ALNS-ADP is able to outperform the MIP on average for the two largest instances in some of the blocks, still, in some cases even the best solutions found by the ALNS is far worse than the solutions found by solving the MIP model. For example for instance B3_N87_V10, the MIP gap is 15.8 % while the average and best gap of the ALNS-ADP are 37.6 % and 25.6 %, respectively.

There are three likely reasons for these results. First, the ALNS might not get to run for a sufficient amount of time, and therefore not able to adequately explore the solution space. The results show that the computational time of the ALNS are only around 2000 s for the largest instances, while the MIP model is run for 3600 s. However, Table 10.12 shows that the improvements after 1000 s are small for the ALNS, and initial testing indicated that further increasing the number of iterations of the ALNS did not significantly impact its performance. A second reason could be that the ALNS gets trapped in local minima. The fact that the ALNS in most cases quickly finds gap values close to its final gap value could indicate that the diversification strategies used in the ALNS are not sufficient, leading the heuristic to have trouble escaping a local minimum once a fairly good solution is found. The third reason can be inferred from the results presented in Section 10.1. These results show that by making the same assumptions about heel-out, FOB-sale, delivered quantity and vessel speed as are made in the ALNS, and using the same partitioning of the problem as is used in the ALNS in a MIP model (Configuration 5 in Section 10.1), can for some instances give large gaps in the original problem, even though the sequential MIP model is solved to optimality. This puts an innate bias on the ALNS model in the sense that even though the ALNS has found an optimal or close to optimal solution, given its assumptions, the routes in this solution might not be performing well in the original problem. The large difference in the final optimality gap between the ALNS and the MIP model for some of the instances could be explained by this bias.

An interesting way of assessing the performance of the ALNS is therefore to compare its results to the MIP model with the same assumptions and partitioning as the ALNS (Configuration 5 in Section 10.1). Table 10.13 shows the optimality gaps (calculated as before) from Configuration 5 and the ALNS on the problem instances used in Section 10.3.1. Configuration 5 is solved to optimality. The table shows that the ALNS outperforms Configuration 5 on all instances, even though

the ALNS is a heuristic version of it. This can be explained by the fact that the ALNS stores the best feasible solutions it finds, and one of the solutions that did not perform the best in the first part of the heuristic gets a higher objective value in the second part of the problem. This corroborate the hypothesis that was behind this modelling choice of the ALNS, and could indicate that the reason for the MIP (in Table 10.11 and 10.12) to outperform the ALNS is more likely due to the bias and not the ALNS getting trapped in local minima.

Table 10.13: Comparison of ALNS and the MIP, where the MIP makes similar assumptions as the ALNS (C5)

Problem instance	Gap [%]	
	MIP - C5	ALNS
N17_V2	5.1	0.2
N28_V3	6.0	0.6
N37_V4	2.8	0.9
N43_V5	4.4	2.9
N53_V6	7.3	2.7
N59_V7	16.3	11.1
N69_V8	14.6	7.0

Even though the ALNS has trouble finding better solutions than the MIP, the time to find decent solutions for large instances is significantly lower for the ALNS (both ALNS-ADP and ALNS-CH), as can be seen from Table 10.12. The MIP is outperformed by the ALNS methods on the two largest instances in all blocks when considering the average gap after 250 s. In many of the cases, the objective value found by the ALNS-ADP after 250 s is not very far from the best objective value found by the MIP after 3600 s. However, in some cases the solution found by the ALNS methods after 250 s is still poor compared to the best known solution.

The type and degree of disruption in the problem instances does not seem to significantly impact the performance of the ALNS, as there are no clear trends across the blocks for neither the ALNS-ADP nor the ALNS-CH.

10.4 Fix and Optimize

In this section, testing for the Fix and Optimize matheuristic is presented. First, preliminary studies are presented in Section 10.4.1. Second, a study on parameter tuning is presented in Section 10.4.2. Third, FO model performance compared to MIP performance is presented in 10.4.3.

10.4.1 Preliminary studies

Fixed Speed

In Section 10.1 it was shown that simplifying the MIP by postponing some of the decisions better computational speed can be achieved. It was shown that assuming maximum vessel speed when finding vessel routes, before re-introducing variable speed later is seems to be effective for the problem type considered in this thesis. In all cases tested, it is possible to make this simplification while still guiding the model to the same optimal vessel routes as found using the extensive MIP. For this reason, the FO imposes maximum vessel speed in the iterative phase before re-optimizing with variable speed later on. When solving the FO using fixed speed, the number of iterations performed increased on average by $\sim 40\%$ for large problem instances. The results are presented in Table 10.14. Although stability of the percentage improvement for larger problem instances are very sensitive to changes in the data, one must remember that the data presented here is the mean value of 10 model runs and that the results are consistent. Furthermore, iterations for large problem instances takes more time than for small problem instances, and if we are able to increase the number of iterations incrementally, without making sacrifices with respect to solution quality, this brings significant value to the FO. On average, the solutions found at the end with FO was equally good using both fixed and variable speed, similarly as we observed for the MIP.

Table 10.14: FO with and without simplifying using maximum vessel speed

		Variable speed	Fixed speed	
	Avg. total time spent [s]	Avg. # of iterations	Avg. # of iterations	% improvement
B1_N32_V3	1000	596	831	39 %
B1_N41_V4	1000	248	335	35 %
B1_N71_V8	1000	16	22	38 %
B1_N87_V10	1000	10	14	40 %
B1_N105_V12	1000	8	11	38 %

Neighborhood Testing

One alternative to the FO is to fix some of the vessel legs found through a construction heuristic or a simplified MIP without telling the heuristic exactly which legs to fix. Adding a new constraint to the MIP which sums all the legs that make up the routes and then requiring that they sum up to the number of legs is the same as requiring that they should all be equal to one. Instead, flexibility can be added by requiring that the vessel legs sum up to for instance x% of the total

number of legs without telling the model which legs to include and which to not include. By using this constraint, the final solution is required to be very close to the path found already, but at the same time the model is allowed to make some changes with respect to the routing decisions. This approach is somewhat similar to local branching techniques. It turns out that the constraint proposed does not constrain the model as much as believed beforehand, and the computational times are scale badly. The results are not included. However, the solutions, when found, were excellent. These findings indicate that this type neighborhood work well, although it is not exactly the same neighborhood as the one used for the FO. This may indicate that using a MIP to search around an existing solution might be effective, and that maybe a fix and optimize heuristic will solve the problem effectively.

10.4.2 Model Tuning

In this section, parameter tuning for the FO is considered. Parameter values are chosen prior to model execution; however, when the term ”real-time parameter tuning” is used, it concerns the dynamicity of the model and the feature of self-tuning in during the neighborhood search.

Sending Instructions During the Destroy Phase

As described in Chapter 7 the model presented in this thesis makes use of the Shaw destroy method, while during some iterations instructing the model to initiate the destroy phase by starting at a specific node (hereby referred to as the root node) in the vessel routes. This root node has the lowest destruction count up until the current time. Parameter settings include that this feature is applied every x iteration, but only after the first y iterations have been performed such that some statistics may be generated for the first y iterations before regularly instructing the destroy method afterwards. Initial testing indicates that parameter values $x = 3$ and $y = 10$ works well. Performance testing shows that the ”instruction” feature is often successful for two reasons. First, it often leads to new, improving solutions. Further testing shows that when analyzing the node sequence before and after this instruction has been applied, the node sequence around the root node has often been altered. This again strengthens the belief that instructing the destroy procedure contributes to the model performance. A performance test also confirms this, as the results were significantly better, and faster, when instructing or guiding the destroy procedure regularly. The instructions sent to the destroy method is believed to not adversely affect the nature and the randomness of the Shaw destroy method. The complete destroy procedure destroys large parts of

a solution, while the instructions sent to the procedure only affect a few nodes. In essence, the instruction feature is making sure that the time it takes until the heuristic has destroyed incoming and outgoing legs for all nodes is not too long. One counter argument is that sooner or later the destroy method will, due to the randomness, destroy all nodes in the set of vessel routes. However, the FO is not performing as many iterations as many other iterative heuristics, and because of this it does not make sense to rely on such statistics and the law of large numbers in this case. A summary of the test results can be found in 10.15. For the two problem sizes included, N87_V10 and N105_V12, the average number of iterations are 25 and 40, and the average total number of iterations where instructions are given to the destroy procedure are 6 and 8. Initial testing indicates that an instruction frequency of $\frac{1}{3}$ is appropriate, meaning that an instruction will be sent to the destroy procedure every third iteration after the first 10 iterations have been performed. The model is believed not to be very sensitive with respect to the exact value of this frequency parameter, but lower frequencies (or excluding the instruction feature) are found to perform worse.

Table 10.15: Percentage of successful instructions sent to the destroy method in FO

Problem instance	% effective instructions
B1_N87_V10	0 %
B2_N87_V10	20 %
B3_N87_V10	13 %
B4_N87_V10	0 %
B5_N87_V10	20 %
B1_N105_V12	13 %
B2_N105_V12	0 %
B3_N105_V12	14 %
B4_N105_V12	25 %
B5_N105_V12	38 %

Search Aggressiveness and Maximum Allowed Iteration Gaps

The implemented FO aggressively increases the destruction rate, guiding the program towards timeouts. The rationale for doing this is that timeouts are believed to be OK, and initial testing shows that the optimal parameter settings include frequent timeouts.

It is believed that there are three reasons for why the model should allow some timeouts. The first reason is that the model often an unnecessary large amount of time on closing the gap at the end of each iteration. The second reason is that even when the model face timeouts, it is often still able to find good, improving solutions. The third and probably most important reason is that the computation time might average on a fairly low level; however, one may observe large peaks for some iterations. These large peaks are a problem as they are both time-consuming and a potential source of over-fitting if not dealt with correctly. Time-consuming in the sense that the extra time spent on completing the iterations where these peaks occur does not seem to improve the model performance (i.e. just because a given iteration is complex and difficult to complete, it does not need to mean that this iteration is likely to lead to a, new, improving solution). By over-fitting it is meant that if the built-in tuning is too sensitive and the model observes a few timeouts, it might immediately respond by lowering the destruction rate or even increasing the maximum time limit for iterations. For this reason, if the iterations in general are very fast then having the model responding like this might not be a good thing. In these situations it might even be a good idea to increase the destruction rate while keeping the maximum time limit for iterations low.

Although it is found effective to allow timeouts, testing shows that it does not add additional benefit by increasing the threshold parameter beyond $\sim 1\%$. As described previously in 7, the threshold parameter controls whether the model should be tuned such that it destroys larger parts of the solution, or such that it increases the time budget for each iteration. The reason for $\sim 1\%$ being a suitable value for this parameter is perhaps that much of the time spent (and perhaps often wasted) at the end of each iteration is to close the final $\sim 1\%$ of the gap. When assessing the current status of the heuristic, whether the degree of destruction or the time budget needs to be changed, a gap is needed in order to compare the current status to this threshold. Therefore, the average of the 5 last iterations are used. Additionally, if there are gap values which are abnormally high, then these values are moderated in order to avoid having the heuristic overly sensitive, in correspondence with the discussion in the previous paragraph.

As described in Chapter 7, the *degree of destruction* and the *max iteration time* parameters are adjusted dynamically. For the heuristic implemented in this thesis, the degree of destruction and the iteration time budget are monotonously increasing. Other alternatives have been considered where for instance these parameters fluctuate up and down from iteration to iteration. By fluctuating, it is meant that for instance, the degree of destruction may be adjusted up because new solutions have not been found for a given number of iterations, but then it may be pushed down again due to timeouts while keeping the max iteration time parameter fixed.

The parameter tuning is supposed to change when, for instance, iterations are frequently stopped due to timeouts and the heuristic has not found new, improving solutions for a long time, then one possible option is to respond by increasing the max iteration time instead of lowering the destruction rate. Testing was performed on the use of non-monotonously changing parameters for real-time tuning, but as it shown little importance while at the same time adding complexity to the model and the tuning process, this feature was excluded from the model. Real-time parameter tuning is based on the two secondary parameters, the number of iterations since the last improving solution was found, and the gaps observed for recent iterations. Additionally, a minimum number of iterations are required in between two sessions of real-time tuning of parameters.

In the following, various tables including the results from parameter testing are presented. The testing is based on similar, although not the exact same, problem instances as the larger ones presented in Section 10.2. In Table 10.16 initial values for the degree of destruction and time budget are tested. As shown in the table, all the values presented here performs decent, indicating that the model is relatively in-sensitive of these parameter values; however, lower values than 15% and larger values than 30% are found less effective. Lower values still performs decent, because of the real-time tuning of the heuristic. However, larger values for the degree of destruction gives poor results, as the computational times explode and unless these initial values are quickly lowered during real-time tuning then FO performance becomes poor. Similar conclusions are made from Tables 10.17 and 10.18. Also here the heuristic has proven relatively in-sensitive to the values of this parameters within the ranges presented here. Gap threshold values up to 10% works fine in most cases, and most values for the maximum # of non-improving iterations also works fine in most cases (as long as the initial values for the degree of destruction and the time budget are reasonable. In essence, this indicates that real-time tuning is not very important. An implication from this is that it probably is a good alternative to only include a simple model with only two static parameters, the degree of destruction and the time budget. As long as the initial values for these are OK, then the results should be fine. Testing also suggests that this is the case. However, in some cases the dynamic model performs better and is able to find new solutions after long periods of only non-improving iterations, and these solutions are probably found due to the real-time tuning.

Tuning the FO is a cumbersome and complex task, and there is probably some improvement potential. As a disclaimer, it must be said that further tuning and testing might lead to other optimal values for these parameters. Testing on multiple real data sets might add additional value. However, testing shows that the model components themselves are very important (like for instance instructing the

Table 10.16: Initial values for degree of destruction and time budget for each iteration

(Degree of destruction, time)	Gap [%]		
	1000s	2000s	3000s
(0.15, 30)	9.6,	7.5,	7.3
(0.2, 30)	7.8,	7.3,	7.1
(0.25, 30)	9.7,	8.2,	7.3
(0.3, 45)	9.5	8.2	7.3

Table 10.17: Destroy more compared to solving longer - gap thresholds. Less than 1% was also tested, but it was found to not perform as well and the model became unstable

Gap threshold [%]	Gap [%]		
	1000s	2000s	3000s
1	8.5	7.2	7.2
2	8.5	7.3	7.3
3	8.5	7.5	7.3

Table 10.18: Maximum # of non-improving iterations before parameter tuning

Max. # of iterations	Gap [%]		
	1000s	2000s	3000s
2	8.1	7.6	7.2
4	8.5	7.4	7.2
6	8.7	8.0	7.2

Table 10.19: FO key parameter values

Parameter name	Initial value	Static or dynamic
Degree of destruction	20%	Dynamic
Allocated time budget for each iteration	30s	Dynamic
Degree of destruction - increment	5%	Static
Time budget - increment	20s	Static
Max # of non-improving iterations without tuning	5	Static
Number of look-back iterations	5	Static
Guiding frequency	$\frac{1}{3}$	Static
# of iterations until guiding commences	10	Static

destroy procedure regularly, and restricting model over-fitting when timeouts are observed), while the exact parameter values for the corresponding parameters are believed to be less important and also to some extent problem size specific.

Conclusion and Parameter Values

In conclusion, it is believed that searching the solution space aggressively despite facing a few timeouts is a good idea, supported by the aforementioned arguments. Initial testing shows that this is key in order for the model to succeed. Furthermore, initial testing shows that it is difficult to find optimal parameter values. The various tuning settings presented here are often equally good; however, for some of these tests the reasons for this is believed to be that the model is making real-time adjustments to the parameters when it believes that the search is unable to escape a local optima. If, on the other hand, the feature of self-tuning is turned off and all parameters values are considered to be static, then the initial values for the parameters are found to be much more important. Tests show that when keeping the parameter values static the model perform almost as good as when the parameters are dynamic. This of course, depends on the right static values being used. It was found that around 20-30% degree of destruction and a time budget of around 30-40s.

10.4.3 Model Performance - Results and Discussion

The overall goal for the LNS is that it is able to find the optimal solution in most cases. Of course, this is a desirable achievement, but if the LNS (only) outperforms the other alternatives, although not always being able to solve the problem instances to optimality, this is also desirable achievement.

The number of iterations spent for each model run of the FO are not included in the tables in this chapter. For the largest files, the FO completes ~ 100 iterations per 3 600 seconds when relatively a small degree of destruction is applied, while it completes ~ 40 iterations when a large degree of destruction ($> 25\%$) is applied. Even though the LNS is far away from performing as many iterations as most of the other destroy and repair methods, including the ALNS, it might still be effective if the program is both fairly quick while at the same time being to find improving solutions frequently.

For the remaining parts of this chapter, test results for the Fix and Optimize heuristic are summarised in tables. Each problem instance for each block has been solved ten times and all statistics for gaps and computational times presented in this section are based on mean values, except for the table columns marked as "best". Furthermore, gaps presented here are based on the upper bound obtained after 3600s for the MIP and not real-time gaps, and the gaps are calculated from $\frac{UB-OBJ}{OBJ} \cdot 100\%$. In the following, three tables summarizing the performance of the FO and the MIP are presented. Cells in the tables are marked with a green color when the given value is a top performer. First, Table 10.20 displays the average and best performance within 3 600 seconds for the different solution method alternatives is presented. Second, Table 10.21 focuses on the solution performance relative to time elapsed. Third, Table 10.22 presents information regarding when best solutions are found and when the models introduced in this thesis outperforms a MIP-solver, on average.

From Table 10.20 it can be observed that the Fix and Optimize heuristic is superior to the MIP both in terms of best recorded performance and average performance. The consistency of the FO is not explicitly proven here, but in general, a single run of the FO has been shown to outperform the MIP. As indicated by the average gaps in the table, the FO shows strong consistency. The fact that the average gaps and the best gaps for the FO is often very close to each other also indicates a low spread with regards to final gap after 3600s.

For the problem instances which the MIP is not able to solve to optimality, the FO almost finds a solution which is considerably better than the ones found by the MIP in all cases except one, problem instance $B3\_N71\_V8$ where the MIP and the FO perform equally good. As both the FO with an initial solution from an ADP and the FO with an initial solution from a construction heuristic seems to hit the exact same gap (2.7%) both on average and for the best iteration, this gap may correspond to the optimal solution. This is somewhat speculative, but it certainly sounds plausible.

There are cases (depicted in Figure 10.3) where the FO gets trapped in what

seems like local optima. Such local optima may also be observed for small problem instances. For example, for problem instance $B2\_N32\_V3$ the FO often miss the optimal solution by $\sim 1\%$ (shown as avg. 0.7% in the table, and further testing shows that the gaps for the specific situations when this happens are $\sim 1\%$). This observation is very interesting, as it means that even for a simple problem instance as the one considered here, together with a degree of destruction of $\sim 30 - 40\%$ and >1000 iterations, it is possible that the FO is unable to find the optimal solution. Such cases seems to be rare, but they indicate that when using a MIP-solver to search the neighborhood defined by the destroy procedure in the FO, this neighborhood does not always connect the feasible region completely.

One final observation from Table 10.20 is that the two options for providing the FO with initial solutions seems to both give decent results, and it is difficult to conclude that one of them is better than the other.

As discussed already, the FO assumes a feasible solution as input. One alternative is to start the LNS from an ADP plan, and ignore any infeasibilities. The FO proceeds as if the solution is feasible and hope that a future destruction will lead to a feasible solution. For instance, if an iteration fails to produce a feasible solution after destroying the ADP-plan, then the next iteration will start over by destroying the ADP-plan. Initial testing shows that this alternative is poor compared to the other starting methods, and sometimes completely useless. Hence, this is not a viable alternative without destroying the ADP more intelligently. For this reason, it option is not included when testing the FO.
From Table 10.21 it can be observed that the ADP-start leads to faster results than when using the construction heuristic, with the ADP-start often outperforming the MIP even during the first iteration in the FO. Moreover, we observe that it is difficult to say that one single way of starting the FO is better than the other. At first, it seems like the ADP is the better option, especially if the model is only to be solved a single time. If multi-start options and multiple computers are available, then we believe that it is better to include the construction heuristic for two reasons. The first reason is that the solution is then more likely to be further away from any local optimum, and thus the search for a local optimum will take more iterations than if the ADP was used, which again might lead to the FO searching larger parts of the solution space. The second reason is that the construction heuristic includes a random component, while the ADP does not. If the ADP is very close to a local optimum, then this local optimum might be difficult to escape from. This discussion is supported by the previous table, Table 10.20 where the "Gap (best) [%]" column seems to be in favour of the Construction Heuristic. Although there are many green (positive) cell markings for the ADP-start in the table, the ADP only barely outperforms the Construction Heuristic once, while

Table 10.20: Overall Fix and Optimize Performance after 1 hour

Problem instances	MIP		Fix and Optimize					
			Initialization method					
			ADP			Construction heuristic		
	Gap (MIP) [%]	Comp. time [s]	Gap (avg.) [%]	Gap (best) [%]	Comp. time [s]	Gap (avg.) [%]	Gap (best) [%]	Comp. time [s]
B1_N32_V3	0.0	1	0.0	0.0	20	0.0	0.0	17
B1_N41_V4	0.0	3	0.0	0.0	34	0.0	0.0	36
B1_N71_V8	7.6	3600	6.5	6.4	3600	6.9	6.5	3600
B1_N87_V10	17.2	3600	7.3	7.1	3600	7.3	7.1	3600
B1_N105_V12	11.4	3600	9.9	9.5	3600	8.8	7.8	3600
B2_N32_V3	0.0	1	0.7	0.0	1264	0.7	0.0	20
B2_N41_V4	0.0	5	0.0	0.0	30	0.0	0.0	31
B2_N71_V8	12.3	3600	5.6	5.6	3600	5.7	5.6	3600
B2_N87_V10	19.3	3600	4.9	4.8	3600	4.9	4.7	3600
B2_N105_V12	8.0	3600	4.0	3.8	3600	4.0	3.6	3600
B3_N32_V3	0.0	1	0.0	0.0	31	0.0	0.0	20
B3_N41_V4	0.0	5	0.0	0.0	30	0.0	0.0	65
B3_N71_V8	2.7	3600	2.7	2.7	3600	2.7	2.7	3600
B3_N87_V10	15.8	3600	9.2	9.2	3600	9.9	9.0	3600
B3_N105_V12	13.0	3600	9.7	9.5	3600	9.9	9.4	3600
B4_N32_V3	0.0	1	0.0	0.0	13	0.0	0.0	20
B4_N41_V4	0.0	4	0.0	0.0	31	0.0	0.0	47
B4_N71_V8	8.8	3600	7.0	6.9	3600	6.7	6.6	3600
B4_N87_V10	6.6	3600	4.5	4.5	3600	4.6	4.5	3600
B4_N105_V12	21.4	3600	11.0	10.6	3600	13.7	11.3	3600
B5_N32_V3	0.0	1	0.0	0.0	18	0.0	0.0	14
B5_N41_V4	0.0	5	0.0	0.0	35	0.0	0.0	37
B5_N71_V8	5.6	3600	4.7	4.3	3600	4.5	4.3	3600
B5_N87_V10	16.1	3600	7.2	7.0	3600	7.5	7.0	3600
B5_N105_V12	14.2	3600	6.1	5.7	3600	8.8	6.3	3600

for many problem instances the best found gap for the CH-start is significantly outperforming the ADP-start. Testing with longer computation times than what have been presented here also favours the CH-start when the model is allowed to run many times.

If however, the data set is highly disrupted (e.g. vessel breakdown) then the construction heuristic might work well or even better, since we build a new solution rather than destroying infeasibilities. If we have to destroy a lot in order to make the ADP plan feasible, then there are many variables which need to be included in the MIP, thus increasing the required computational time.

Table 10.21: Fix and Optimize performance for selected comp. times

Problem	Gap after 500s [%]			Gap after 1000s [%]			Final gap [%]		
instances	MIP	FO-ADP	FO-CH	MIP	FO-ADP	FO-CH	MIP	FO-ADP	FO-CH
B1_N32_V3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B1_N41_V4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B1_N71_V8	10.4	6.9	14.0	10.4	6.7	12.5	7.6	6.5	6.9
B1_N87_V10	19.8	8.8	12.0	17.2	8.1	7.4	17.2	7.3	7.3
B1_N105_V12	n/a	12.9	45.5	17.3	12.2	17.2	11.4	9.9	8.8
B2_N32_V3	0.0	0.7	0.7	0.0	0.7	0.7	0.0	0.7	0.7
B2_N41_V4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B2_N71_V8	12.3	5.7	5.7	12.3	5.6	5.7	12.3	5.6	5.7
B2_N87_V10	27.1	5.2	5.1	27.0	5.2	5.0	19.3	4.9	4.9
B2_N105_V12	n/a	5.4	5.7	32.0	4.8	4.6	8.0	4.0	4.0
B3_N32_V3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B3_N41_V4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B3_N71_V8	4.2	2.8	6.3	4.2	2.7	2.7	2.7	2.7	2.7
B3_N87_V10	56.1	15.1	17.2	15.7	9.5	11.1	15.8	9.2	9.9
B3_N105_V12	51.1	10.3	65.9	31.6	10.2	20.2	13.0	9.7	9.9
B4_N32_V3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B4_N41_V4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B4_N71_V8	8.8	10.1	10.2	8.8	7.3	7.6	8.8	7.0	6.7
B4_N87_V10	8.4	6.1	17.9	6.8	5.5	8.8	6.6	4.5	4.6
B4_N105_V12	n/a	13.1	49.9	n/a	12.9	37.3	21.4	11.0	13.7
B5_N32_V3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B5_N41_V4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B5_N71_V8	6.5	9.2	6.7	6.5	5.0	4.7	5.6	4.7	4.5
B5_N87_V10	21.2	11.8	20.1	17.4	10.5	10.1	16.1	7.2	7.5
B5_N105_V12	81.8	9.7	54.4	19.2	8.4	29.2	14.2	6.1	8.8

Tests not included here shows that the Construction Heuristic often produces initial solutions far away from the optimal solution, both in terms of objective value and variable values. Table 10.21 indicates that even if the initial solution is relatively poor compared to starting from an initial solution from an ADP plan, it is still be easy for the FO to iterate its way to better solutions, instead of trying to solve everything simultaneously with a MIP. Figures 10.2 and 10.3 are presented later also supports this argument.

Table 10.22: Fix and Optimize time to outperform MIP

Problem instances	Time when best solution is found [s]			Time to outperform MIP [s]	
	MIP	FO-ADP	FO-CH	FO-ADP	FO-CH
B1_N32_V3	1	17	18	-	-
B1_N41_V4	3	42	45	-	-
B1_N71_V8	3591	1811	2926	93	1144
B1_N87_V10	893	3342	2545	0	316
B1_N105_V12	3572	3172	3252	1178	1563
B2_N32_V3	1	19	19	-	-
B2_N41_V4	5	29	28	-	-
B2_N71_V8	2597	1295	1709	0	0
B2_N87_V10	3545	2088	2805	0	0
B2_N105_V12	3584	3131	3186	0	0
B3_N32_V3	1	24	22	-	-
B3_N41_V4	5	29	52	-	-
B3_N71_V8	3507	1674	1020	594	761
B3_N87_V10	3587	3165	3467	562	427
B3_N105_V12	1554	3566	3553	0	1355
B4_N32_V3	1	14	22	-	-
B4_N41_V4	4	30	51	-	-
B4_N71_V8	2474	2885	2896	519	482
B4_N87_V10	3117	2780	3448	370	1114
B4_N105_V12	1885	3402	3605	0	1676
B5_N32_V3	1	14	19	-	-
B5_N41_V4	5	31	35	-	-
B5_N71_V8	3594	1991	1733	547	428
B5_N87_V10	2851	2221	3435	88	709
B5_N105_V12	3556	3076	3331	0	1721

From Table 10.22 two main observations can be made. First, we observe that for large problem instances, both the MIP and the FO often finds new improving solutions throughout computation time budget. Second, it can again be observed that the FO starting from an ADP often outperforms the best solution found using the MIP early or even immediately. Note: setup-times are the same for both the

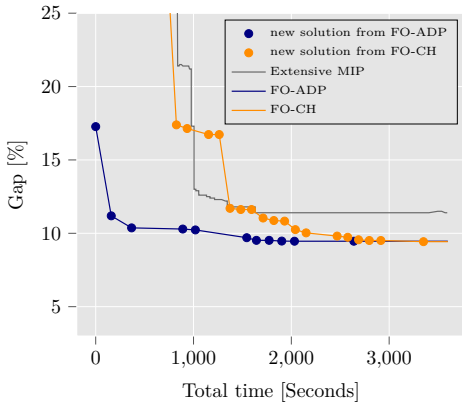
MIP and the FO, and they are excluded from the analysis. Therefore, computation times down to 0 seconds are observed.

Visual Analysis - Plotting of Solution Gap and Time Spent

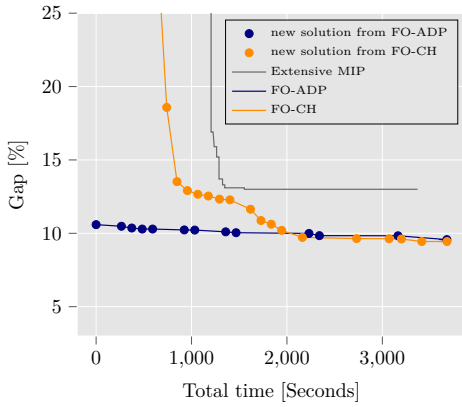
In Figures 10.2 and 10.3 solution method performance with respect to gap is plotted against time spent for each method. The plots depicted in Figure 10.2 are randomly chosen and should represent the typical situation well, while Figure 10.3 depicts selected model runs in order be able to comment on specific interesting findings.

As previously seen from Table 10.21, it can again be observed strong FO performance despite slow starts. In 10.2a and 10.2d it can be observed that a point in time the FO using a constructed feasible initial solution is actually performing worse than the MIP; however, after some time the FO effectively searches and finds new, improving solutions.

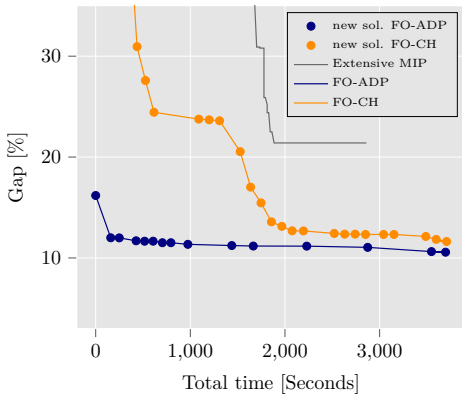
From the plots in Figure 10.2 it appears that the greater the total disruption (10.2b to 10.2d), the further away the (destroyed) ADP-start is from the optimal solution, as expected. In general, the ADP-start seems to lead to finding good, feasible solutions faster than for the construction heuristic. Similar to what has been observed earlier from Table 10.21. Furthermore, the figure underpins FO as a method regardless of how the initial solution is found. This is not to say that the initial solution is not important, but the method seems to often perform well also when the initial solution is very different from the optimal solution. This difference is difficult to show graphically, but in the Appendix Section A.1 then Table A.1 shows an example of this happening for a large problem instance. This is not to say that the local optima found by the FO are always good solutions, and shortly Figure 10.3a depicts this occurrence.



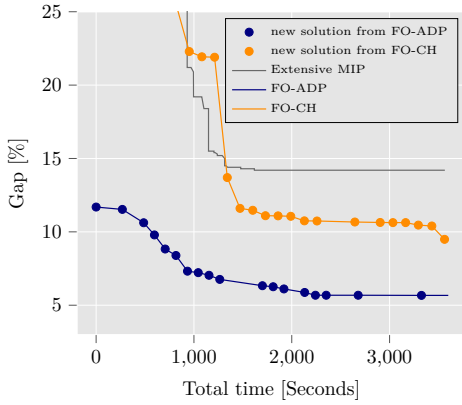
(a) Test runs for file B1_N105_V12



(b) Test runs for file B3_N105_V12



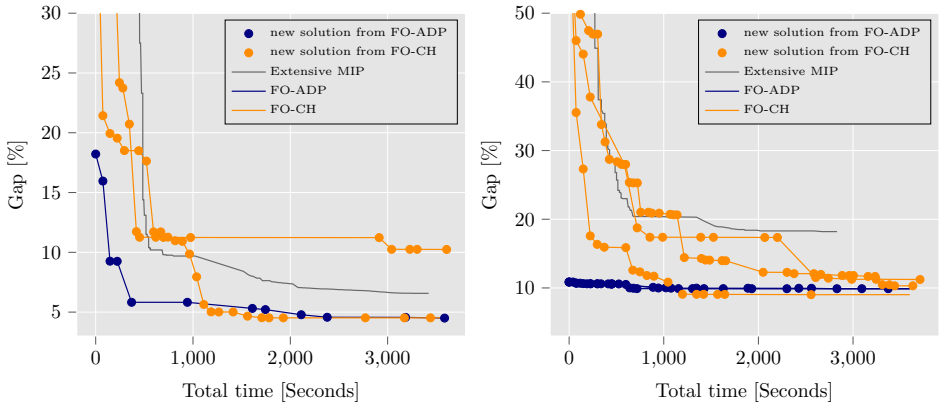
(c) Test runs for file B4_N105_V12



(d) Test runs for file B5_N105_V12

Figure 10.2: Test results for randomly selected model runs for problem size N105_-V12 for blocks 1, 3, 4, 5.

Local optima can be observed for both the ADP and the Construction Heuristic. Figure 10.3a depict this for the Construction Heuristic. In general, starting the FO from the Construction Heuristic seems to provide better diversification than ADP start. An example of this feature is presented in figure 10.3b.



(a) Multiple runs for file B4_N87_V10 showing a local optimum from FO outperformed by the MIP

(b) Three selected test runs for both ADP-start and CH-start for file B2_N87_V10

Figure 10.3: Selected gap vs. time developments for the FO heuristic

10.5 Comparison of the ALNS and the Fix and Optimize Heuristics

In this section a final comparisons of the two methods for solving the short-term LNG routing and scheduling problem is presented. The average gaps at various computational times for the two heuristics are shown in Table 10.23. The numbers presented here are using the ADP-start for both heuristics. It was not found necessary to also include the construction heuristic as it tells the same story. The results are clearly in favor of the FO. The reason for choosing the ADP-start was that perhaps the ALNS was able to outperform the FO at 250 and 500 seconds, but this is not the case.

It is believed that there are two main reasons for the FO outperforming the ALNS. The first reason is that the ALNS is biased as discussed in Section 10.3.2, and based on postponing decisions. Although the performance of the ALNS is better than the a MIP having the same assumptions (configuration 5 in Section 10.1), it is still a source of poor guidance in the search. The second reason is that the FO is performing very good. Many FO heuristics struggle since the repair procedure is relatively expensive, compared to other large neighborhood search methods. The

FO presented in this thesis is fast, and even when the initial solution is far away from any good solutions (in the solution space) the heuristic is able to iterate its way to good solutions in a reasonable amount of time ($< 1000s$). The FO performs a relatively low amount of iterations, but since the heuristics allow for large values for the degree of destruction parameter even for large problem instances, then for each iteration it is more likely to observe a new "best" solution than observing a non-improving solution. In addition, the novel destruction procedure used by the FO where the destruction phase is guided, has shown to be very effective.

Table 10.23: Comparison of ALNS and Fix and Optimize - initial solution from ADP has been used

Problem instance	Gap after 250s [%]		Gap after 500s [%]		Gap after 1000s [%]		Final gap [%]	
	FO	ALNS	FO	ALNS	FO	ALNS	FO	ALNS
B1_N32_V3	0.0	0.1	0.0	0.1	0.0	0.1	0.0	0.1
B1_N41_V4	0.0	2.0	0.0	2.0	0.0	2.0	0.0	2.0
B1_N71_V8	7.4	9.1	6.9	9.1	6.7	9.1	6.5	9.1
B1_N87_V10	12.0	22.5	8.8	22.5	8.1	18.8	7.3	17.0
B1_N105_V12	16.2	23.2	12.9	21.7	12.2	17.9	9.9	17.9
B2_N32_V3	0.7	1.8	0.7	1.8	0.7	1.8	0.7	1.8
B2_N41_V4	0.0	1.1	0.0	1.1	0.0	1.1	0.0	1.1
B2_N71_V8	5.8	18.3	5.7	15.6	5.6	11.8	5.6	11.8
B2_N87_V10	5.3	32.2	5.2	31.3	5.2	26.7	4.9	21.9
B2_N105_V12	5.6	9.4	5.4	9.4	4.8	9.4	4.0	9.4
B3_N32_V3	0.0	1.2	0.0	1.2	0.0	1.2	0.0	1.2
B3_N41_V4	0.0	1.9	0.0	1.9	0.0	1.9	0.0	1.9
B3_N71_V8	7.3	34.9	2.8	31.2	2.7	23.6	2.7	23.5
B3_N87_V10	19.7	41.2	15.1	41.2	9.5	39.5	9.2	37.6
B3_N105_V12	10.6	17.1	10.3	17.1	10.2	17.1	9.7	17.1
B4_N32_V3	0.0	1.3	0.0	1.3	0.0	1.3	0.0	1.3
B4_N41_V4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B4_N71_V8	15.7	38.8	10.1	29.6	7.3	27.4	7.0	27.3
B4_N87_V10	14.0	36.3	6.1	28.7	5.5	24.5	4.5	24.0
B4_N105_V12	18.3	19.9	13.1	19.9	12.9	19.9	11.0	19.9
B5_N32_V3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
B5_N41_V4	0.0	0.9	0.0	0.9	0.0	0.9	0.0	0.9
B5_N71_V8	12.5	17.0	9.2	15.2	5.0	15.2	4.7	14.9
B5_N87_V10	16.2	52.7	11.8	40.9	10.5	32.7	7.2	31.1
B5_N105_V12	11.7	15.8	9.7	11.8	8.4	10.9	6.1	10.9

10.6 Simulation and Robustness Strategies

In this section, we present a computational study of the proposed robustness strategies presented in Section 7.3. First, a short description of the chosen problem instance and a schematic overview of the testing procedure are given. Second, two different aspects of evaluating different solutions are discussed. Third, the three robustness strategies are tested with different values. Finally, two approximations of the Pareto Frontier for the given instance are presented and compared.

The aim of this study is two-fold. The first one is to assess how exogenous events and activities like delay in sailing time and port unavailability may influence profit and violation of time constraints. The second one is to test if any of the robustness strategies proposed in Section 7.3 are able to enforce/guide the solution method to find more robust solutions compared with a base case where no robustness strategies are used.

10.6.1 Test Settings and Problem Instance

The framework has been programmed in Python 3.6. The part of the input of the framework which represents a solution is an output of an FO model programmed in Python and uses Gurobi as a solver. The tests were performed on a standard computer with a processor of Intel Core i7-7700 3.6 GHz and RAM of 32 GB.

Problem Instance

The problem instance used in this computational study is the instance B3_N73_V8 introduced in Section 10.2 as a part of block three which indicates a low level of disruption in origin and in allocated volume. The problem instance has been chosen based on what is considered realistic in terms of number of vessels, deliveries, and level of disruption. The main characteristics of the instance are described in Table 10.24.

Table 10.24: Problem instance characteristics

Instance	Nodes			Vessels	Block
	Pickup nodes	Fixed nodes	Spot nodes		
B3_N81_V8	40	34	7	8	3

Figure 10.4 shows a schematic overview of the testing procedure. The problem instance B3_N73_V8 is the input of the solution method FO, which produces a set of solutions in a deterministic setting. The problem instance and a solution constitute

a configuration which is the input to the simulation model. The configuration is simulated for 1000 times and performance estimates such as the mean of profit, mean of total time violation and confidence intervals are calculated. In the last step, solutions are evaluated based on the Pareto dominance relation and the bounding box test as described in Algorithm 11. Note that all evaluations and comparisons of profit and total time violations done in this section are based on simulated values.

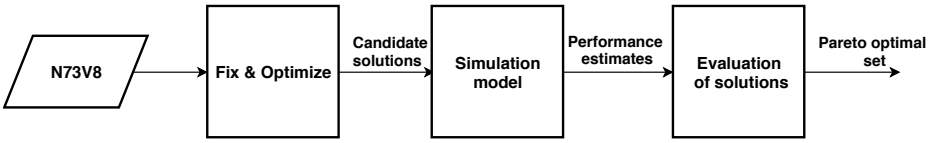


Figure 10.4: Schematic overview of the testing procedure

Solution Evaluation In this section, two different techniques are introduced to ensure a fair and efficient comparison of different solutions.

10.6.2 Variance Reduction

To obtain estimates of the objective value and the total time violations that are statistically representative of the true values, a solution is simulated 1000 times. The Mean is then used to provide an estimate of the true values. These estimates usually have an error variance of $\frac{\sigma^2}{n}$ where σ^2 is the sample variance, and n is the number of iterations. Hence, in order to increase the precision of these estimates, one has either to increase n or reduce σ. Considering a simulation "budget" with limited computing time, increasing n is undesired since a confidence interval is only reduced by a rate of $\frac{1}{\sqrt{n}}$ when n is increased. For this reason, the variance reduction technique Common Random Numbers is used. This technique is widely used in Monte Carlo simulations and is useful when two uncertain estimates, say for instance X_1 and X_2 are compared. An estimator of the difference between X_1 and X_2 is apparently $E(X_1 - X_2) = \bar{X}_1 - \bar{X}_2$. The intuition behind this technique

can be explained by the variance of the difference between the two variables

$$\mathrm{Var}(X_1 - X_2) = \mathrm{Var}(\frac{1}{n}\sum_{i=1}^{n} X_{1,i} - \frac{1}{n}\sum_{i=1}^{n} X_{2,i}) \tag{10.1}$$

$$= \frac{1}{n}\mathrm{Var}(X_1) + \frac{1}{n}\mathrm{Var}(X_2) - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{Cov}(X_{1,i}, X_{2,j}) \tag{10.2}$$

$$= \frac{1}{n}\mathrm{Var}(X_1) + \frac{1}{n}\mathrm{Var}(X_2) - \mathrm{Cov}(X_1, X_2). \tag{10.3}$$

As seen from the third term in the Equation 10.3, the variance is reduced if the two estimators are positively correlated. In practice, a positive correlation is introduced by using the same sequence of random numbers when two systems are simulated. However, a drawback of this method is known as the problem of synchronization, that is, how to assign random numbers when the two systems do not have the same structure or elements. This problem is present in our case since two different solutions are likely to have different arrangements of nodes and vessels. This problem is solved by generating the same random number seed for each pair of nodes and iteration, i.e. the random number generator is now a function of the node pair (i, j) and the iteration number. For instance, let's assume that two solutions, SOLUTION1 and SOLUTION2, are obtained from the FO algorithm and the solutions are simulated 1000 times. Then when a vessel v is assigned the same voyage in both solutions, the generated sequence of random numbers needed to simulate the voyage will be the same during iteration 1 in both solutions. In iteration 2, a new sequence is generated and is used in simulating the voyage in both solutions. This continues until iteration 1000 is carried out.

10.6.3 Ranking and Selection

In this part of the thesis, we discuss the issue of ranking a finite set of solutions and selecting the best of them, where best is defined with respect to the highest mean of profit and the lowest mean of total time violation, and where the performance of each solution is estimated by simulation. Accordingly, profit and total time violation constitute a multi-objective problem where both objectives apparently are in conflict since minimizing the total time violation incurs costs which reduces profit. Profit and total time violation are also expressed in different measurement units that are difficult to convert into one common unit. For these reasons, ranking and selection are not straight-forward in this case.

Pareto Dominance

For comparing solutions, we use the Pareto dominance relation, which results in a set of solutions that represent different trade-offs between the objectives. Hence, the ranking and selection procedure can be divided into two steps: 1) selecting a set of non-dominated solutions 2) choose a solution according to a set of preferences. The last step is left to the decision-maker. A solution x is said to dominate solution y if x is strictly better than y in one of the objectives and is no worse than y in the other objectives. For a maximization problem, this is mathematically expressed as

$$x \succ y \quad iff \begin{cases} \forall i \in 1, 2, \ldots, k & f_i(x) \geq f_i(y) \\ \exists j \in 1, 2, \ldots, k & f_j(x) > f_j(y). \end{cases} \tag{10.4}$$

Additionally, a Pareto optimal solution x is defined such that there does not exist another solution y that dominates it. However, a solution x is called a weakly Pareto optimal if there is no other solution y such that $f_i(y) > f_i(x) \forall 1, 2, \ldots k$. Accordingly, the set of non-dominated solutions is defined as

$$\wp^* = \{x_i \in X \mid \nexists y \in X : y \preceq x\} \tag{10.5}$$

as given by Jaimes et al. (2009).

Uncertainty in Selection

Since the performance of solutions is uncertain, it is not possible to infer if solution A dominates solution B without using a probability of correct selection. For this reason, confidence interval (CI) with a confidence level of 95 % is used. CI is constructed such that the unknown true value we want to estimate may be within this range with a probability of 95%.

Figure 10.5 shows the profit mean and the associated confidence interval for four solutions. Using only the estimated means, it is easy to rank the solutions, however, this is statically invalid since the confidence intervals of these solutions are overlapping which makes it impossible to infer which solution is the best in absolute term.

Considering the performance of a solution estimated by two measures (profit and total time violation), we get a two-dimensional confidence interval for each solution. Such a representation of uncertainty is called Bounding Box (BB) (Mlakar et al., 2014). Figure 10.6 shows the mean of solutions represented by S1 and S2 and the associated BB. For the S1, the uncertainty in total time violation is bounded by $[S1_x - \epsilon_1, S1_x + \epsilon_1]$, given a confidence interval of 95%. Similarly, the uncertainty of

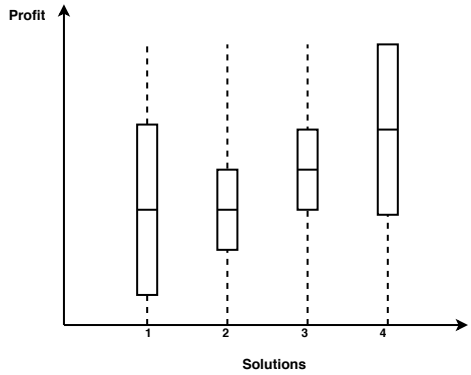


Figure 10.5: 99 % confidence interval of four different solutions

profit is bounded by $[S1_y - \epsilon_2, S1_y + \epsilon_2]$. Note that neither S1 nor S2 is dominated since their bounding boxes are overlapping.

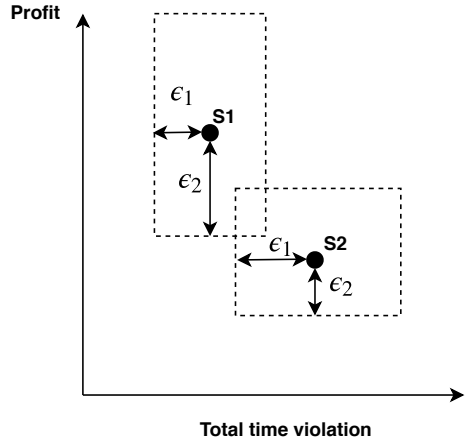


Figure 10.6: Two-dimensional confidence interval represented by a bounding box

The algorithm below describes how the Pareto optimal set is decided when combining the concept of bounding box (or confidence intervals) and the Pareto dominance relation. The input of the algorithm is a set of solutions $\{x_1, x_2, \dots\}$ with the associated total time violation $\{f_1(x_1), f_1(x_2), \dots\}$, profit $\{f_2(x_1), f_2(x_2), \dots\}$ and the confidence vector $\{(\epsilon_1 1, \epsilon_1 2), (\epsilon_2 1, \epsilon_2 2), \dots\}$. The output is the Pareto op-

timal set $\wp^*$ and the set of dominated solutions S. The algorithm loops over each solution and checks if it is dominated by an other solution. If yes, the solution is added to S, otherwise, the solution is added to $\wp^*$.

Algorithm 11 Deciding the Pareto Optimal set

Input: $X = \{x_1, x_2, \dots\}, \{f_1(x_1), f_1(x_2), \dots\}, \{f_2(x_1), f_2(x_2), \dots\}, \{(\epsilon_{11}, \epsilon_{12}), (\epsilon_{21}, \epsilon_{22}), \dots\}$
Output: $\wp^*, S$

```
℘* ← {}
S ← {}
for x_i ∈ X do
    is_dominated ← 0
    for x_j ∈ X | i ≠ j do
        if (f1(x_j) + ε_j1 < f1(x_i) − ε_i1) & (f2(x_j) − ε_j2 > f2(x_i) + ε_i2) then
            is_dominated = 1
            Add x_i to S
            Break inner loop
        else
            Do nothing
        end if
    end for
    if is_dominated = 0 then
        Add x_i to ℘*
    end if
end for
return ℘*, S
```

10.6.4 Robustness Strategies

Three different robustness strategies for the problem studied in this thesis are evaluated by the simulation framework proposed in 8. The three strategies are described in detail in Section 7.3. In addition, each strategy is evaluated with three different parameter values and benchmarked against a base case where no strategies are used. The evaluation is done based on the strategy's ability to generate robust and high-quality solutions in terms of minimum total time violation and maximum profit.

The parameters associated with each strategy are described in Table 10.25. The parameter values used in the testing procedure are presented in Table 10.26.

Table 10.25: Parameters of robustness strategies

Robustness Strategy	Parameter	Description
Penalizing late arrivals	α	Penalizing arrivals occurring after T^{MIN}
Increasing buffer quantity	β	Percentage increase in buffer quantities
Increasing sailing time	ω	Percentage increase in sailing times

Table 10.26: Parameter values used in testing robustness strategies

Parameter	Values	Denoted as
α	{0, 1000, 10000, 100000}	$\{\alpha_0, \alpha_1, \alpha_2, \alpha_3\}$
β	{ 0%, 10%, 20%, 30%}	$\{\beta_0, \beta_1, \beta_2, \beta_3\}$
ω	{0%, 3%, 5%, 7%}	$\{\omega_0, \omega_1, \omega_2, \omega_3\}$

10.6.5 Penalizing Late Arrivals

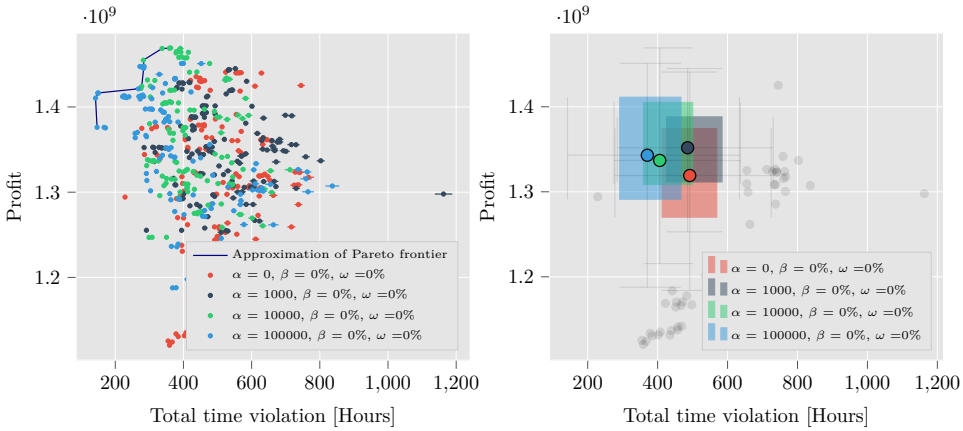
Figure 10.7a shows all the solutions generated with the various values of α. Figure 10.7b shows two-dimensional box plots of the solutions obtained by the different parameter values. Each box represents the interquartile range (IQR) indicating the range of the middle 50% of simulated profits and total time violations. The interquartile range is represented by the length and width of the box. The whiskers represent solutions with profit and total time violation outside the middle 50%. The whiskers and outliers in the figures below are toned down to improve readability. Note that we could only show the range for profit and total time violation of solutions, however, range as a statistical measure is too easily influenced by extreme values which may render any conclusion to be baseless.

In general, the results resemble what is excepted when a penalty is added to the objective function. From Figure 10.7b, we observe that the obtained solutions depend on the α value used such that the greater α, the less total time violation. It is worth mentioning that the simulated profit of each solution does not include the penalty term.

The median of the solutions obtained with α_2 and α_3 in terms of total time violations is clearly less than the median of total time violation associated with solutions obtained by lower values of α (1000 and 0). Both the interquartile ranges and the whiskers follow the same pattern. When it comes to profit, no pattern is observed. However, the profit median of solutions obtained with α_2 and α_3 are surprisingly greater than the median of the base case which indicates that higher values of α may find robust solutions with a lower degree of time violation while performing well in term of profit. This observation can also be confirmed by studying the approximation of the Pareto frontier defined by 16 solutions. All these solutions are exclusively found by α_2 and α_3. Note that some solutions on the approximation of the Pareto frontier may seem dominated at first glance. This is true if a deterministic Pareto relation is used, however since the two-dimensional confidence interval is considered, many of these solutions cannot be declared as dominated.

Comparing α_2 and α_3, we observe that the area of interquartile range and the whiskers associated with α_3 are greater than the whiskers and the interquartile range associated with α_2. This indicates that the solutions found by $alpha_3$ are more diversified and have higher variance. Additionally, we observe that the whiskers and the blue area are evenly distributed around the center (median of profit and time violation of solutions for α_3), while the solutions obtained with α_2 are skewed toward lower profit and lower total time violation since the median (green point) lies in the left lower quadrant in the green box. This observation manifests itself through the greater share of solutions obtained by α_3 in the optimal

Pareto set than solutions obtained by α_2



(a) Solutions found for different α (b) Box plot of solutions for different α

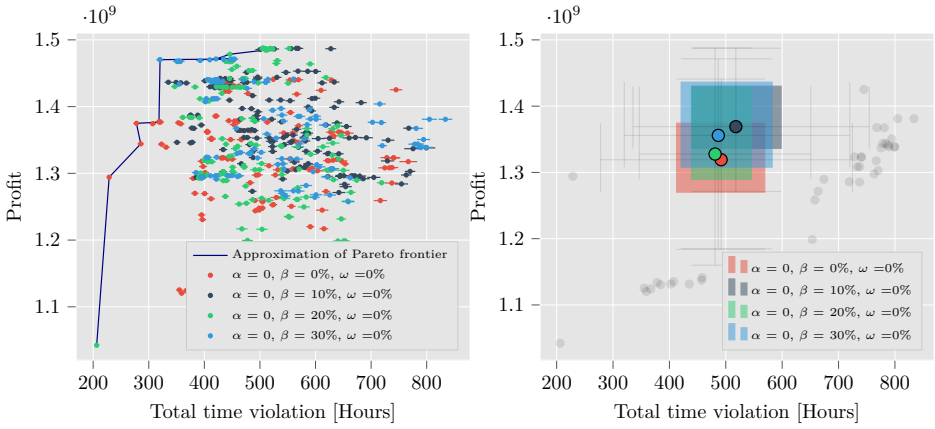
Figure 10.7: Profit and total time violation when penalizing late arrivals

10.6.6 Increasing Buffer Quantity to Avoid Cool-Down

Similar to penalizing late arrivals, the same test procedure is carried out to evaluate the strategy of increasing buffer quantity. The main hypothesis behind implementing this strategy is that increasing buffer quantity may lead to fewer emergency cool-downs. Additionally, the buffer quantity is often too small compared to the vessel's capacity (usually around 3% of a vessel's capacity) which might have a little impact on the profit.

All solutions with their own profit and total violation are shown in Figure 10.8a. The solutions are discriminated based on the buffer quantity increment used when solutions are generated. Comparing the solutions obtained by base case and the other buffer quantity increments, we observe that increasing buffer quantity with 10%, 20% or 30% has a negligible impact on the solutions found as no pattern stands out. Likewise, the optimal Pareto set consists of 20 solutions originating from the four different parameter values. These solutions are almost equally distributed among the four parameter values. This resembles the observation made previously; the impact of this strategy is limited at least for the values used in this study. However, the strategy may have an impact if higher increments are used.

Similarly to Figure 10.7a, some solutions on the approximation Pareto frontier may seem dominated by other solutions at first glance, however, these solutions are not dominated when the two-dimensional confidence interval of the solutions are considered.



(a) Solutions found for different buffer quantity increments

(b) Box plot of solutions for different buffer quantity increments

Figure 10.8: Profit and total time violation when increasing buffer quantity

10.6.7 Increasing Sailing Time

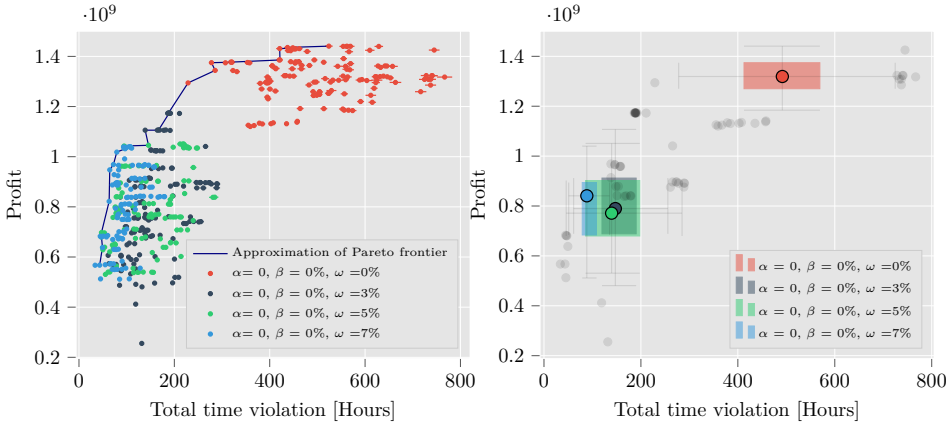
Figure 10.9a shows all the solutions found when the strategy of increasing time is applied and benchmarked against a base case. Figure 10.9b shows the distributional characteristics of each group of solutions. Our first notice is that increasing sailing time has a significant impact on the solutions generated compared to the base case. As expected and discussed in Section 7.3, the solutions found with $\omega > 0$ generate in general less profit and less total time violation. However, it is deserving notice that an increase in sailing time with as little as 3% also has a high impact on the produced solutions. A comparison of the solution with the highest profit associated with ω_1 (SOL1) and the solution with the lowest total time violation associated with ω_0 (SOL0) may illustrate how different these solutions are. SOL1 has 20% less time violation than SOL0 while SOL0 has 10% more profit than SOL1. The high level of difference between solutions generated with base case and $\omega > 0$ indicates that many legs between nodes do not tolerate as little as 3% increase in sailing time, even though a recovery measure like speeding up is carried out. This may be attributed to a combination of narrow time windows and a near to or fully utilized fleet in the solutions associated with ω_0.

A second notice is the difference in variability in profit and total time violation between solutions associated with ω_0 and $\omega > 0$. While the first group of solutions shows high variability in time violation and low variability in profit, the second group exhibits a different pattern with relatively low variability in time violation and high variability in profit. This indicates that adding extra sailing time does not only eliminate risky legs, but it contributes to eliminating combinations of legs that cause extreme values in time violation.

Comparing the solutions associated with ω_1, ω_2, ω_3, we observe that ω_1 and ω_2 produce a set of solutions with almost the same median, both in profit and time violation. While ω_3 stands out with less time violation and higher profit than ω_1 and ω_2. This is a surprising result since one may expect that increasing sailing time may render many high-profit solutions (as seen by the base case) infeasible. Although this behavior may seem strange and unexpected at first glance, the reasoning behind it is related to BOG and emergency cool-downs. On the one hand, increasing sailing time with 3% or 7% appears to eliminate the riskiest legs. However, many legs are still sailed with BOG between delivery ports and pickup ports. This makes the solutions vulnerable for emergency cool-downs.

Additionally, when an emergency cool-down is implemented, the vessel is often forced to empty the tanks at the last delivery port visited before cooling down. This introduces randomness in gained income. On the other hand, increasing sailing time with 7%, few legs can now be sailed with design speed and with BOG.

Hence, vessels are often totally emptied at delivery ports, and cool-downs are scheduled. This makes solutions with ω_3 less vulnerable to disruption in sailing time and emergency cool-downs.



(a) Solutions found for different sailing time increments

(b) Box plot of solutions for different sailing time increments

Figure 10.9: Profit and total time violation when increasing sailing time

10.6.8 Approximation of Pareto Frontier

In this part of the computational study, we study the ability of robustness strategies to guide the solution method, FO in this case, toward more robust solutions and compares these solutions with the case base. This test is carried out by running FO with each robustness strategy and proposed parameter value five times, in total 45 times. The set of solutions obtained during this stage is denoted $S1$. As a benchmark, we generate solutions with no robustness. To give the solution method a fair chance to find high-quality solutions, FO is then run 45 times with the base case. The solutions obtained based on the base case are denoted $S2$. In addition to a box plot for each set's profit and total time violation, Figure 10.10 shows the approximation of the Pareto frontier based on $S1$ and $S2$.

Comparing the box plots, we observe that the median of profit and time violation of set $S1$ is less than these of $S2$. However, $S1$ has, in general, a higher level of diversification and variance than the set $S2$. This applies both in terms of profit and time violation and can be seen both from the interquartile range of $S1$

represented by the red box and the upper and lower quartiles represented by the whiskers as they cover larger area than these associated with $S2$. We observe also that the IQR of $S2$ represented by the blue box is near-to-symmetric around its center, while the IQR, upper and lower quartiles of $S1$ are asymmetric and show that the profit and time violation of $S1$ are skewed left. This indicates that the solutions in the northeast quadrant of the red box have close but height profit and time violation together compared to solutions in the other quadrants. Given the asymmetry and the high variance of $S1$, we may infer that the robustness strategies are likely to diversify the search and guide FO toward extreme solutions both in a positive and negative manner.

In order to assess if some of the solutions in $S1$ pulls in a positive direction, i.e., higher profit and lower total time violation, the Pareto frontier for all solutions obtained in the 90 runs, denoted as $\wp^*$ is approximated and compared with the one obtained by $S1$ and $S2$. It turns out that $\wp^*$ is identical to the approximation of Pareto frontier associated with S_1. For this reason and to increase the readability, $\wp^*$ is not shown in 10.10. In total, there are 57 solutions that define $\wp^*$. Only 7 of them originate from the runs done with the base case. This demonstrates that robustness strategies have the potential to guide the search toward more robust and reliable solutions.

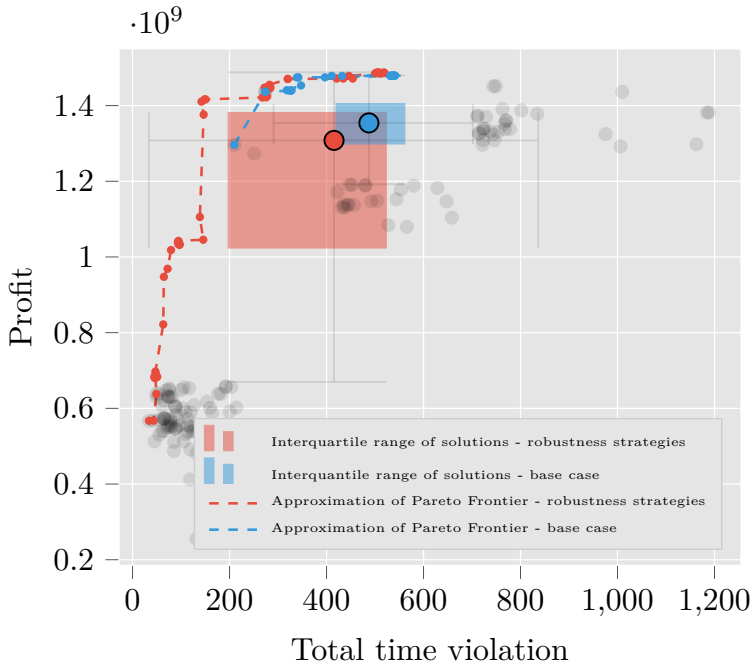


Figure 10.10: An approximation of Pareto frontier based on base case and robustness strategies

11. Concluding Remarks

In this Chapter, we conclude this master's thesis and present future research opportunities. Section 11.1 presents the conclusion, while Section 11.2 discusses future research.

11.1 Concluding Remarks

This thesis presents two solution methods for the short-term ship routing and scheduling problem for an LNG producer with a heterogeneous fleet of vessels. The proposed solution methods are an adaptive large neighborhood search (ALNS) and large neighborhood fix and optimize (FO). The objective when solving this problem is to maximize the producer's profit while satisfying the customer's time and quantity requirements. The problem is solved by deciding which vessel servicing which customers, in what sequence, when they are serviced and how much to load and unload in each port. The input of this problem is based on an annual delivery program that is inefficient in the face of disruptions and emergence of opportunities in the spot market.

Realistic test instances of the problem considered in this thesis are not possible to solve to optimality through the use of a general-purpose commercial solver. Reviewing related literature led to the hypothesis that heuristic approaches that incorporate a MILP as a part of the search procedure performs well in solving problems with combinatorial complexity and dependency between a large number of different decisions. For this reason, the FO and ALNS were developed.

The adaptive large neighborhood search heuristic presented in the thesis is a partial optimization method that uses an ALNS framework in the first part and solves a MIP in the second part with the routes from a solution in the first part fixed. Decisions that are difficult to handle in a neighborhood search based heuristic are postponed to the second part of the heuristic. This was shown to introduce a bias in the heuristic, in the sense that solutions that were the best in the first part of

the heuristic were not necessarily the best in the second part. This bias seems to have a significant impact on the performance of the ALNS heuristic. Even though the ALNS tends to use significantly less time to reach decent solutions than a the solved MIP model for large instances, the final solutions found by the ALNS when run for 10 000 iterations and time limit of 3600 s were not able to consistently outperform the solution found by running the MIP model for 3600 s. The ALNS was tested using both the original ADP plan and a construction heuristic as initial solution. Using the original plan yielded slightly better solutions on large instances and marginally more consistent performance. Still, the added value of starting from the ADP plan was not remarkable.

The fix and optimize heuristic searches for good solutions using a MIP-solver iteratively and fixing a subset of the variables. The FO outperforms both the MIP solvers tested and the ALNS. Within 3600 seconds the FO is much better than the alternatives. The FO is often although not always able to find good solutions in a relatively short period of time (< 500 seconds), especially when started from an initial solution from an ADP plan. Unfortunately, even when starting from an initial solution corresponding to an ADP the heuristic does not provide good solutions in a very short time (< 60 seconds). This is especially true when the problem instance is highly disrupted, as can be seen from Figure 10.2. The FO was found to outperform the ALNS in terms of both solutions quality and time to find good solutions.

Additionally, a simulation model has been developed to assess the robustness of solutions obtained. The simulation model serves the purpose of evaluating the solutions in an environment that mimics a real-world setting. The evaluation is done with respect to robustness (e.g. ability to handle realistic, problem-specific, disruptions such as delays and production volume fluctuations). Approximations of Pareto frontiers are presented, and three robustness strategies have been developed and tested. A benchmark between solutions found with robustness strategies and solutions found without applying robustness strategies show the strategies' ability to guide a solution method like FO toward more robust solutions.

11.2 Future Research Opportunities

In this section we present potential areas of future work that may improve the models and solution approaches presented in this thisis.

Several improvements to the ALNS heuristic were identified during the development and testing of the heuristic. More destory and repair operators, as well as local search operators can be developed, and the local search can also be made

adaptive. New partitionings of the problem could be developed, where more decisions are included in the first part of the heuristic. Different acceptance criteria could be implemented, or the simulated annealing approach used in the thesis could be analyzed further, for instance by adding Re-heat strategies or different temperature schedules.

For the Fix and Optimize heuristic, we did not find it necessary to accept non-improving solutions, but it might be good to do further research on whether this should be done and also how it should be done, as there might be certain types of non-improving solutions which are interesting and certain types which are not; however, it seems most relevant to do this for other repair alternatives than the MIP.
Tuning the Fix and Optimize LNS is as discussed earlier a complex task, and further tuning and testing of the model might be worth the effort.
Heuristic methods are often iterative, not only by their nature, like for the FO, but perhaps also through the use of multi-start methods and parallel solving. Thus, a heuristic does not necessarily need to consistently outperform other alternatives for every single model run, but for the Fix and Optimize heuristic presented in this thesis, the results on these particular data sets which we have tested on are promising. Still, options for improving the Fix and Optimize LNS include parallelisation and multi-start solving. These approaches should probably be combined with the development of additional methods for creating initial paths and a tabu list, so that it is possible to avoid repeatedly search of the same local solution spaces. However, to the best of our knowledge from testing on our data, multiple starts from the same initial solution may lead to different final solutions. So far we have only one repair method and one destroy method (although having some dynamically adjusted parameters). Therefore, it might be good to develop additional methods for repair/destroy for the Fix and Optimize LNS.
Future work may also include work on improving the co-contributions from the Fix and Optimize LNS and the simulations. As an example, the Fix and Optimize LNS might find the 1st, 3rd, 5th and 10th deterministic best integer solutions; however, the 2nd best solution (which has not been found) might prove itself as an overall superior solution after introducing realistic disruption and simulating the solutions and their performance. This concept of fetching additional solution candidates from the Fix and Optimize LNS (maybe especially those being close to the optimal objective value) has been addressed implicitly in section 7.3; however, this feature can be developed further.

For the model presented in this thesis, the scope have been confined an it is not found necessary to include the modeling and testing of accepting non-improving solutions between iterations. However, it might be good to perform future research

on whether this should be done and how it should be done, as there might be certain types of non-improving solutions which are interesting and certain types which are not. It is possible, that if the model is combined with multi-start solution program or if additional repair methods are introduced then this becomes more relevant.

The simulation framework could be improved upon by including more robustness strategies, or combining several of the existing strategies. In addition, one may combine the simulation framework into the solutions methods to solve the problem by a simulation-optimization approach.

Bibliography

Agra, Agostinho, Marielle Christiansen, Alexandrino Delgado, and Luidi Simonetti (2014). "Hybrid heuristics for a short sea inventory routing problem". In: *European Journal of Operational Research* 236.3, pp. 924–935.

Andersson, Henrik, Marielle Christiansen, and Guy Desaulniers (2016). "A new decomposition algorithm for a liquefied natural gas inventory routing problem". In: *International Journal of Production Research* 54.2, pp. 564–578.

Andersson, Henrik, Marielle Christiansen, Guy Desaulniers, and Jørgen Glomvik Rakke (2017). "Creating annual delivery programs of liquefied natural gas". In: *Optimization and Engineering* 18.1, pp. 299–316.

Andersson, Henrik, Marielle Christiansen, and Kjetil Fagerholt (2010). "Transportation planning and inventory management in the LNG supply chain". In: *Energy, natural resources and environmental economics*. Springer, pp. 427–439.

Archetti, Claudia and M Grazia Speranza (2014). "A survey on matheuristics for routing problems". In: *EURO Journal on Computational Optimization* 2.4, pp. 223–246.

Bausch, Dan O, Gerald G Brown, and David Ronen (1998). "Scheduling short-term marine transport of bulk products". In: *Maritime Policy & Management* 25.4, pp. 335–348.

Berle, Øyvind, Inge Norstad, and Bjorn E Asbjørnslett (2013). "Optimization, risk assessment and resilience in LNG transportation systems". In: *Supply Chain Management: An International Journal* 18.3, pp. 253–264.

Boschetti, Marco A, Vittorio Maniezzo, Matteo Roffilli, and Antonio Bolufé Röhler (2009). "Matheuristics: Optimization, simulation and control". In: *International Workshop on Hybrid Metaheuristics*. Springer, pp. 171–177.

Bramel, Julien and David Simchi-Levi (1995). "A location based heuristic for general routing problems". In: *Operations research* 43.4, pp. 649–660.

Brønmo, Geir, M Christiansen, and Bjørn Nygreen (2007). "Ship routing and scheduling with flexible cargo sizes". In: *Journal of the Operational Research Society* 58.9, pp. 1167–1177.

Brønmo, Geir, Bjørn Nygreen, and Jens Lysgaard (2010). "Column generation approaches to ship scheduling with flexible cargo sizes". In: *European Journal of Operational Research* 200.1, pp. 139–150.

Brown, Gerald G, Glenn W Graves, and David Ronen (1987). "Scheduling ocean transportation of crude oil". In: *Management Science* 33.3, pp. 335–346.

Campbell, Ann Melissa and Martin WP Savelsbergh (2004). "A decomposition approach for the inventory-routing problem". In: *Transportation science* 38.4, pp. 488–502.

Chandra, Vivek (2017). *Fundamentals of natural gas: an international perspective*. PennWell Corporation.

Coelho, Leandro C, Jean-François Cordeau, and Gilbert Laporte (2012). "The inventory-routing problem with transshipment". In: *Computers & Operations Research* 39.11, pp. 2537–2548.

Demir, Emrah, Tolga Bektaş, and Gilbert Laporte (2012). "An adaptive large neighborhood search heuristic for the pollution-routing problem". In: *European Journal of Operational Research* 223.2, pp. 346–359.

DESFA (n.d.). *List of additional LNG services*. URL: `http://www.desfa.gr/userfiles/pdf/List-of-LNG-Additional-Services2.pdf`.

Dobrota, ore, Branko Lalić, and Ivan Komar (2013). "Problem of boil-off in LNG supply chain". In: *Transactions on maritime science* 2.02, pp. 91–100.

Drawn-out ball game: Asian spot LNG prices to stay below long-term (n.d.). URL: `http://www.gasstrategies.com/blogs/drawn-out-ball-game-asian-spot-lng-prices-stay-below-long-term-contract-prices`.

Exxon Mobil Corporation (2018). "2018 Outlook for Energy: A view to 2040". In: URL: `https://cdn.exxonmobil.com/~/media/global/files/outlook-for-energy/2018/2018-outlook-for-energy.pdf`.

Fagerholt, Kjetil (2001). "Ship scheduling with soft time windows: An optimisation based approach". In: *European Journal of Operational Research* 131.3, pp. 559–571.

Fischer, Andreas, Håkon Nokhart, Henrik Olsen, Kjetil Fagerholt, Jørgen Glomvik Rakke, and Magnus Stålhane (2016). "Robust planning and disruption management in roll-on roll-off liner shipping". In: *Transportation Research Part E: Logistics and Transportation Review* 91, pp. 51–67.

Fisher, Marshall L and Ramchandran Jaikumar (1981). "A generalized assignment heuristic for vehicle routing". In: *Networks* 11.2, pp. 109–124.

Fodstad, Marte, Kristin Tolstad Uggen, Frode Rømo, Arnt-Gunnar Lium, Geert Stremersch, and Stéphane Hecq (2010). "LNGScheduler: a rich model for coordinating vessel routing, inventories and trade in the liquefied natural gas supply chain". In: *The journal of energy markets* 3.4, pp. 31–64.

GIIGNL (2018). *The LNG industry GIIGNL ANNUAL REPORT 2018*. URL: https://giignl.org/sites/default/files/PUBLIC_AREA/About_LNG/5_LNG_Markets_And_Trade/giignl_2018_annual_report.pdf.

Glover, Fred and Jin-Kao Hao (2011). "The case for strategic oscillation". eng. In: *Annals of Operations Research* 183.1, pp. 163–173. ISSN: 0254-5330.

Goel, Vikas, Kevin C Furman, Jin-Hwa Song, and Amr S El-Bakry (2012). "Large neighborhood search for LNG inventory routing". In: *Journal of Heuristics* 18.6, pp. 821–848.

Grasmair, Markus (2018). "BASIC PROPERTIES OF CONVEX FUNCTIONS". In: URL: https://wiki.math.ntnu.no/_media/tma4180/2016v/note2.pdf (visited on 12/21/2018).

Grønhaug, Roar and Marielle Christiansen (2009). "Supply chain optimization for the liquefied natural gas business". In: *Innovations in distribution logistics*. Springer, pp. 195–218.

Grønhaug, Roar, Marielle Christiansen, Guy Desaulniers, and Jacques Desrosiers (2010). "A branch-and-price method for a liquefied natural gas inventory routing problem". In: *Transportation Science* 44.3, pp. 400–415.

Al-Haidous, Sara, Mohamed Kais Msakni, and Mohamed Haouari (2016). "Optimal planning of liquefied natural gas deliveries". In: *Transportation Research Part C: Emerging Technologies* 69, pp. 79–90.

Halvorsen-Weare, Elin E and Kjetil Fagerholt (2011). "Robust supply vessel planning". In: *International Conference on Network Optimization*. Springer, pp. 559–573.

— (2013). "Routing and scheduling in a liquefied natural gas shipping problem with inventory and berth constraints". In: *Annals of Operations Research* 203.1, pp. 167–186.

Halvorsen-Weare, Elin E., Kjetil Fagerholt, and Mikael Rönnqvist (2013). "Vessel routing and scheduling under uncertainty in the liquefied natural gas business". eng. In: *Computers Industrial Engineering* 64.1, pp. 290–301. ISSN: 0360-8352.

Hemmati, Ahmad, Lars Magnus Hvattum, Kjetil Fagerholt, and Inge Norstad (2014). "Benchmark suite for industrial and tramp ship routing and scheduling problems". In: *INFOR: Information Systems and Operational Research* 52.1, pp. 28–38.

IGU (2012). *Natural Gas Conversion Pocketbook*. URL: http://http://members.igu.org/old/IGU%5C%20Events/wgc/wgc-2012/wgc-2012-proceedings/publications/igu-publications (visited on 10/24/2018).

— (2018). *2018 World LNG Report*. URL: https://www.igu.org/sites/default/files/node-document-field_file/IGU_LNG_2018_0.pdf.

Iversen, øivin Iversen and Roy-Inge Sørensen Sørensen (2005). *Advances in design and layout of Moss LNG carriers*. URL: http://www.ivt.ntnu.no/ept/

fag/tep4215/innhold/LNG%5C%20Conferences/2005/SDS_TIF/050208.pdf (visited on 11/14/2018).

Jaimes, Antonio López, Saúl Zapotecas Martınez, and Carlos A Coello Coello (2009). "An introduction to multiobjective optimization techniques". In: *Optimization in Polymer Processing*, pp. 29–57.

King, Alan J and Stein W Wallace (2012). *Modeling with stochastic programming*. Springer Science & Business Media.

Korsvik, Jarl Eirik and Kjetil Fagerholt (2010). "A tabu search heuristic for ship routing and scheduling with flexible cargo quantities". In: *Journal of Heuristics* 16.2, pp. 117–137.

Korsvik, Jarl Eirik, Kjetil Fagerholt, and Gilbert Laporte (2011). "A large neighbourhood search heuristic for ship routing and scheduling with split loads". In: *Computers & Operations Research* 38.2, pp. 474–483.

Koza, David Franz, Stefan Ropke, and Anna Boleda Molas (2017). "The liquefied natural gas infrastructure and tanker fleet sizing problem". In: *Transportation Research Part E: Logistics and Transportation Review* 99, pp. 96–114.

Lindahl, Michael, Matias Sørensen, and Thomas R. Stidsen (2018). "A fix-and-optimize matheuristic for university timetabling". In: *Journal of Heuristics* 24.4, pp. 645–665. DOI: 10.1007/s10732-018-9371-3.

Liquefied Gas Carrier (n.d.). *Gassing-up tanks procedure for loading LNG cargo on board*. URL: http://www.liquefiedgascarrier.com/gassing-up-tanks.html.

Liquefied natural gas: understanding the basic facts (2005). U.S. Department of Energy, Office of Fossil Energy.

Mlakar, Miha, Tea Tušar, and Bogdan Filipič (2014). "Comparing solutions under uncertainty in multiobjective optimization". In: *Mathematical Problems in Engineering* 2014.

Moon, Kiho, Daejun Chang, Donghun Lee, Myung-Bae Kim, Ho-Jong Ahn, and Jong-Pil Ha (2005). "Comparison of spherical and membrane large lng carriers in terms of cargo handling". In: *Hyundai Heavy Industries, Co., Ltd*, pp. 1–11.

Msakni, Mohamed Kais and Mohamed Haouari (2018). "Short-term planning of liquefied natural gas deliveries". In: *Transportation Research Part C: Emerging Technologies* 90, pp. 393–410.

Mutlu, Fatih, Mohamed K Msakni, Hakan Yildiz, Erkut Sönmez, and Shaligram Pokharel (2016). "A comprehensive annual delivery program for upstream liquefied natural gas supply chain". In: *European Journal of Operational Research* 250.1, pp. 120–130.

National Energy Technology Laboratory (2013). *Cost and Performance Baseline for Fossil Energy Plants*.

Notteboom, Theo E (2006). "The time factor in liner shipping services". In: *Maritime Economics & Logistics* 8.1, pp. 19–39.

Pedersen, Chris (2017). *LNG Prices Pricing Mechanisms*. URL: https://www.platts.com/IM.Platts.Content/ProductsServices/ConferenceandEvents/americas/liquefied-natural-gas/presentations2017/Chris_Pederson.pdf (visited on 10/30/2018).

Psaraftis, Harilaos N and Christos A Kontovas (2013). "Speed models for energy-efficient maritime transportation: A taxonomy and survey". In: *Transportation Research Part C: Emerging Technologies* 26, pp. 331–351.

Rakke, Jørgen G (2012). "Optimization Models and Methods for Maritime Routing and Scheduling Problems". In:

Rakke, Jørgen Glomvik, Henrik Andersson, Marielle Christiansen, and Guy Desaulniers (2014). "A new formulation based on customer delivery patterns for a maritime inventory routing problem". In: *Transportation Science* 49.2, pp. 384–401.

Rakke, Jørgen Glomvik, Magnus Stålhane, Christian Rørholt Moe, Marielle Christiansen, Henrik Andersson, Kjetil Fagerholt, and Inge Norstad (2011). "A rolling horizon heuristic for creating a liquefied natural gas annual delivery program". In: *Transportation Research Part C: Emerging Technologies* 19.5, pp. 896–911.

Rodrıguez-Martın, I and Juan José Salazar-González (2011). "The multi-commodity one-to-one pickup-and-delivery traveling salesman problem: a matheuristic". In: *International Conference on Network Optimization*. Springer, pp. 401–405.

Rodrıguez-Martın, Inmaculada and Juan José Salazar-González (2012). "A hybrid heuristic approach for the multi-commodity one-to-one pickup-and-delivery traveling salesman problem". In: *Journal of Heuristics* 18.6, pp. 849–867.

Ropke, Stefan and David Pisinger (2006). "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows". In: *Transportation science* 40.4, pp. 455–472.

Sea routes and distances (n.d.). URL: http://ports.com/sea-route/.

Shaw, Paul (1997). "A new local search algorithm providing high quality solutions to vehicle routing problems". In: *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*.

Ship & Bunker (2018). *Rotterdam Bunker Prices*. URL: https://shipandbunker.com/prices/emea/nwe/nl-rtm-rotterdam (visited on 10/15/2018).

Spot LNG Price Statistics (2018). URL: http://www.meti.go.jp/english/statistics/sho/slng/index.html.

Stålhane, Magnus, Jørgen Glomvik Rakke, Christian Rørholt Moe, Henrik Andersson, Marielle Christiansen, and Kjetil Fagerholt (2012). "A construction and

improvement heuristic for a liquefied natural gas inventory routing problem". In: *Computers & Industrial Engineering* 62.1, pp. 245–255.

Stopford, Martin (2013). *Maritime economics*. Routledge.

Toth, Paolo and Daniele Vigo (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

Tusiani, Michael D. and Gordon Shearer (2007). *LNG: A Nontechnical Guide*. Tulsa, Okla: PennWell Books. ISBN: 9780878148851.

Uggen, Kristin Tolstad, Marte Fodstad, and Vibeke Stærkebye Nørstebø (2013). "Using and extending fix-and-relax to solve maritime inventory routing problems". In: *Top* 21.2, pp. 355–377.

Wen, Min, Stefan Ropke, Hanne L Petersen, Rune Larsen, and Oli BG Madsen (2016). "Full-shipload tramp ship routing and scheduling with variable speeds". In: *Computers & Operations Research* 70, pp. 1–8.

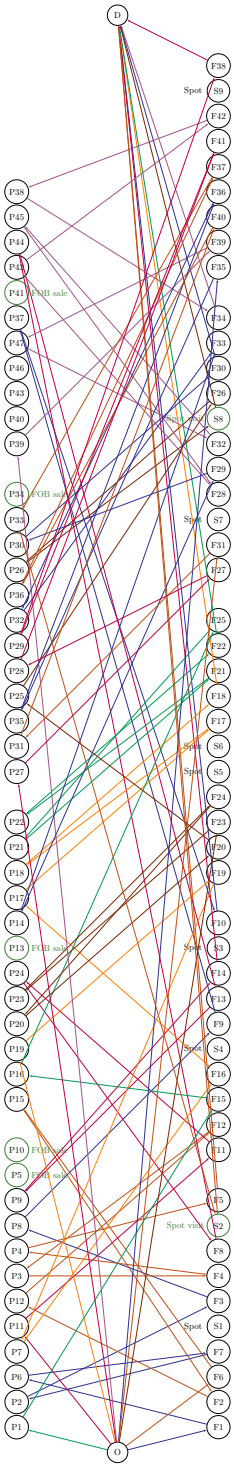
World Energy Outlook (2018). eng. OECD Publishing.

Yan, Shangyao, Rong-Chang Jou, Chia-Hung Chen, and Chyi-Feng Lee (2005). "Neighborhood search algorithms with restricted infeasible solution sets–application to a transportation project evaluation problem". In: *Journal of the Chinese Institute of Engineers* 28.3, pp. 545–550. ISSN: 0253-3839.

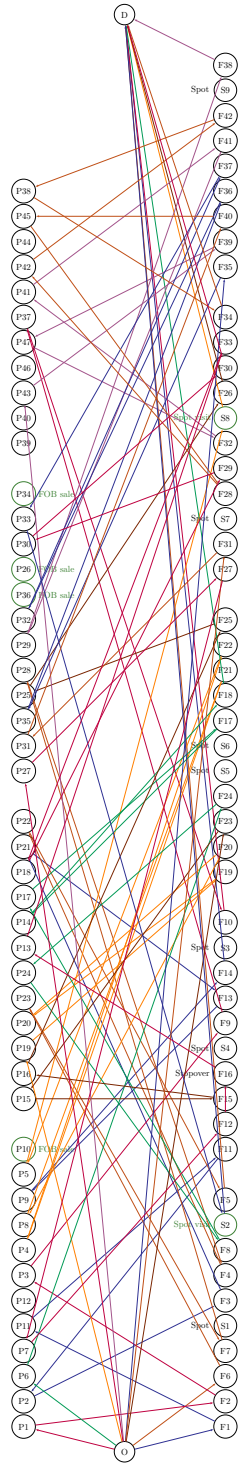
Yıldırım, Umman Mahir and Bülent Çatay (2014). "A parallel matheuristic for solving the vehicle routing problems". In: *Computer-based Modelling and Optimization in Transportation*. Springer, pp. 477–489.

A. Appendices

A.1 Figures



(a) Fix and optimize, ADP start, benchmark



(b) Fix and optimize, CH start, 0.7% away from the solution found with ADP start

Figure A.1: Example of two competing solutions being close to each other in objective value but far away from each other in the solution space

A.2 Introducing Variable Speed to the Mathematical Model

So far we have assumed that each vessel's speed is an implicit input to the model since it is only used to compute explicit inputs like voyage costs (VOYEX) and sailing time between ports. Although fixed speed is traditionally used in long term planning by shipping companies and LNG operators, each vessel has in reality a range of speeds which it can sail at. An optimal solution of the model for a given speed is only optimal for that speed and might be infeasible or a bad solution for other speeds. This is especially important in short term LNG planning as time windows tend to be very narrow which makes an optimal solution very sensitive to small changes in speed-related inputs like sailing time. This property is often called a knife edge to describe how an optimal solution of a deterministic problem perfectly fits a set of given inputs but might fail to generalize to small changes in these inputs (King and Wallace, 2012). Additionally, modeling speed as an implicit input might render the solution sub-optimal as the degree of flexibility in the overall decision making process is reduced (Psaraftis and Kontovas, 2013).

As mentioned in Chapter 2, changes in speed have a considerable impact on the voyage costs. Furthermore, a solution where a vessel v with a prescribed speed is scheduled to arrive at a certain port before the opening hours of that port is a more expensive solution than allowing the vessel to lower its speed and arrive after the lower limit of the port's time window. One way to deal with this problem is to perform sensitivity analysis on speed for each leg and each vessel. However, this approach can be cumbersome and computationally heavy since the model has to be solved for each single change in speed for each vessel. Given these points, speed is henceforth treated as a decision variable.

A.2.1 Modelling Fuel Consumption Functions

To include speed as a decision variable it is of vital importance to carefully estimate fuel consumption as it is an important item in the voyage costs of a vessel. Considering a heterogeneous fleet of vessels adds some complexity to the problem since each vessel might have a unique and different engine configuration and boil-off handling technology (e.g., re-liquefaction). Consequently, each vessel might have a different relationship between speed and fuel consumption. For instance, a vessel with re-liquefaction technology transforms BOG to a liquefied form, hence, 100 % of the picked up cargo is delivered to the customer. Such a vessel might have one or several diesel engines running on Heavy Fuel Oil or the more environmentally friendly type Intermediate Fuel Oil (IFO 380). For a vessel that can sail at speeds

between 15 and 22 Knots, a typical curve of the fuel consumption per time unit as function of speed can be shown in fig:speed curve1. Note that the curve is convex and incorporates a cost to be minimized, a Special Ordered Sets of type 2 (SOS2) can be used to linearize the curve as shown by the red lines.

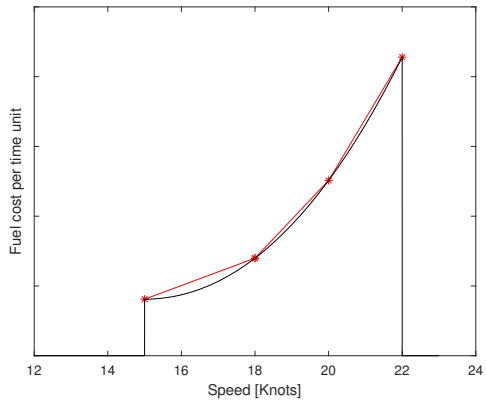


Figure A.2: Fuel Consumption ton/time unit as a function of speed for a LNG vessel with re-liquefaction technology

Differently, a vessel without re-liquefaction technology and a traditional steam engine uses a blend of fuel oil and BOG. Assuming that the produced BOG is enough to run the vessel at cruising speed, the fuel cost can be assumed equal to zero at all speeds below the vessel's cruising speed. This is reasonable since BOG is inevitable in this case. However, sailing with speeds above the assumed cruising speed requires additional fuel oil at a cost similar to the vessel with re-liquefaction technology. fig:speed curve2 shows how such a curve looks like.

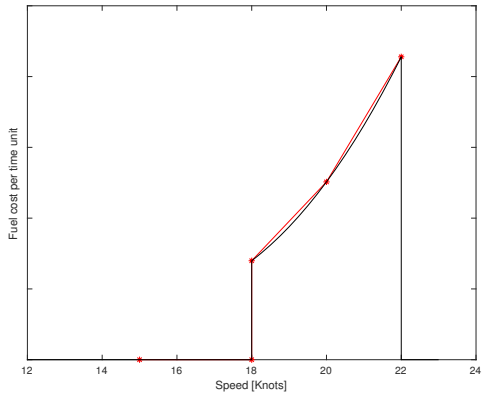


Figure A.3: Fuel Consumption ton/time unit as a function of speed for a LNG vessel with steam engine

Note that this curve is not convex. Although this can be proven mathematically, we will confine ourselves to show this fact graphically. Recall that a function is convex if all points on or above the function's graph is a convex set. This means that the connecting line segment between any two points on the function's graph remains above or on the graph of the function (Grasmair, 2018). Denoting speed for s and the fuel cost per time unit at speed s for $f(s)$, it is easy to show, for instance, the line connecting $(18, f(18))$ and $(20, f(20)$ lies below the graph, hence, $f(s)$ for steam engines is non-convex. This is illustrated by the dashed line in fig:speed curve non convex 1.

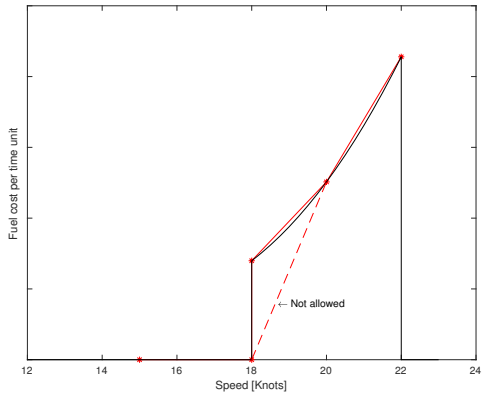


Figure A.4: Non-convex fuel consumption curve

An important consequence of $f(s)$ being discontinuous and non-convex is that using SOS2 to linearize $f(s)$ does not hold anymore. The convex combination model offers an appropriate modelling for our problem. The formulation requires $f(s)$ to be a linear combination of at most two ends of one section/piece on the graph. Hence, in addition to the weight variables used in SOS2, a set of binary variables are needed to require that the weight variables representing the linear combination of two points are actually two ends of one section.