Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Economics and Management
Dept. of Industrial Economics and Technology Management

Karl Petter Ulsrud
Anders Helgeland Vandvik

The Time-Dependent Vessel Routing Problem with Speed Optimization

An Adaptive Large Neighborhood Search

Master's thesis in Industrial Economics and Technology Management
Supervisor: Kjetil Fagerholt
Co-supervisor: Andreas Breivik Ormevik
June 2021

sismarine.com

NTNU
Norwegian University of Science and Technology

Karl Petter Ulsrud
Anders Helgeland Vandvik

The Time-Dependent Vessel Routing Problem with Speed Optimization

An Adaptive Large Neighborhood Search

Master's thesis in Industrial Economics and Technology Management
Supervisor: Kjetil Fagerholt
Co-supervisor: Andreas Breivik Ormevik
June 2021

Norwegian University of Science and Technology
Faculty of Economics and Management
Dept. of Industrial Economics and Technology Management

NTNU
Norwegian University of
Science and Technology

Preface

This master's thesis concludes our Master of Science at the Norwegian University of Science and Technology, Department of Industrial Economics and Technology Management. The work has been conducted during the spring of 2021 and is a continuation of the specialization project in TIØ4500 Managerial Economics and Operations Research in the fall of 2020 (Ulsrud and Vandvik, 2020).

The thesis is written in collaboration with Equinor and SINTEF's LowEmission Research Centre and aims to provide new insights into the operational offshore routing of supply vessels. The work is focused on reducing costs and emissions from Equinor's upstream supply chain operations at the Norwegian Continental Shelf, applying appropriate solution methods.

We would like to thank our supervisor Professor Kjetil Fagerholt for his enthusiasm and valuable guidance throughout writing this thesis. We would also like to express our gratitude to Ph.D. candidate Andreas Breivik Ormevik for his input on the domain. Finally, we extend our thanks to SINTEF and Equinor for helpful input and access to relevant data.

Trondheim, June 11, 2021

Karl Petter Ulsrud, Anders Helgeland Vandvik

Abstract

Supply vessel planning in offshore logistics is a complex issue faced by Equinor, the leading operator at the Norwegian Continental Shelf. The planning problem involves routing platform supply vessels (PSVs) from an onshore depot to offshore installations requiring supplies to perform their daily operations. The planning process is currently assisted by a tactical planning tool providing weekly voyages and schedules. On the operational level, adjustments for varying weather conditions and order sizes are done manually. This thesis defines and solves the Time-Dependent Vessel Routing Problem with Speed Optimization (TDVRPSO), aiming to provide operational support to decision-makers.

The TDVRPSO concerns routing PSVs from an onshore depot on a given day to offshore installations, servicing delivery and pickup orders in a manner that minimizes the costs related to contracted PSVs in the fleet and external spot PSVs that can be chartered for a specific voyage. The inputs to the TDVRPSO include contracted and available spot PSVs and their associated operating costs, a set of installations with orders, and weather forecasts for the considered planning horizon. In order to best adhere to restrictions imposed by weather conditions on sailing speeds and the servicing of orders at installations, time-dependent speed optimization is performed. This allows PSVs to slow down to fuel-efficient speeds or speed up if this is cost-efficient. Furthermore, some of the low-priority orders can be considered as optional to service. Optional orders can be postponed at a penalty cost. Following the problem's objective, the TDVRPSO outputs cost-optimal voyages for the available contracted PSVs and potentially additional spot market PSVs.

An exact mathematical arc-flow model of the TDVRPSO is introduced, but using the model to solve realistic instances of the TDVPRSO is too time-consuming. In response, an Adaptive Large Neighborhood Search (ALNS) heuristic is implemented and extended with a local search and a set partitioning problem for finding improving voyages for the PSVs. Further, the heuristic includes solving a subproblem to find optimal sailing speeds for the given voyages. The ALNS heuristic is tailored to the TDVRPSO, and extensive tests demonstrate that it is able to provide high-quality solutions to all test instances based on a real supply vessel planning case.

For the test instances solved, the results show that the solution methods consider the operational aspects of the problem. Planning for these aspects suggests decreases of up to 24% in CO_2 emissions and up to 40% in operational costs for realistic instances.

Thus, the study adds to the existing literature by considering time-dependent speed optimization while simultaneously allowing for order selection of both delivery and pickup orders. Additionally, it provides an extensive ALNS heuristic that delivers high-quality solutions to problems including these features.

Sammendrag

Planlegging og bruk av forsyningsfartøy innebærer komplekse oppgaver som Equinor, den største operatøren på norsk kontinentalsokkel, står overfor. Oppgavene omhandler ruting av forsyningsfartøy, også kalt *platform supply vessels* (PSV), fra et forsyninsglager på land til offshore olje- og gassplattformer som trenger forsyninger for å kunne opprettholde daglig drift. Per i dag støttes planleggingsprosessen av et taktisk planleggingsverktøy som gir ukentlige ruter og tidsplaner. På operasjonelt nivå gjøres planleggingen manuelt, noe som betyr at planleggerne selv justerer de taktiske planene slik at disse tar hensyn til forskjellige faktorer som værforhold og ordreavvik. Med sikte på å gi operativ beslutningsstøtte definerer og løser vi i denne denne avhandlingen det *tidsavhengige skipsrutingproblemet med hastighetsoptimering* (TDVRPSO).

I TDVRPSO minimeres kostnadene tilknyttet bruk av fartøy i den originale flåten, samt eventuelle kostnader som følger ved leie av forsynigsfartøy. Problemet løses ved å betrakte en mengde med installasjoner med ordre, samt informasjon om værforholdene for den planlagte perioden. For å kunne møte restriksjonene som er satt på seilingshastigheter og betjening av ordre ved installasjoner gjennomføres tidsavhengig hastighetsoptimering. Dette gjør at fartøyene kan seile ved drivstoffeffektive hastigheter der dette er mulig, eller øke hastigheten hvis dette er kostnadseffektivt. Valgbare ordre kan utsettes mot en straffekostnad. Etter å ha løst problemet sitter planleggerne igjen med kostnadsoptimerte ruter for tilgjengelige fartøy i flåten og potensielle innleide fartøy.

I løsingen av problemet introduseres en eksakt matematisk modell. Denne modellen er dog for tidkrevende å løse for realistiske probleminstanser. For å kunne løse disse instansene innen respektabel tid har en Adaptive Large Neigborhood Search (ALNS) heuristikk blitt implmentert. Utvidelser til heuristikken inkluderer et lokalsøk samt et tilleggsproblem for å kombinere ruter som er funnet tidligere i søket. Videre løses et sub-problem for å finne optimale hastigheter tilknyttet de ulike rutene. Heuristikken er spesialtilpasset avhandlingens problem, og omfattende testing viser at den genererer løsninger av høy kvalitet for samtlige testinstanser basert på situasjonen som betraktes i oppgaven. Analysen av løsningene som er funnet viser at løsningsmetodene hensyntar de operasjonelle aspektene i problemet. Den operasjonelle planleggingen indikerer reduksjoner på opptil 24% i CO_2-utslipp og opptil 40% i driftskostnader for realistiske instanser av problemet.

Gjennom å gjøre tidsavhengig hastighetsoptimering samtidig som valgbare ordre og henteordre hensynstas tilfører arbeidet ny innsikt og bidrar til litteraturen på området. I tillegg presenterer det en ALNS-heuristikk som produserer løsninger av høy kvalitet på problemer som innehar disse karakteristikkene.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The petroleum industry is Norway's largest industry in terms of both value creation and government revenue. According to the Norwegian government (2020), petroleum production on the Norwegian Continental Shelf (NCS) has generated present values of more than 15 700 billion NOK since production began in 1971. The NCS has three areas available for petroleum production: the North Sea, the Norwegian Sea, and the Barents Sea. Most current fields are located in the North Sea. The Norwegian government (2020) estimates that out of the 15.6 billion standard cubic meters of petroleum resources in the NCS, 53% is yet to be extracted.

In 1972, the Norwegian State Oil Company, Statoil, was formed. The company, which is now called Equinor, is the leading operator in the Norwegian petroleum industry, currently operating 42 fields on the NCS and maintaining a daily production of 2.5 million barrels of oil. Consequently, Equinor accounts for 70% of Norway's oil and gas production and intends to sustain the current production level until 2030 (Equinor, 2020c).

Equinor's petroleum operations are primarily set to offshore installations at the fields on the NCS. These are generally self-sufficient with energy and water. However, a variety of supplies such as food and chemicals are required to be transported to the installations on a regular basis. These supplies are transported from an onshore depot to the installations using platform supply vessels (PSVs) specifically designed for this purpose. Further, PSVs collect used supplies from the installations. In the upstream supply chain of oil companies, operating and chartering PSVs account for a major portion of the total costs. Maintaining the current production level, increasing efficiency, and reducing costs related to PSVs will thus be of great interest.

Additionally, the environmental impact of the petroleum industry is an issue. On a domestic level, emissions from production and logistics within the petroleum industry amount to 14 million tonnes of CO_2 equivalents, which is about a quarter of Norway's total greenhouse gas emissions. A substantial part of these emissions comes from the fuel consumed

by PSVs supplying offshore installations, and it is currently one of the dominating contributors to Equinor's carbon footprint. Hence, optimizing logistics involving PSVs is essential for their upstream supply chain, both in terms of monetary costs and carbon footprint.

Supply vessel planning may be approached on the strategic, tactical, and operational levels. The current optimization-based tactical planning tool used by Equinor generates a fleet composition and weekly schedules minimizing chartering and sailing costs related to the PSVs. In addition to reducing monetary costs, the application of tactical plans has helped in reducing CO_2 emissions from logistics by 32% since 2011 (Equinor, 2020b). However, the planners have identified that operational variations may result in costs exceeding what was predicted and planned for on the tactical level. An essential operational variation is rough weather conditions, causing delays to the PSVs and forcing deviations from the original plans. Inaccurate estimates of actual order sizes and unforeseen urgent orders may also contribute to these deviations. Correcting actions include rerouting PSVs, sending out extra PSVs, and potentially chartering PSVs from the spot market. Consequences of these actions include additional costs and increases in CO_2 emissions. Further, impact from operational variations and correcting actions may propagate to other parts of the supply chain. Hence, an operational planning tool could be of great value to Equinor.

This thesis addresses the mentioned operational variations by approaching supply vessel planning at the operational level, providing new insights into the domain. By including weather forecasts in the planning process, delays and disruptions may be avoided by allowing PSVs to adjust their sailing speed and service installations at appropriate times. The operational approach to the problem is well suited to account for inaccuracies in the estimates of order sizes, as the problem is solved when these sizes are known. Further, orders may be prioritized by making certain orders optional to service within the planning horizon considered. Prioritizing the orders in such a way may help optimize the supply of the offshore installations and avoid expensive chartering of spot vessels where this is not beneficial. Furthermore, considering pickup orders from the offshore installations allows for better utilization of the PSV capacities, as the voyages are adapted to be able to service more of these orders.

The Time-Dependent Vessel Routing Problem with Speed Optimization (TDVRPSO) considers a set of PSVs servicing offshore petroleum installations' delivery and pickup orders from an onshore supply depot. The problem aims to provide cost-efficient voyages and schedules where weather conditions are adjusted for and sailing speeds are optimized. Weather forecasts are used to decide which time intervals are preferred for a specific voyage and determine the sailing speeds along this voyage. If the demand for services exceeds the capacity of the considered fleet of PSVs, additional PSVs may be hired from a spot market at a higher cost. The total number of PSVs must be able to service the mandatory orders from the offshore installations. Optional orders may also be serviced, but this is not required. The objective of the problem is to minimize the sum of monetary costs asso-

ciated with fuel consumption of the PSVs and chartering of PSVs from the spot market, as well as the penalty costs arising from not servicing the demand for optional orders at the installations. These costs are minimized under restrictions imposed on the visitation of installations, load capacities, delivery deadlines, and other crucial aspects. The output is an operational plan outlining the voyages, schedules, and sailing speeds for all vessels for the next departure day such that delivery and pickup orders are fulfilled.

In solving the TDVRPSO, an arc-flow model is formulated. The model provides an exact solution method for solving smaller-sized instances of the problem. To solve real-sized problem instances, an Adaptive Large Neighborhood Search (ALNS) heuristic is implemented. The ALNS heuristic is tailored to consider the problem specifications in the TDVRPSO. Improving the ALNS heuristic's performance, a local search and a set partitioning problem for combining voyages found throughout the search are included. In order to perform speed optimization and assign reasonable costs to the voyages produced by the ALNS heuristic, the Supply Vessel Speed Optimization Problem (SVSOP) is formulated and included as a subproblem in the ALNS heuristic. Consequently, this thesis contributes to the literature on both vehicle routing problems (VRP) and ALNS heuristics, as we formulate a time-dependent rich VRP with speed optimization and solve real-sized instances with our proposed solution method. Performing speed optimization and allowing for order selection and pickup orders, we provide extensions that are often avoided due to their added complexity. The ALNS heuristic is tested on instances based on real data and produces high-quality solutions. Planning for the considered operational aspects, the results suggest decreases of up to 24% in CO_2 emissions and up to 40% in operational costs for realistic instances.

The problem is described in detail in Chapter 2. Literature related to the problem is reviewed in Chapter 3, and Chapter 4 presents the mathematical formulation of the problem, including a detailed description of a preprocessing procedure developed. Chapter 5 presents a subproblem to the Adaptive Large Neighborhood Search (ALNS) heuristic detailed in Chapter 6. Chapter 7 details the specific case considered in the thesis, including the setting of key parameters and the process of generating problem instances. The computational performance of the ALNS heuristic when applied to this case is investigated Chapter 8, while the managerial insights gained from running the heuristic on various test instances are discussed in Chapter 9. Chapter 10 concludes the thesis before Chapter 11 provides suggestions for future work.

Chapter 2

Problem Description

This chapter presents the Time-Dependent Vessel Routing Problem with Speed Optimization (TDVRPSO). In the TDVRPSO, supply vessel planners route platform supply vessels (PSVs) to service several offshore installations to meet their demand for delivery and pickup of cargo. The supply vessel planners solve the problem daily, and the planned voyages start from the depot on the next departure day. The planning horizon lasts for around three days as the PSVs typically return to the depot after three days. The transformation from the input received by the planners to the decisions made in the TDVRPSO is summarized in Figure 2.1.

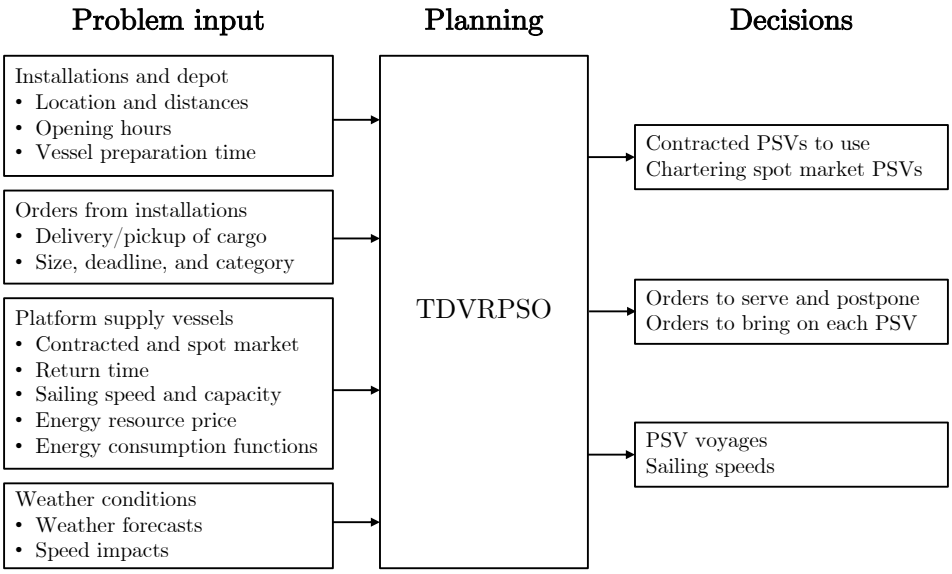


Figure 2.1: Mapping of problem input to decisions in the TDVRPSO.

The TDVRPSO consists of offshore installations that rely on service from a set of PSVs operating from an onshore supply depot. We assume that each installation can be visited at most once in the planning horizon. The locations of all installations and the supply depot are known, together with the intermediate distances between all installations and

all installations and the supply depot. Some of the installations operate under limited opening hours. The opening hours define the intervals of time during the day a PSV can service the installations. It is assumed that PSV deliveries cannot exceed the installations' capacities for storing cargo.

Each installation has demand for delivery and pickup of cargo. Cargo delivery involves shipments of supplies from the depot to the installations, while cargo pickup involves the shipment of used supplies from the installations to the depot. The cargo can be shipped on deck in containers or bulk tanks below deck if it is liquid. The total demand from all installations is represented by a set of orders of certain cargo sizes. Each order is categorized by whether it is a delivery or pickup order and whether it is mandatory or optional to service the order within the current planning horizon. If an optional order is not serviced within the planning horizon, it is postponed to the next planning horizon. It is assumed that delivery orders can be either mandatory or optional, while pickup orders will always be optional. This leads to three order types: *mandatory delivery orders*, *optional delivery orders*, and *optional pickup orders*. Orders of the same type from the same installation are aggregated into one order.

The set of PSVs consists of available vessels from the contracted fleet, referred to as *fleet vessels*, and, potentially, chartered vessels from the spot market, referred to as *spot vessels*. Spot vessels will only be chartered when the fleet vessels cannot handle the mandatory demand from the installations. A certain amount of preparation is required before a PSV can depart from the depot. This preparation process is referred to as *vessel preparation*. Vessel preparation involves loading cargo onto the PSV deck, filling the bulk tanks, and other vessel-specific activities. The vessel preparation takes place during limited opening hours with fixed start and end times every day. These times are equal for all PSVs. Only the PSVs present at the supply depot when vessel preparation starts are considered available when solving the TDVRPSO. All vessels leave the depot immediately after vessel preparation at the fixed end time. A return time must also be specified for each PSV. This deadline is required to make other planning activities more predictable, as other operations may require the PSVs. Upper and lower limits on the sailing speed of the PSVs are provided. When sailing in rough weather conditions, the upper limit on sailing speed is reduced. Each PSV's load capacity limit is also provided.

The PSVs service installations on voyages. A *voyage* consists of a sequence of installations that are to be visited by a given PSV. Each available PSV performs at most one voyage in the planning horizon. The maximum length of a voyage is restricted by the return time of the PSV. However, there are no restrictions on how many platforms a PSV can visit on a voyage. A PSV on a voyage can perform one of four different *activities*: (1) vessel preparation at the depot before departure, (2) sailing between installations and to and from the depot, (3) idling at an installation while waiting for the commencement of service, and (4) servicing an installation by delivering and picking up cargo.

To calculate the cost of energy consumption, the price of the energy resource used, e.g., marine diesel oil, is provided as input. The activities have unique energy consumption functions depending on different factors. The energy consumed during vessel preparation only depends on the duration of the activity. When a vessel is sailing, the energy consumption depends on the duration, sailing speed, and weather conditions. For idling at and servicing an installation, energy consumption depends on the duration and weather conditions. The servicing duration is given by a function varying with the amount of cargo that needs to be unloaded and loaded. Figure 2.2 summarizes which factors the different energy consumption functions depend on.

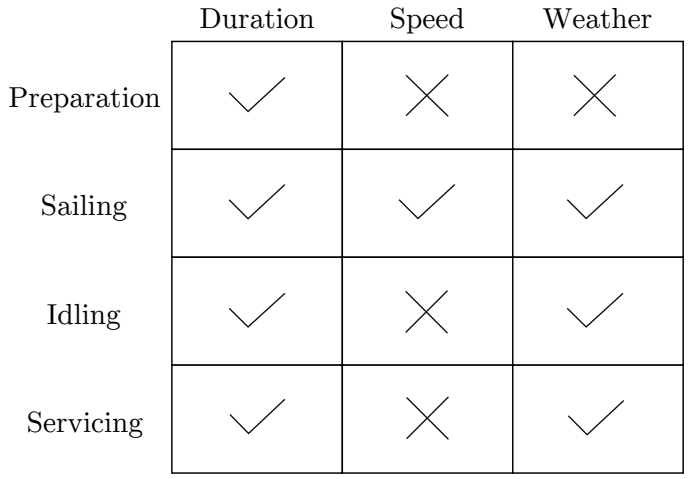


Figure 2.2: Factors in the energy consumption of the four vessel activities

Weather dependency is the factor that makes the TDVRPSO *time-dependent*. As the weather conditions at a given time affect the operational costs, specifically the energy costs of performing vessel activities, the costs of performing an activity will depend on the time the activity is performed. As an example, consider a vessel sailing between two installations. If the vessel starts at t_1 and ends at t_2, the weather in this time interval results in cost c_{1-2} as the weather conditions are good. However, if the vessel starts at t_3 and ends at t_4, the weather may be worse, yielding a higher cost, c_{3-4}. Hence, the costs in the problem are time-dependent, making the problem in this thesis time-dependent.

The weather conditions are provided as input to the problem and refer to the significant wave height provided by a weather forecast. The significant wave height is defined as the mean height of the highest third of the waves. Higher significant wave height, referred to as rough weather, yields a higher energy consumption in all activities. As previously stated, the upper limit on sailing speed is also reduced when sailing in rough weather. Furthermore, due to safety measures, the installations are closed for service when the wave height exceeds a provided safety limit.

The objective of the TDVRPSO is to minimize the sum of monetary costs and penalty costs associated with the problem. The monetary costs consist of the variable operating

costs of the contracted fleet and the costs of chartering and operating PSVs from the spot market. Variable operating costs include the costs of the energy consumed by all vessels. Fixed operating costs include costs associated with chartering PSVs. The penalty costs represent the penalties involved when an optional delivery or pickup order is not serviced within the planning horizon. The penalty costs are specified per order.

Following the problem's objective, the TDVRPSO outputs cost-optimal voyages, schedules, and order selection plans for the available contracted PSVs and potentially additional spot market PSVs. This output represents the three types of decisions made in the TDVRPSO. The first type determines which of the available PSVs from the contracted fleet to use and whether external spot market PSVs are needed. The second type lays out which optional delivery and pickup orders should be postponed and assigns the remaining orders to the chosen PSVs. The third type is the voyages that the chosen PSVs shall sail and the sailing speeds that should be held along these voyages. As stated above, a higher sailing speed implies a higher energy consumption, leading to higher monetary costs. As the TDVRPSO seeks cost-optimal solutions, the energy consumption will be minimized, adding emission reduction to the scope of the TDVRPSO.

2.1 Example Problem

This section presents an example of the TDVRPSO and a corresponding feasible solution. A set of 13 offshore installations is provided. Five of the installations have placed orders that need to be serviced by a fleet of PSVs. These orders are summarized in Table 2.1.

Table 2.1: Orders from the five installations in the example problem. Order sizes are given in cargo units, i.e., the space one offshore container occupies on deck (HACON Containers, 2020).

Order	**Installation**	**Mandatory/Optional**	**Delivery/Pickup**	**Size** [cargo units]
1	1	Mandatory	Delivery	15
2	1	Optional	Delivery	7
3	4	Mandatory	Delivery	25
4	4	Optional	Pickup	30
5	6	Mandatory	Delivery	7
6	6	Optional	Delivery	9
7	11	Mandatory	Delivery	12
8	11	Optional	Pickup	10
9	13	Mandatory	Delivery	20
10	13	Optional	Delivery	7

The fleet consists of four PSVs: *PSV1*, *PSV2*, *PSV3*, and *PSV4*. A group of planners determines voyages for the available fleet vessels leaving the supply depot the next day,

i.e., the departure day. *PSV1* and *PSV2* will be available, while *PSV3* and *PSV4* are unavailable as they have not returned from their current voyages when vessel preparation starts at 08:00 in the morning on the departure day. If the available fleet vessels *PSV1* and *PSV2* are not able to service all mandatory demand from the installations, the planners can hire a spot vessel at a high charter rate. The vessels must be back at the depot before 08:00 after three days and must stay within the lower and upper speed limits of 10 and 14 knots, respectively. The capacities of the PSVs are shown in Table 2.2.

Table 2.2: PSV capacities in the example problem given in cargo units, i.e. the space one offshore container occupies on deck (HACON Containers, 2020).

PSV	*PSV1*	*PSV2*	*PSV3*	*PSV4*
Capacity [cargo units]	110	85	100	95

The hourly forecast for the significant wave height for the next three days is given in Figure 2.3. The forecast shows that the significant wave height increases steadily during Day 0 and 1 before it exceeds the critical limit of 4.5 meters. At this point, the installations are closed for servicing due to safety reasons. A few hours after midnight on Day 2, the significant wave height drops below the limit, reopening the installations for servicing.

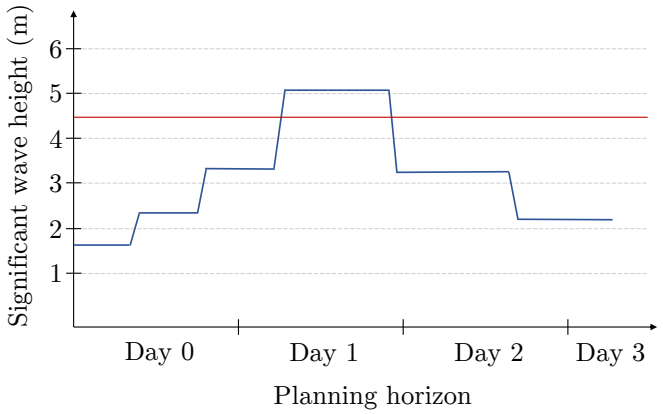


Figure 2.3: Weather forecast in the example problem. The horizontal red line marks the significant wave height above which installations close.

The installation opening hours are given in Figure 2.4 and show that all installations except installation 1 are open for service at all times. In this example, installation 1 is open for servicing between 10:00 and 22:00.

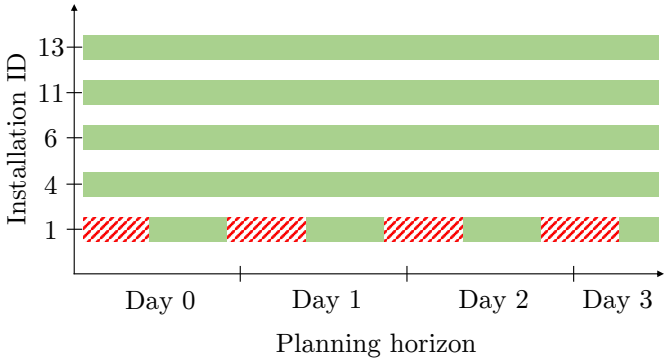


Figure 2.4: Installation opening hours in the example problem. The dotted red areas mark time periods where installation 1 is closed.

Figure 2.5 shows a feasible solution to the example problem. Both available fleet vessels are used, and no spot vessel is chartered. *PSV1* visits installations 1, 13, and 11, while *PSV2* visits installations 4 and 6. All orders are fulfilled. The green and red numbers on the map indicate how the load is updated along each voyage. *PSV1* (solid line) starts with 67 units on deck when leaving the depot, as this is the sum of all delivery orders from installations 1, 11, and 13. It then delivers 22 units to installation 1, 27 units to installation 13, and 18 units to installation 11. At installation 11, it picks up 12 units that it brings back to the depot. This is also shown along *PSV2's* voyage (dashed line).

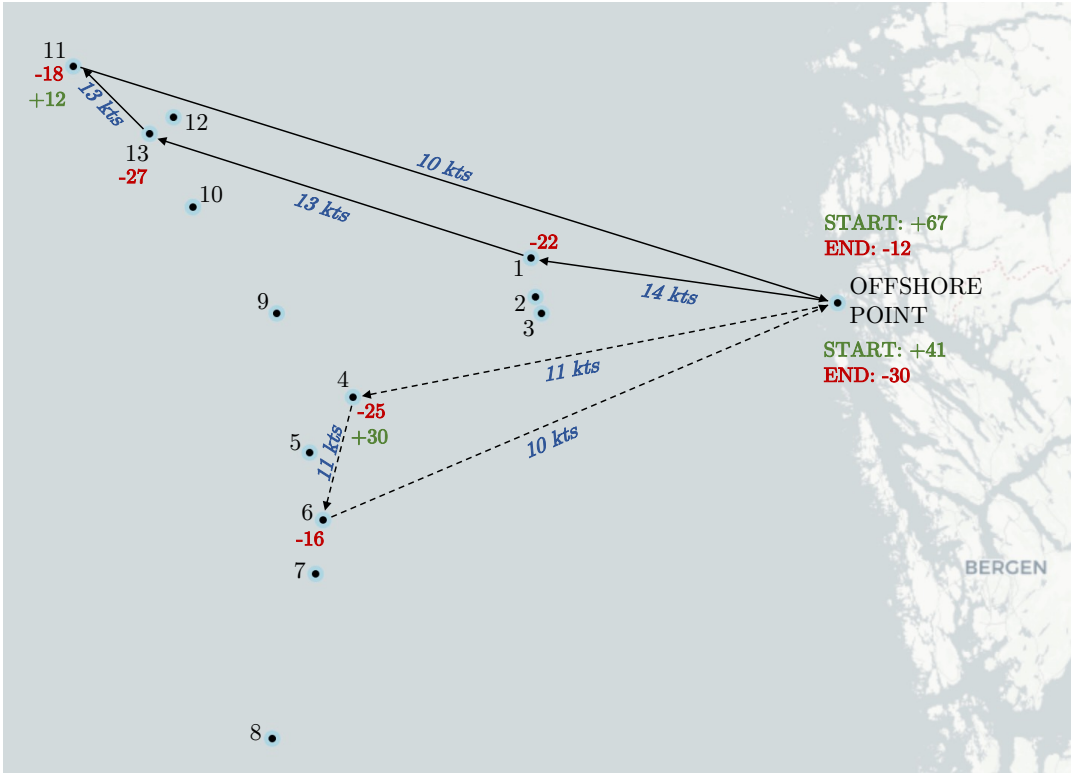


Figure 2.5: Feasible voyages solving the example problem. The solid line is *PSV1's* voyage, and the dashed line is *PSV2's* voyage. The red numbers represent delivery and the green numbers represent pickup.

Figure 2.5 also shows the speeds the PSVs sail at. The speeds are further studied in the speed profiles in Figure 2.6 and Figure 2.7. As shown by the weather forecast and installation opening hours in Figures 2.3 and 2.4, the installations will be closed at certain times in the planning period. This solution avoids arriving at closed installations by utilizing speed optimization and visiting the installations in a smart sequence.

To avoid arriving at installation 1 when it is closed between 22:00 and 10:00, *PSV1* visits this installation first and speeds up to be able to finish servicing before 22:00. Another reason for the *PSV1* speed-up is that it must finish servicing all installations before the installations close due to weather at 06:30 on Day 1. Therefore, it sails at 13 knots between installation 1 and 13 and between installation 13 and 11. As Figure 2.6 shows, it manages to finish servicing at 06:24. *PSV2* is not that short on time, so it sails at 11 knots from the

depot to installation 4 and from installation 4 to installation 6. Notice that both PSVs sail at 10 knots back to the depot. The vessels are in no hurry when they return to the depot, as they are well within the specified required return time. Therefore, the vessels sail at 10 knots, which is the most energy-efficient speed.

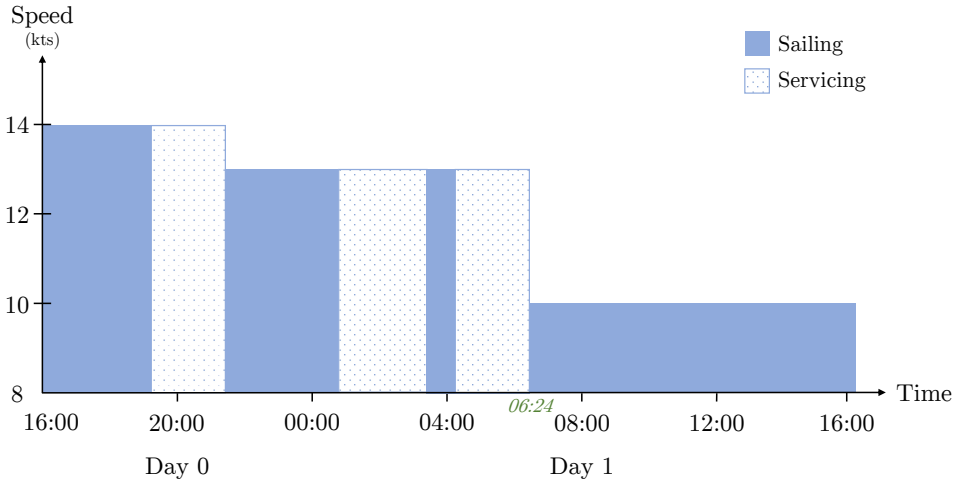


Figure 2.6: Speed profile for *PSV1*. The profile shows the PSV speeding up (14 kts) to complete servicing at installation 1 within opening hours. The speed is then kept at a high level (13 kts) to service the two remaining installations before they close due to weather. Finally, the speed is reduced to 10 kts for the final sailing back to the depot.

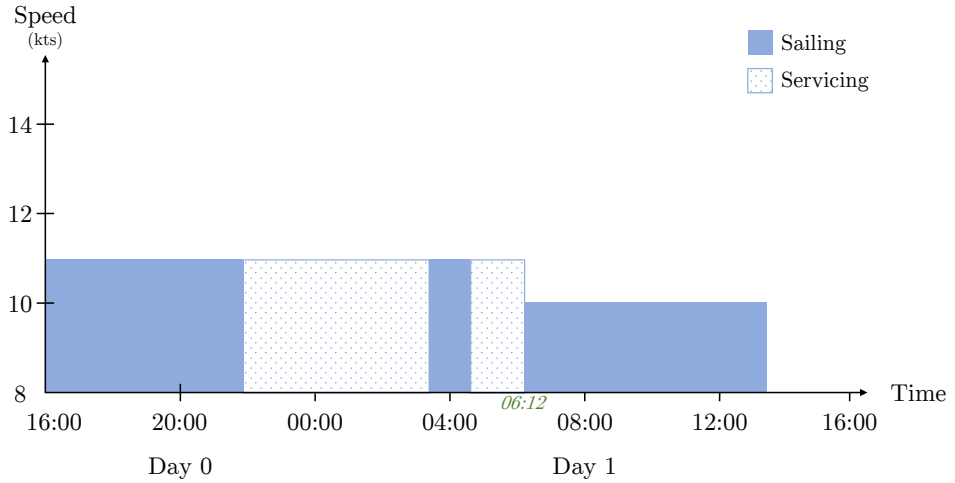


Figure 2.7: Speed profile for *PSV2*. The profile shows the PSV keeping a constant speed (11 kts) in order to have time to service all installations before they close due to weather. After the final installation is serviced, the speed is reduced to 10 kts for sailing back to the depot.

Chapter 3

Literature review

This chapter reviews the existing literature found relevant for the Time-Dependent Supply Vessel Planning Problem with Speed Optimization (TDVRPSO), building on the review presented by Ulsrud and Vandvik (2020). Section 3.1 deals with the Supply Vessel Planning Problem (SVPP), which is a problem at the strategic and tactical levels. This thesis considers planning on the operational level, but literature on the SVPP is reviewed to provide useful insights on supply vessel planning. Further, the routing problems relevant to the problem considered in this thesis are reviewed in Section 3.2. These include problems handling pickups, deliveries, weather, and time dependency. Where the solution methods used are relevant to those presented in this thesis, this is also emphasized. In addition to being supported by weather and time dependency, the operational nature of the TDVRPSO is highlighted through performing speed optimization. Speed optimization is discussed in Section 3.3. Finally, Section 3.4 focuses on this thesis' contribution to the literature on problems concerning offshore logistics and supply vessel planning.

3.1 The Supply Vessel Planning Problem

The Supply Vessel Planning Problem (SVPP) concerns the case of determining the optimal fleet of supply vessels and their corresponding voyages and schedules from an onshore supply depot to a set of offshore installations. The SVPP was first addressed by Fagerholt and Lindstad (2000). The authors aimed to evaluate the effects on the total supply cost of having some or all offshore installations closed for service at night and determine an optimal routing policy. Their recommended solution involves two large vessels sailing two weekly schedules each and making a single visit to all installations. For this schedule to be feasible and robust, all offshore installations had to be open at all times. Halvorsen-Weare et al. (2012) consider the SVPP on a tactical level in their voyage-based formulation of the SVPP, solving the problem in two phases. The first phase generates feasible candidate voyages. The second phase applies the voyages from the first phase in a voyage-based model

to find the optimal fleet composition and associated weekly routes and schedules. This voyage-based model is compared to an arc-flow model by Halvorsen-Weare and Fagerholt (2017). For larger problem instances, the voyage-based model outperforms the arc-flow model introduced. However, the arc-flow model provides a more detailed description of the problem. In this thesis, an arc-flow model is used in the exact solution method addressing the problem under consideration.

Shyshou et al. (2012) also build on the two-phase modeling approach of Halvorsen-Weare et al. (2012), but develop a Large Neighborhood Search (LNS) heuristic instead of using a voyage-based formulation. The heuristic outperforms the voyage-based model for large instances but does not guarantee an optimal solution. Kisialiou et al. (2018a) extend the formulation of the Periodic Supply Vessel Planning Problem (PSVPP) introduced by Halvorsen-Weare et al. (2012) to include flexible departure times and coupled vessels and propose a voyage-based model that may be solved exactly. Further, the Adaptive Large Neighborhood Search (ALNS) developed yields near-optimal solutions for small to medium-sized instances and outperforms previous heuristics developed for the same problem when handling larger instances. In this thesis, an ALNS is also introduced to solve real-sized instances of the TDVRPSO.

Borthen et al. (2019) define and solve a bi-objective formulation of the SVPP, generalizing the hybrid genetic search heuristic with adaptive diversity control proposed in Borthen et al. (2018). The considered problem involves a single and fixed departure time, a homogeneous fleet of vessels, and no time windows. The authors focus on the trade-off between cost and persistence. In this context, persistence concerns finding solutions with few changes from the previous solution, implying consistency. For real-sized instances, the costs of the solutions found are less than 1% higher than the costs found when applying a single-objective heuristic, suggesting that the solutions are not extensively affected by adding persistence as an objective.

Uncertainty in the SVPP was first addressed by Halvorsen-Weare and Fagerholt (2011). The authors focus on the impact of weather on sailing speed and the loading and unloading operations performed by supply vessels and present various approaches to construct robust schedules in the SVPP. Robust approaches are also constructed by Kisialiou et al. (2018b). Further, the demand requested by offshore installations in the SVPP may fluctuate and challenge the capacities of the PSVs. Pursuing a more reliable supply vessel planning and scheduling, Kisialiou et al. (2019) present a methodology for addressing stochasticity in demand. They generate delivery schedules with different levels of reliability and solve the problem using a combination of an ALNS and a discrete event simulation procedure. Maisiuk and Gribkovskaia (2014) introduce stochastic sailing and service times when evaluating different fleet size configurations in the SVPP.

3.2 Routing Problems Related to the TDVRPSO

This section presents routing problems that are related to the TDVRPSO. As the problem allows both pickup and delivery orders, vessel routing problems with pickups and deliveries are discussed in Section 3.2.1. Both pickups and certain deliveries are made optional in the TDVRPSO. Hence, routing problems with order selection are presented in Section 3.2.2. In performing speed optimization, dependency on weather and time must be addressed. Relevant literature connecting these aspects to routing problems is presented in Section 3.2.3 and Section 3.2.4.

3.2.1 Vessel Routing Problems with Pickups and Deliveries

While the SVPP presented in Section 3.1 is a planning problem on the strategic and tactical levels, there are also crucial operational planning considerations in offshore logistics. Some of these considerations are presented in Vessel Routing Problems with Pickups and Deliveries (VRPPD).

Berbeglia et al. (2007) present a survey of different pickup and delivery problems and a three-field classification scheme for these problems. These fields are many-to-many problems, one-to-one problems, and one-to-many-to-one problems. The survey also details methods for solving these types of problems. Numerous studies discuss routing problems with pickups and deliveries. This section focuses on offshore logistics problems.

Jahre et al. (2007) suggest an approach to find optimal voyages for supply vessels by extending the Single Vessel Routing Problem with Pickup and Deliveries (SVRPPD) covered by Gribkovskaia et al. (2007). This problem includes constraints on customer capacities and investigates how this affects the vessel routing. In the SVRPPD, each location may be visited twice during the planning horizon. Both delivery and pickup may be performed during the first visit. However, the formulation also allows for performing delivery during the first visit and pickup during the second. Gribkovskaia et al. (2008a) build on Jahre et al. (2007) when formulating the Single Vessel Routing Problem with Capacitated Customers. In this problem, each customer must be visited. For solving the problem, the authors propose multiple construction heuristics and a tabu search algorithm. Sopot and Gribkovskaia (2014) also consider a single vessel and allow a maximum of two visits to the same installation. The authors propose a metaheuristic algorithm that outperforms a commercial solver when solving medium-sized instances. In the context of pickup and delivery problems, the problems of Jahre et al. (2007), Gribkovskaia et al. (2008a), and Sopot and Gribkovskaia (2014) may be classified as one-to-many-to-one problems. All problems concern a single vessel making deliveries to customers from a supply depot and collecting pickups to be returned to the same depot. Such problems may also be formulated with multiple vessels, which is the case for this thesis.

Another approach to the VRPPD is taken by Cuesta et al. (2017) when introducing the Vessel Routing Problem with Selective Pickups and Deliveries (VRPSPD). The problem is formulated for both the single and multi-vessel case, dividing pickup and delivery operations into pickup and delivery nodes. This ensures that pickup and delivery operations are performed non-simultaneously. Each installation is coupled to one pickup node and one delivery node, which both may have several orders associated with them. The total load for each vessel must not exceed its total capacity, ensured by load continuity constraints. The handling of different order types and load continuity by Cuesta et al. (2017) is similar to what is presented in this thesis.

3.2.2 Routing Problems with Order Selection

Classical vehicle or vessel routing problems with pickups and deliveries may be extended by allowing pickups or deliveries to be selective. When presenting the Single Vehicle Routing Problem with Unrestricted Backhauls, the servicing of all backhauls is one of the common prevalent assumptions that Süral and Bookbinder (2003) relax. They propose a mixed-integer model that minimizes the total tour cost by choosing the best optional backhauls and placing these in a revised tour from the original. The model is solved exactly for small instances of customers with either delivery or pickup demand associated with them. Gribkovskaia et al. (2008b) model customers with demand for both pickup and delivery, and allow multiple visits to the same customer. The authors present the Single Vehicle Routing Problem with Deliveries and Selective Pickups (SVRPDSP), which concerns designing minimal cost routes including all deliveries and a subset of the pickups in question. In this problem, a pickup revenue is associated with each vertex and serviced if profitable to do so. This makes the net cost of a route equal to the difference between the total routing costs and the total collected revenue. Their handling of selective orders is similar to that of Süral and Bookbinder (2003). Building on the exact branch-and-bound algorithm by Süral and Bookbinder (2003), Gutiérrez-Jarpa et al. (2009) propose a branch-and-cut method in solving the SVRPDSP exactly, enabling satisfactory solutions to instances previous methods were unable to solve. The SVRPDSP is also addressed by Bruck et al. (2012), Coelho et al. (2012) and Coelho et al. (2016). Bruck et al. (2012) propose using an evolutionary algorithm benefiting from data mining strategies in its crossover and mutation procedures to identify good characteristics from the population considered. Another heuristic approach is introduced by Coelho et al. (2012) in their hybrid heuristic approach, inspired by a variable neighborhood search (VNS). Coelho et al. (2016) continue this work, aiming to provide routes in satisfactory computational time.

Ting and Liao (2013) propose a memetic algorithm for solving the Selective Pickup and Delivery Problem. Specifically, this problem aims at finding the shortest route that can supply *all* delivery nodes from *some* pickup nodes, and hence the pickups are treated as optional. In this thesis, pickup orders are also treated as optional.

Whereas the storage capacities of the vehicles is a restriction in most studies on routing problems with order selection, Gutiérrez-Jarpa et al. (2010) introduce time restrictions in the Vehicle Routing Problem with Deliveries, Selective Pickups, and Time Windows. Consequently, this addition affects the solutions obtained as it renders the servicing of customers outside of the given time windows impossible. The Vessel Routing Problem with Selective Pickups and Deliveries (VRPSPD) presented by Cuesta et al. (2017) also introduce a variation by allowing for selective deliveries as well as selective pickups, both for the single- and multi-vessel case. As described in Section 3.2.1 the problem aims to minimize the sum of the total routing costs and the total costs of orders not serviced. Since all orders are optional, penalty costs are essential to which routes are found to be optimal. The authors assign penalty costs to orders based on urgency and importance. Variations in these factors between different orders to the same platforms make consolidation of orders generally not possible. Most realistic instances of the problem were solved using a commercial solver. However, an Adaptive Large Neighborhood Search (ALNS) was required to reduce the computational time in the multi-vessel case. The ALNS introduced moves nodes instead of orders, making an order selection problem necessary. This problem has to be solved whenever an order is inserted into a solution. The order selection is handled through a greedy heuristic, sorting orders by decreasing penalty costs and inserting them sequentially until this is no longer feasible. As this may not necessarily yield an optimal set of orders for a specific route, post-optimization is performed in negligible time compared to the main search procedure. This thesis presents an ALNS heuristic in which orders are moved, and is not dependent on a separate order selection problem.

3.2.3 Weather-Dependent Routing Problems

Previous studies on the Supply Vessel Planning Problem (SVPP) and other offshore logistics problems have shown that the resulting plan is significantly affected by the operating weather conditions. Specifically, this means that factors such as wind speed and the height and direction of waves may disrupt or potentially prevent PSVs from realizing the planned routes. Hence, integrating weather effects in the relevant problems is of interest. In general, operational planners consider the wave height to be the most influential factor on sailing and servicing time, as is mentioned by Halvorsen-Weare and Fagerholt (2011) and Kisialiou et al. (2018b).

When aiming to create robust solutions to the supply vessel planning problem, Halvorsen-Weare and Fagerholt (2011) account for the weather impact by introducing four weather states. These states are defined from significant wave heights, i.e., the average of the one-third highest waves in 20 minutes, influencing the sailing speeds and servicing times of the PSVs. This treatment of weather as a univariate variable of wave height is equivalent to what is done by Norlund et al. (2015) and Moan and Ødeskaug (2020), and how weather is treated in this thesis.

Kisialiou et al. (2018b) focus on robustness when addressing the Periodic Supply Vessel Planning Problem (PSVPP), specifically the effects of weather on the schedules generated. In this context, a voyage is deemed robust if it is feasible for all weather conditions. The problem consists of generating repeatable weekly schedules able to maintain stability and predictability. In order to account for travel- and service-time delays, the authors introduce *intra-* and *inter-voyage slack*. These two types of slack refer to a time buffer for each of the installations visited during a voyage and a time buffer between two consecutive voyages for the same vessel, respectively. The amount of slack assigned is controlled by a robustness parameter depending on the current weather conditions. Inspired by Kisialiou et al. (2018a), the problem is solved using an ALNS. Pursuing the schedule with the highest robustness, the ALNS is included in an algorithm performing a binary search to find the highest robustness parameter. Realistic and large-sized instances of the problem are solved.

Disruptions in supply vessel logistics related to weather conditions and unforeseen events are addressed from an operational perspective by Albjerk et al. (2016), introducing weather-dependent parameters. The goal is to handle disruptions so that a supply vessel may return to the long-term plan before the next voyage is scheduled. Hence, the authors are faced with the trade-off of minimizing costs and maintaining the planned service level. The pickup and delivery problem considered is modeled both as an arc-flow and a path-flow model, where the path-flow model is found to be superior in terms of computational performance for realistic instances. However, exact solutions for real-sized instances are not found within a satisfactory time, encouraging a heuristic approach. This is done by Stålhane et al. (2019), addressing the same problem as Albjerk et al. (2016). A construction heuristic provides an initial solution to a variable neighborhood search (VNS), which alternates between two phases; local minima are pursued in the *intensification* phase, and exploration of the solution space is carried out in the *diversification* phase. Stålhane et al. (2019) compare their solutions to those find by Albjerk et al. (2016), and found superior results for all but the smallest instances.

As implied in this section, weather dependency affects the sailing and servicing times of offshore supply vessels. Hence, there will be an impact on the optimal sailing speeds for different weather conditions. This will be further detailed in Section 3.3.

3.2.4 Time-Dependent Routing Problems

To the operational problem considered in this thesis, time dependency is an essential feature. The Time-Dependent Vehicle Routing Problem (TDVRP) incorporates time dependency by making travel times, and possibly also servicing times, dependent on the time at which trips are performed. For onshore routing, time dependency is mainly related to variations in traffic, while weather conditions introduce dependency in offshore routing. Malandraki and Daskin (1992) define and formulate the TDVRP and discuss key charac-

teristics and heuristics for solving it. The Time-Dependent Travelling Salesman Problem is investigated as a special case of the TDVRP. Ichoua et al. (2003) present a model that satisfies the "first-in-first-out"-property, i.e., that the first of multiple vehicles leaving a node i for a node j will arrive at node j first. When modeling, the authors disregard the assumption of constant speed over the entire arc, allowing for speed to change from one time to the next. Christiansen et al. (2017) also allow for speeds to vary when introducing the fuel supply vessel routing problem (FSVRP). This problem is a version of the multi-trip vehicle routing problem with dependent sailing times, introducing a time-discrete model to account for the complexity of sailing times varying with the time of day. Similarly, this thesis discretizes time to account for weather effects on the sailing times of platform supply vessels.

Further, several solution methods for the TDVRP have been introduced. Dabia et al. (2013) propose a branch-and-price algorithm with time windows. The master problem is a set partitioning problem derived from decomposing an arc-flow formulation. The pricing problem is solved as a time-dependent shortest path problem with resource constraints. Addressing the Time-Dependent Pollution Routing Problem, Franceschetti et al. (2017) develop a metaheuristic based on an ALNS, using a departure time and speed optimization procedure to decide on optimal departure times and vehicle speeds. Ma et al. (2017) investigate a combined order selection and time-dependent vehicle routing problem with time windows in the context of perishable product delivery. The authors propose a hybrid ant colony algorithm for solving the problem in acceptable computational time.

3.3 Speed Optimization in Offshore Logistics

Studies on planning problems related to offshore supply vessels have shown increased interest in reducing emissions as the primary objective. More efficient planning will help offshore supply chains reduce their environmental impact and lower fuel consumption costs related to operating supply vessels.

The issue of reducing emissions in supply vessel planning is addressed by Norlund and Gribkovskaia (2017). The authors investigate possible reductions in fuel consumption from choosing appropriate speeds given different weather conditions. Their modeling of weather is inspired by that of Halvorsen-Weare and Fagerholt (2017), detailed in Section 3.2.3. The paper evaluates two of the four different speed optimization strategies presented by Norlund and Gribkovskaia (2013) and applies them to pre-generated voyages, linking speed optimization to varying weather conditions. When applying these strategies on a voyage, it is necessary to assume that the time windows and the days of visits to installations closed at night are unchanged after speed optimization. Moreover, the optimized speed on a voyage leg is not allowed to be less than the defined minimum vessel speed. For the first strategy, if the vessel in question is facing waiting time, the speed is set to be

the leg distance divided by the sum of the sailing time and waiting time or the lowest allowable speed given the circumstances. The second strategy focuses on voyages in their entirety and optimizes speed recursively. The minimum voyage speed is, for all voyages, calculated as the ratio of distance to the sailing and waiting time considered. When considering cases where arrival times at installations exceed the closing times, the voyage with the largest violation is split into two partial voyages. The speeds for the two partial voyages are then optimized. This process is repeated until all legs in the pre-generated voyages are considered. Subsequently, a discrete event simulation model decides how the different weather states in the planning horizon affect fuel consumption and voyage duration. Hence, the problem outputs speed-optimized schedules for all vessels that are simulated to account for weather impacts.

Norstad et al. (2011) address speed optimization in tramp ship routing and scheduling. The authors introduce speed variables in extending the fixed speed arc-flow formulation presented by Christiansen et al. (2007). Real-size instances are approached using a multi-start local search heuristic for solving the problem addressed within reasonable time. The authors set up a subproblem referred to as the Speed Optimization Problem (SOP), and propose various methods for solving this problem. The first method bases on discretizing arrival times within the time windows of the respective nodes, similar to what was done by Fagerholt (2001) and later by Fagerholt et al. (2010). The problem is solved as a Shortest Path Problem (SPP) on an acyclic graph. The second method is based on using a Recursive Smoothing Algorithm (RSA), taking advantage of the fuel consumption function being convex for a range of feasible speeds. From this, speeds should be held constant and as low as possible for a given leg. This method performs better than the SPP in terms of computational time.

When addressing the RoRo-shipping routing problem, Andersson et al. (2015) handle speed optimization differently from what is presented up until this point. The authors introduce an arc-flow model and a rolling horizon heuristic to solve the combined problem of planning routes and optimizing vessel speeds on these routes. Further, they introduce a speed decision variable that weighs speed alternatives for a vessel along an arc. A non-linear, convex, and strictly increasing fuel consumption function is approximated as linear combinations of neighboring discrete speed values. Consequently, the curve provides a slight overestimate of the fuel consumption.

Another approach to speed optimization is taken by Moan and Ødeskaug (2020) when introducing the Operational Supply Vessel Planning Problem with Speed Optimization (OSVPPSO). The authors consider discrete time intervals and use weather forecasts and installation opening hours when generating feasible arcs in a preprocessing procedure. The generated arcs represent various speed alternatives. The arc-flow model presented chooses the optimal arcs, and consequently, the optimal speeds. This approach to speed optimization is similar to approach taken in this thesis.

In order to do speed optimization for PSVs, estimating fuel consumption is essential. Each vessel exhibits a unique fuel consumption function detailing the relationship between speed and fuel consumption, complicating obtaining exact functions for specific vessels. Psaraftis and Kontovas (2014) study fuel consumption functions that display a cubic increase in consumption per distance when increasing speed. Such functions are considered good approximations for a range of PSVs, especially those of small size. The authors approach speed optimization by calculating fuel consumption per unit time as a polynomial function of time and payload. They point out that the optimal speeds are apt to increase if chartering rates are high, causing important considerations for hiring spot market PSVs. Norstad et al. (2011) apply a quadratic function when estimating fuel consumption. As mentioned, Andersson et al. (2015) use a non-linear, convex and strictly increasing fuel consumption function. Finally, Norlund and Gribkovskaia (2017) also take into account the impact of weather, i.e., the significant wave height experienced, and how this reduces the fuel efficiency of the vessel. Finally, Lindstad et al. (2013) consider emissions by varying speed with sea conditions and the freight market, providing the optimal speeds for dry bulk vessels. The maximum economic speeds considering hydrodynamic characteristics were lower than the design speeds, even in a good freight market.

3.4 Our Contribution

The supply vessel planning problem and related offshore routing problems are well studied, but mainly from the strategic and tactical levels. While studies on weather conditions have been incorporated to perform speed optimization in various studies, an operational approach has only been explored by Moan and Ødeskaug (2020). The majority of studies on offshore vessel routing only consider mandatory delivery orders, disregarding the possible variations in the importance of orders and the added benefits of allowing for pickup orders at the installations visited by supply vessels. To the best of our knowledge, this thesis is the first to investigate and perform time-dependent speed optimization while allowing for order selection of both delivery and pickup orders. The aim is to provide insights related to how these considerations may lead to cost reductions, lower emissions, and a more accurate depiction of the situation faced by supply vessel planners.

Chapter 4

Mathematical Model

This chapter presents the mathematical model of the TDVRPSO. The model is a time-discrete arc-flow model consisting of two phases. The first phase, referred to as the *arc-generation procedure*, handles some of the problem constraints and generates time-discrete, feasible arcs with respect to these constraints. The network of arcs is input to the model in the second phase. The model finds the cheapest combination of arcs that satisfies the constraints that are not handled in the first phase while optimizing the vessels' speeds and order fulfillment. Section 4.1 presents the modeling assumptions, while Section 4.2 describes the approach taken to model the problem as an arc-flow model. This includes a description of how time is discretized to handle non-linearities in the problem, a definition of the nodes and arcs in the model, and the procedure for generating arcs. Section 4.3 presents the mathematical formulation of the TDVRPSO as an arc-flow model.

4.1 Modeling Assumptions

The problem is interpreted as a network graph, where each order or the depot is a node at a specific point in time. The directed arcs between these nodes represent a vessel servicing orders or sailing from or to the depot. The voyages vessels may sail are given as a combination of directed arcs. Key assumptions when modeling the TDVRPSO this way are listed below.

- A vessel arriving at an open installation will commence service immediately after arriving. Hence, idling will only occur when an installation is closed, either due to rough weather conditions or opening hours.
- The servicing of an installation must happen continuously. It is not possible to start servicing, get interrupted by rough weather conditions or opening hours, and continue servicing at a later time.

- The weather forecasts are treated as deterministic, even though they are stochastic by nature. This is justified by the short planning horizon.
- The opportunity to charter from the spot market is represented by the ability to charter one PSV with roughly the same attributes as the ships in the long-term contracted fleet at a high chartering cost.
- Unloading bulk cargo is assumed to be faster than unloading deck cargo, and the demand for bulk cargo is assumed to be met by only considering the demand for deck cargo. Hence, only the demand for deck cargo is considered.

4.2 Modeling Approach

Section 4.2.1 describes how weather dependency and speed optimization are handled in the model. Section 4.2.2 defines the nodes and arcs in the arc-flow model before Section 4.2.3 outlines the procedure that generates the arcs that are input to the mathematical model.

4.2.1 Handling Speed Optimization and Weather Dependency

The fuel consumption from sailing depends on the sailing speed. To optimize vessel speeds, speed could be a decision variable in the mathematical model of the TDVRPSO. If so, the relationship between speed and fuel consumption would need to be linear to solve the model with linear programming. However, modeling the fuel consumption as a linear function of the sailing speed does not provide a realistic approximation. Therefore, this thesis will incorporate a non-linear fuel consumption approximation function as it is desirable to model fuel consumption realistically.

As stated in Chapter 2, the fuel consumption and, consequently, the costs of performing vessel activities are time-dependent in the TDVRPSO. This further complicates the approximation of fuel consumption as there will exist multiple non-linear fuel consumption curves for a vessel sailing from one installation to another. A vessel departing from an installation at t_1, arriving at the destination at t_2, will have a different fuel consumption curve if it departs at t_3 and arrives at t_4 instead. Hence, it is not enough to only assign sailing speeds to vessels sailing between installations to calculate fuel consumption costs; we must also know the departure and arrival times to use the correct fuel consumption curve. The relationship between vessel speed and fuel consumption is outlined in Figure 4.1. The red curve describes the fuel consumption in rougher weather conditions than the blue curve.

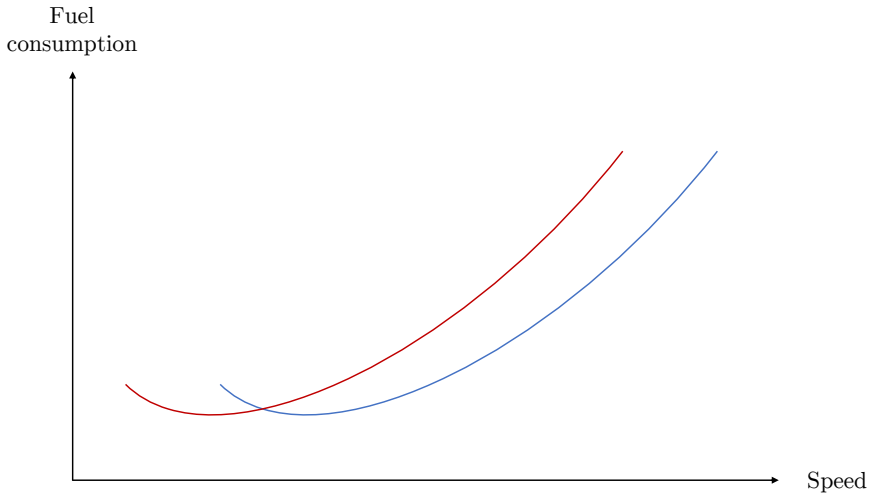


Figure 4.1: General visualization of fuel consumption as a function of speed for different weather conditions. The blue consumption curve represents consumption for the highest significant wave height of the two, i.e., the worst weather.

To handle the non-linearities introduced by the weather dependency and speed optimization, time is made discrete and attributed to the nodes in the arc-flow model. Let a node be defined as an installation being visited by a vessel at a certain point in time, and the arcs represent a vessel sailing between installations. This will be redefined in the next section, but we pretend that this is the case for the sake of this argument. As arcs are defined by their start and end nodes, the arcs will get a start and end time. With these definitions, possible arcs between two installations can be visualized in a time-space diagram such as in Figure 4.2.

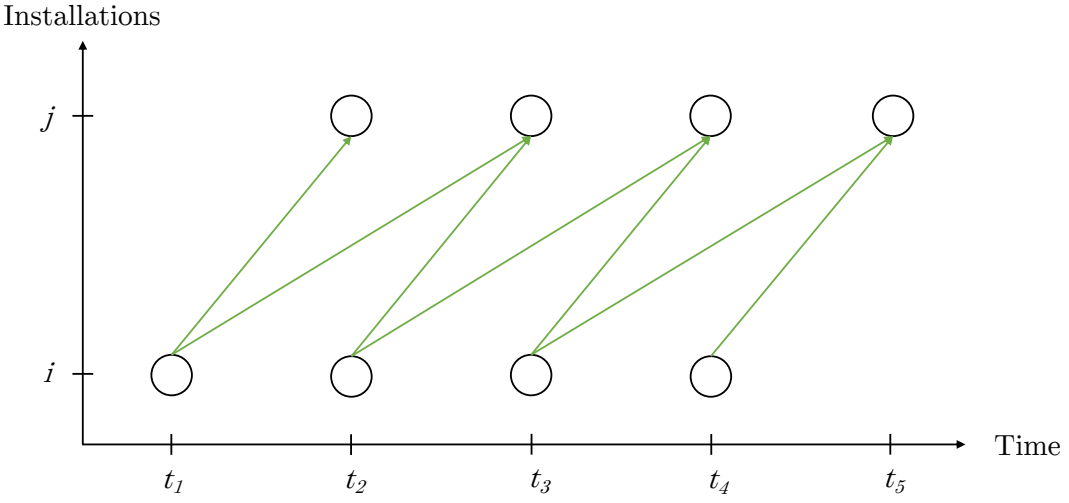


Figure 4.2: Arcs between nodes when nodes represent installations at certain time points. Note that the nodes will be defined differently from Section 4.2.2.

Each arc yields a set of times that a vessel can depart from installation i and arrive at installation j. This set of times is equivalent to a chosen average sailing speed between the installations. For example, if the vessel were to leave installation i at t_1, it would

have two speed alternatives: (1) the speed that would yield arrival time t_2, and (2) the speed that would yield arrival time t_3. Given the distance between the installations, the speed can be calculated. This way, the arcs represent different speed alternatives that the mathematical model of the TDVRPSO can choose from when minimizing costs. This is how speed optimization is handled in this thesis.

Another result of discretizing time as described above is that the non-linear fuel consumption calculations can be performed outside the mathematical model. The calculations are done in the arc-generation procedure that generates arcs and assigns arc costs, described in Section 4.2.3. The weather dependency can be handled because the set of times defined by an arc yield the weather conditions that the vessel will be operating in. These weather conditions may be modeled in several ways. In this thesis, they are modeled using weather states for intervals of significant wave heights, inspired by the work of Halvorsen-Weare and Fagerholt (2011).

4.2.2 Definitions of Nodes and Arcs

In the above discussion, it is temporarily assumed that each node represents an installation being visited by a vessel at a certain point in time. As the TDVRPSO considers mandatory delivery orders, optional delivery orders, and optional pickup orders, each installation may place up to three different order types. The actual approach taken in this thesis is to define a node for the servicing of an order, the depot at the beginning of the planning horizon, and the depot at the end of each voyage. The names and descriptions of the nodes in the problem are listed below.

1. Order node: An order from an installation being visited by a vessel at a certain time point, broken down into the three order types in the TDVRPSO.
   (a) Mandatory delivery (MD) order node
   (b) Optional delivery (OD) order node
   (c) Optional pickup (OP) order node
2. Start depot node: The supply depot housing a vessel at the beginning of the planning horizon, specifically at the end of vessel preparation.
3. End depot node: The supply depot housing a vessel at the end of its voyage.

Arcs link the nodes. An arc is defined by a start and end node and the vessel that is visiting the nodes. The start and end times of an arc are given by the time points of the start and end nodes. It is important to note that no time is spent by a vessel in a node, and the time consumed by vessel activities, i.e., sailing, idling, and servicing orders, is completely contained in the arcs. Hence, the start time of an arc defines the discrete time

point the vessel leaves the start node, while the end time defines the discrete time point all required vessel activities between the start and end nodes are finished. There are three possible combinations of vessel activities that can be integrated into an arc, depending on the types of the start and end nodes. Figures 4.3 to 4.5 visualizes these combinations. The blue circles represent the start and end nodes, while the black circles are checkpoints between one activity and the next. The black arrows are vessel activities, and the green arrows represent arcs.

Figure 4.3 shows the first combination of vessel activities that are integrated into an arc, namely sailing, idling, and servicing. In this case, the start node is either an order node or the *start* depot node, while the end node is an order node. If both nodes are order nodes, their orders must be from different installations. Otherwise, the sailing time would be zero. Note that the dashed line represents idling. This represents the fact that idling will only be performed if the vessel arrives at a closed installation.

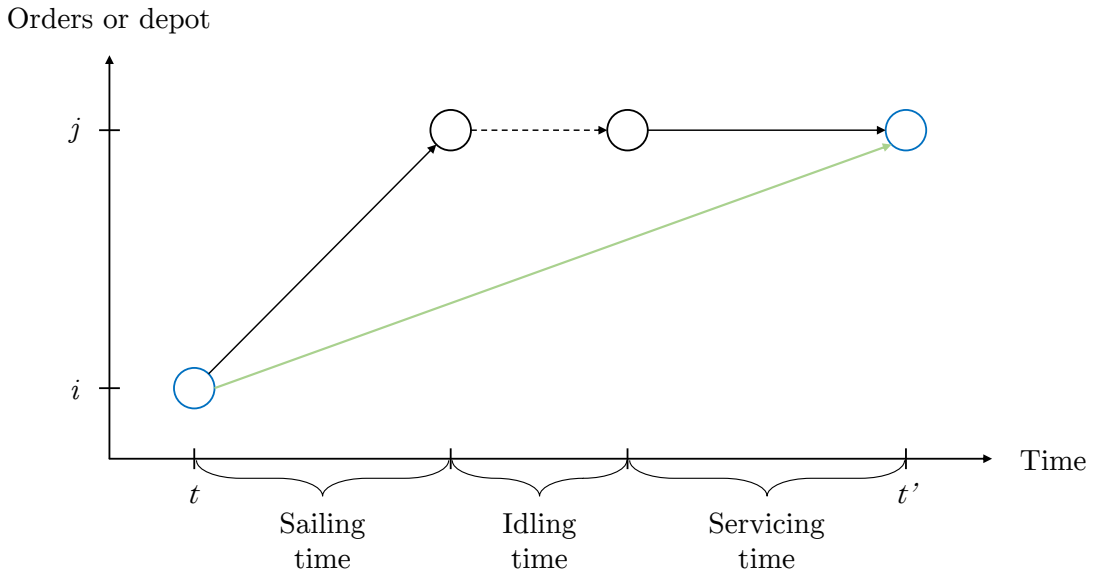


Figure 4.3: Arc when node $(i,\ t)$ is the start depot node and node $(j,\ t')$ is an order node, or node $(i,\ t)$ and $(j,\ t')$ are order nodes with orders from different installations. The arc will certainly include sailing and servicing, and potentially idling.

If the start and end nodes are both order nodes with orders from the same installation, no sailing or idling will be performed. There is no need for sailing, as the vessel is already at the installation, and as we assume that servicing must be performed continuously once it has started, no idling can be performed. Figure 4.4 outlines this case.

Finally, if the start node is an order node and the end node is the *end* depot node, only sailing will be performed. This is because the end depot node has no order demand, so idling and servicing will not be performed. This is visualized by Figure 4.5.

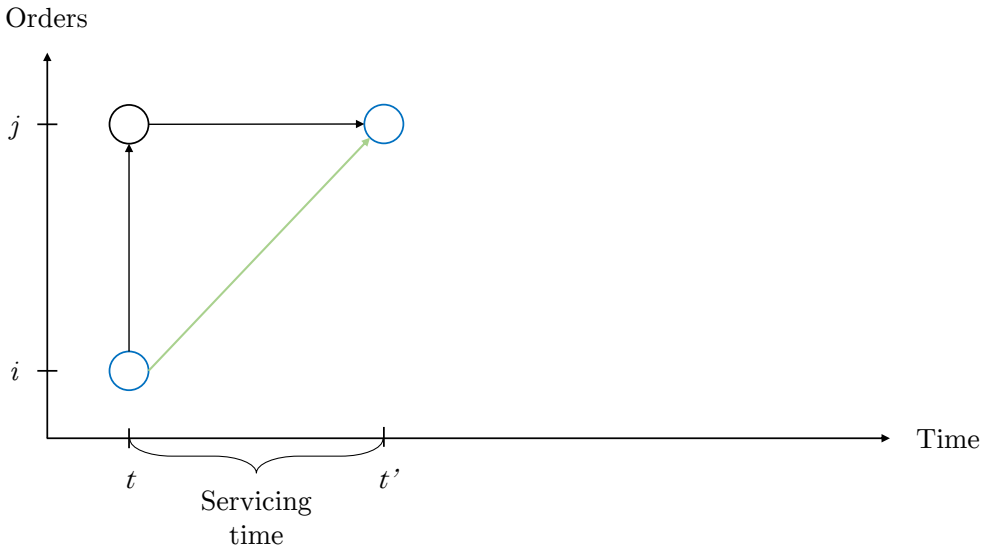


Figure 4.4: Arc when the orders in node (i, t) and (j, t') are from the same installation. The arc will only include servicing, as the distance between the orders is zero.

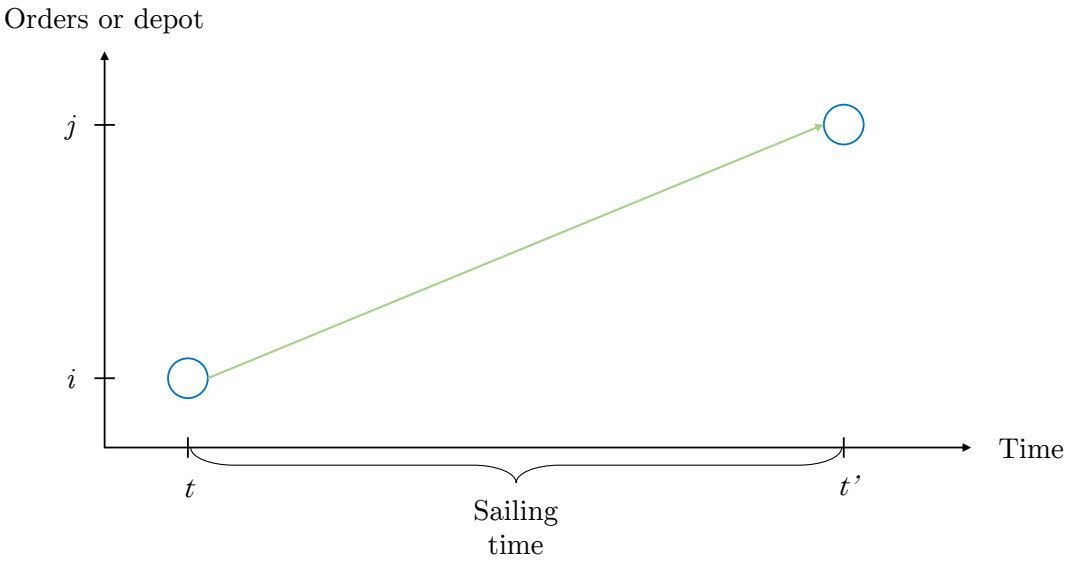


Figure 4.5: Arc when node (i, t) is an order node and node (j, t') is an end depot node. The arc will only include sailing, as the depot has no orders to be serviced.

The above discussion does not mention the vessel preparation at the supply depot as a time-consuming vessel activity. This is because the start and end times for vessel preparation are fixed and equal for all PSVs. Hence, the contribution from vessel preparation to arcs leaving a start depot node would be equal for all these arcs. From the perspective of the model choosing between these arcs, this contribution would not separate them from each other. Furthermore, as we assume that vessels leave the supply depot immediately after vessel preparation, all arcs leaving the start depot node have their start time set to the end time of vessel preparation. Note that this could easily be changed in the model if the vessels should leave at different time points.

For the model to be able to choose cost-minimizing arcs, the arcs have associated arc costs. The arc costs are calculated as the fuel cost of all fuel consumed by the vessel activities in

the arc, plus potential chartering costs. Note that the calculation of fuel consumption and cost happens outside the mathematical model, allowing these calculations to be non-linear and account for the weather dependency in the problem, as explained in Section 4.2.1. The fuel consumption calculations performed in this thesis are described in Chapter 7.

4.2.3 Arc-Generation Procedure

The arc-generation procedure generates all feasible arcs each available vessel may sail in the planning horizon. The vessels that receive arcs include the available vessels from the contracted fleet and the vessel that can be chartered from the spot market. All feasible arcs for all vessels are then provided as input to the mathematical model of the TDVRPSO performing speed optimization and order selection to choose the cheapest combination of feasible arcs. The implementation of the arc-generation procedure is described in Algorithm 1. The algorithm generates one or more arcs between start or end depot nodes and order nodes, and between order nodes.

Lines 2 to 14 control how the orders and the depot are combined with the discrete time points in the planning horizon to form arcs. Lines 3 and 4 suggest combinations of orders and the depot that can be used as start and end locations, while Line 8 suggests possible start times for these combinations. The run time of the mathematical model of the TDVRPSO increases drastically with the number of arcs generated, as each arc corresponds to a variable in the model. To limit the number of generated arcs, the if-clause in Lines 9 to 14 is added. The first if-statement ensures that the earliest start time of an arc will be the first time point a vessel can arrive at the installation with order i. As long as i is not the supply depot, the earliest start times t^s close to the preparation end time will not be feasible start times, so these start times are avoided. The second if-statement ensures that the depot can always be reached within the return time if the vessel were to sail at maximum speed from the installation with order j starting at t^s. Arcs with start times later than this time point should not be generated as they will lead to infeasible voyages. The third if-statement ensures that all start times are time points where the installation with order i is open. No arcs should have start times at time points where the installation is closed, either due to opening hours or bad weather conditions.

If an installation has more than one of the three order types, there will be more than one order representing the same installation. Figure 4.6 illustrates how arcs between nodes representing these orders could be generated when an installation has orders of all three order types. The green and blue arcs are inbound and outbound from and to nodes from different installations, respectively. The red arcs connect the order nodes from the same installation. If arcs are generated this way, the mathematical model of the TDVRPSO will form symmetric solutions. An example of two symmetric solutions is choosing (1) the inbound arc to OD, OD to MD, and MD to OP, or (2) the inbound arc to MD, MD to OD, and OD to OP. Both solutions represent servicing all orders at the installation.

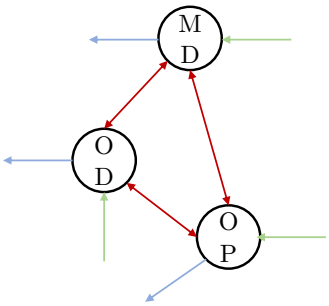


Figure 4.6: Arc-generation pattern resulting in symmetric solutions.

To avoid these symmetries, we make sure that arcs are generated according to predefined patterns. Figure 4.7 shows all possible combinations of an installation's order nodes, referred to as a *cluster*, when it places more than one order type. To generate these patterns, we make three considerations. Firstly, we restrict the number of arcs entering a cluster. If an installation has a mandatory order, this node will be the only entry point as it must be serviced. Otherwise, the entry points are the optional delivery order node and the optional pickup order node. Both optional order nodes must have arcs entering so that one can be reached without visiting the other. Secondly, orders are always visited in the order MD $\rightarrow$ OD $\rightarrow$ OP within a cluster. This ensures that delivery is done before pickup at an installation, thus freeing up the given vessel's available capacity for collecting pickup orders. There is also an arc going directly from MD to OP to allow postponing the OD order but servicing the OP order. Thirdly, all orders must have an outbound arc exiting the cluster to make the optional orders optional. By ensuring that only arcs following these patterns are generated, we avoid symmetric solutions in the mathematical model without restricting the solution space. The if-clause in Lines 5 and 6 ensures that these patterns are followed when choosing order i and j.

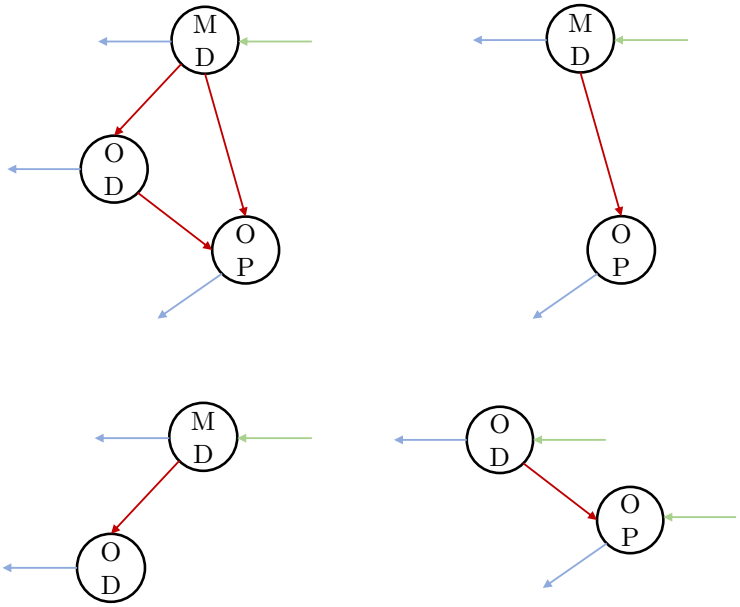


Figure 4.7: Arc-generation patterns resulting in no symmetric solutions.

When a valid combination of orders i and j and start time t^s for vessel v has been found after Lines 2 to 14, Lines 16 to 32 calculate possible end times and generates arcs for each of the end times. Starting in Line 16 with the start time t^s, the algorithm calculates the possible arrival times resulting from different speed alternatives within the speed limits. From this point, the rest of the algorithm is separated into two parts, the first calculating one or many end times if j is an order, the second calculating one end time if j is the depot.

If j is an order, Lines 18 to 22 calculate possible servicing start times. With the servicing duration required by order j found in Line 18, Line 19 checks whether continuous servicing can be performed for each of the possible arrival times. If this is possible, the arrival time is added to the set of servicing start times. Now, this set will contain servicing start times identical to arrival times, implying that no idling will be performed in the arcs with these time points. However, it could be the case that none of the arrival times lead to servicing start times that yield continuous servicing. In other words, the servicing of order j cannot be performed without idling. The if-statement in Lines 20 to 22 handles this case. If order j cannot be serviced without idling, the first possible servicing start time that yields continuous servicing is calculated. Then, this time point is added to the set of servicing start times.

With at least one servicing start time in the set of servicing start times, Lines 24 to 27 find possible arc end times, calculate arc costs, and update the set of arcs. First, an arc end time t' is calculated for each servicing start time by adding the required servicing duration for order j to the servicing start time. Hence, the number of arcs that are generated for a specific vessel v, order or depot i, order j, and start time t^s will be equal to the number of servicing start times. Second, the arc cost given the start, arrival, servicing, and end time points is calculated. The arc cost is the sum of the sailing, idling, servicing, and, potentially, chartering costs. Third, the arc with the arc cost is added to the set of arcs.

Note that in the case where a vessel is idling between the arrival time and the servicing start time, only one arc is added. As all arrival times would lead to idling in this case, the servicing start time is equal for all arrival times. Consequently, the arc from node $(i,\ t)$ to $(j,\ t')$ can only have one end time, as the servicing duration is fixed. Multiple arcs with different arc costs could be generated between these nodes, but since they would have the same end time, the choice among these would not influence the rest of the voyage. Therefore, it is clear that the model would choose the cheapest one. To simplify the model's calculations, only the arc with the lowest cost is added to the set of arcs.

If j is the depot, the else-clause in Lines 30 to 32 make sure that the arc with the cheapest arc cost is added to the set of arcs. Similar to the argument made in the previous paragraph, the choice of arcs between an order node and the end depot node does not affect the voyages, only their costs. Hence, we know that the model would choose the cheapest one, so only the arc with the cheapest arc cost is added.

Algorithm 1: Arc-Generation Procedure

Input : Set of available vessels, set of orders, start depot, end depot, vessel preparation end time, vessel return time

Output: Set of arcs

```
1  initialize the set of arcs as an empty set
2  for v in the set of all available vessels do
3      for i in the set of all orders and the depot do
4          for j in the set of all orders and the depot do
5              if moving from i to j is not allowed then
6                  skip to next j
7              end
8              for t^s in the set of time points from preparation end to return time do
9                  if i is not the depot and t^s < earliest arrival time at i from depot then
10                     skip to next t^s
11                 else if t^s+ min sailing duration from j to depot > return time then
12                     skip to next t^s
13                 else if installation with order i is closed at t^s then
14                     skip to next t^s
15                 end
16                 calculate the set of possible arrival times at order or depot j
17                 if j is an order then
18                     calculate the servicing duration required for j
19                     calculate the servicing start times with no idling before start
20                     if servicing of order j cannot be performed without idling then
21                         calculate the first possible servicing start time
22                         add the time point to the set of servicing start times
23                     end
24                     for t^ss in the set service start times do
25                         calculate end time t' by adding service duration to t^ss
26                         calculate the arc cost given the start, arrival, servicing, and end
                            times
27                         add the arc (i, t), (j, t') and its arc cost to the set of arcs
28                     end
29                 else
30                     calculate the arc costs for all arrival times
31                     set the end time t' equal to the arrival time with the cheapest cost
32                     add the arc (i, t), (j, t') and its arc cost to the set of arcs
33                 end
34             end
35         end
36     end
37 end
```

The arc-generation procedure handles some of the restrictions in the TDVRPSO, allowing the mathematical model to omit four constraints that otherwise would need to be formulated. These constraints are listed below.

1. An installation can only be visited once in the planning horizon.
2. Each vessel must return to the supply depot within its specified return time.
3. The sailing speeds must be within the provided upper and lower limits.
4. An installation must be visited within its opening hours.

The first constraint is handled by generating the arcs for installations with more than one order type to avoid symmetries as described above, together with the visitation constraints that will be presented in Section 4.3. By only having one arc that can enter a cluster of order nodes representing the same installation and restraining the number of visits to a node to one, an installation will be visited at most once in the planning horizon. The second constraint is handled by excluding arcs entering the end depot node with an end time later than the vessel return time. The third constraint is handled by only calculating arrival times resulting from speeds within the speed limits. The fourth constraint is handled by checking that an installation is open when servicing is performed. If it is not open, servicing cannot be performed. To keep the mathematical model identical to the implementation, these constraints are not part of the mathematical formulation presented in Section 4.3.

Finally, note that an arc between the start depot node and the end depot node is provided. The arc has no associated cost. Choosing this arc for a vessel is equivalent to not using the vessel.

4.3 Mathematical Formulation

This section presents the mathematical formulation of the arc-flow model of the TDVRPSO, with sets and indices, parameters, decision variables, objective function, and constraints.

4.3.1 Notation

Sets and Indices

All sets provided are zero-indexed.

$\mathcal{V}$	-	set of available vessels v
$\mathcal{N}$	-	set of orders i
$\mathcal{N}^{MD}$	-	subset of $\mathcal{N}$ consisting of mandatory delivery orders
$\mathcal{N}^{OD}$	-	subset of $\mathcal{N}$ consisting of optional delivery orders
$\mathcal{N}^{OP}$	-	subset of $\mathcal{N}$ consisting of optional pickup orders
$\mathcal{A}_v$	-	set of arcs $((i,\ t),\ (j,\ t'))$ for vessel v
$\mathcal{T}$	-	set of all discrete time points t in the planning horizon
$\mathcal{T}^{S}_{ijv}$	-	subset of $\mathcal{T}$ with start times for arcs between i and j for vessel v
$\mathcal{T}^{SS}_{ijtv}$	-	subset of $\mathcal{T}$ with specific start times for arcs between i and j with end time t for vessel v
$\mathcal{T}^{SE}_{itjv}$	-	subset of $\mathcal{T}$ with specific end times for arcs between i and j with start time t for vessel v

Figures 4.8 to 4.10 provide an illustrative view of the time subsets defined above. The figures connect nodes with arcs, demonstrating which discrete time points are included in the given subsets. The red nodes define the time points in the different sets. The black nodes are the nodes at the other end of the arc, and the grey nodes and arcs are outside the definition of the set in question. The blue-shaded rectangles represent time points with no arcs, as no nodes containing order i exist at t_4 and t_5. This could be due to the installation with order i being closed at these times.

In Figure 4.8, the set $\{t_1,\ t_2,\ t_3,\ t_6,\ t_7,\ t_8\}$ is an example of the set with the start times for arcs between nodes containing orders i and j for vessel v, $\mathcal{T}^{S}_{ijv}$. The time points in the set are given by the red *start* nodes in the figure. Hence, this set contains all start times for the arcs with start nodes containing order i and end nodes containing order j for a specific vessel v.

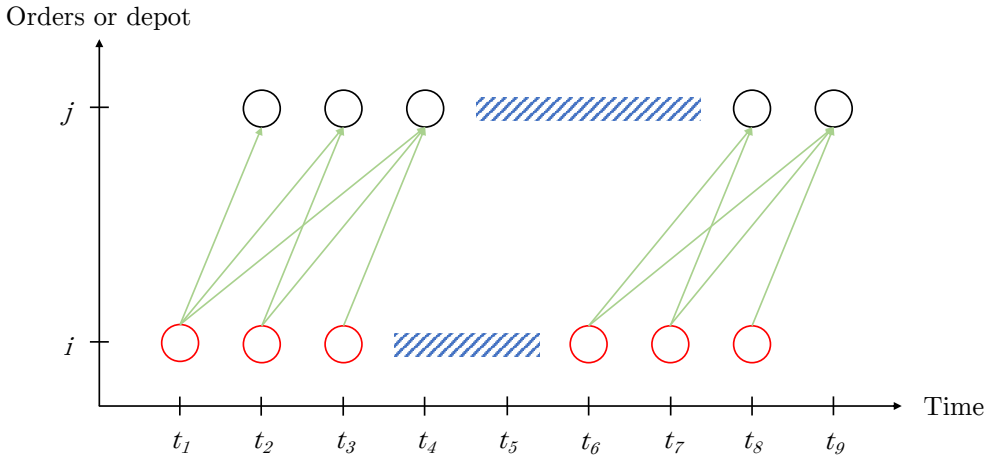


Figure 4.8: Visualization of T^S_{ijv}. This set contains all start times for arcs between nodes containing orders i and j for vessel v.

In Figure 4.9, the set $\{t_6,\ t_7,\ t_8\}$ is an example of the set with the specific start times for arcs between nodes containing orders i and j ending at time t_9 for vessel v, $\mathcal{T}^{SS}_{ijt_9v}$. As the time points in the set are given by the red *start* nodes in the figure, it is seen that these are the start times of all arcs between nodes containing orders i and j and ending at time t_9 for vessel v.

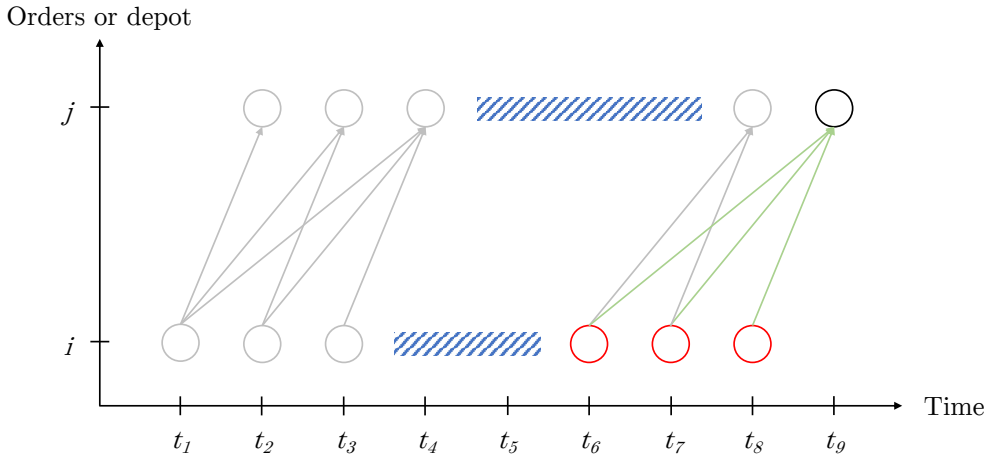


Figure 4.9: Visualization of $T^{SS}_{ijt_9v}$. This set contains all start times for arcs between nodes containing orders i and j with end time t_9 for vessel v.

In Figure 4.10, the set $\{t_2,\ t_3,\ t_4\}$ is an example of the set with specific end times for arcs between nodes containing orders i and j starting at time t_1 for vessel v, $\mathcal{T}^{SE}_{it_1jv}$. The time points in the set are given by the red *end* nodes in the figure. Consequently, these are the end times of all arcs between nodes containing orders i and j and starting at time t_1 for vessel v.

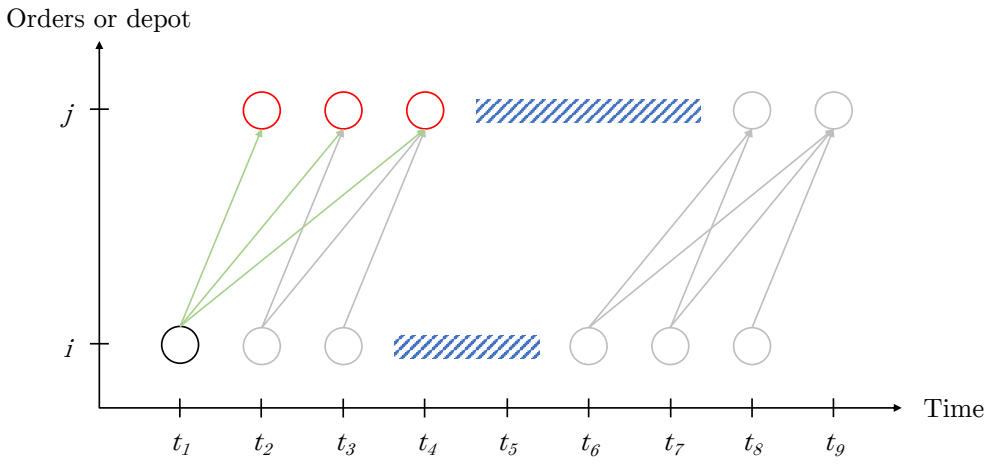


Figure 4.10: Visualization of $T^{SE}_{it_1jv}$. This set contains all end times for arcs between nodes containing orders i and j with start time t_1 for vessel v.

Parameters

S_i	-	size of order i
Q_v	-	maximum load capacity of vessel v
$C^A_{itjt'v}$		fuel consumption and potential chartering cost for vessel v on arc $((i,\ t),\ (j,\ t'))$
C^P_i	-	penalty cost of not servicing optional order i
o	-	supply depot at the beginning of a voyage, modeled as an origin node
d	-	supply depot at the end of a voyage, modeled as a destination node
t^*	-	time at which vessel preparation ends

Decision Variables

$x_{itjt'v}$	-	$\begin{cases} 1, & \text{if arc } \{(i,t),(j,t')\} \text{ is used by vessel } v \\ 0, & \text{otherwise} \end{cases}$
u_{iv}	-	$\begin{cases} 1, & \text{if order } i \text{ is serviced by vessel } v \\ 0, & \text{otherwise} \end{cases}$
l^D_{iv}	-	delivery load for vessel v after servicing order i
l^P_{iv}	-	pickup load for vessel v after servicing order i

4.3.2 Arc-Flow Model

Objective Function

The objective function (4.1) in the arc-flow model minimizes the sum of the monetary costs from fuel consumption and chartering of the spot vessel, as well as penalty costs assigned to optional delivery or optional pickup orders that are postponed.

$$\min \sum_{v \in \mathcal{V}} \sum_{\{(i,t),(j,t')\} \in \mathcal{A}_v} C^A_{itjt'v} x_{itjt'v} + \sum_{i \in \mathcal{N}^{OD} \cup \mathcal{N}^{OP}} C^P_i (1 - \sum_{v \in \mathcal{V}} u_{iv}) \tag{4.1}$$

Constraints

Constraints (4.2) ensure flow conservation, i.e., that a node having an arc entering it also has an arc leaving it. In practice, this means that when a vessel has finished servicing an order at a node, it is required to leave that node at that exact time. Constraints (4.3) and (4.4) ensure that there is exactly one arc leaving the start depot node and one arc entering the end depot node for each vessel.

$$\sum_{j \in \mathcal{N}} \sum_{t' \in \mathcal{T}^{SS}_{ijtv}} x_{jt'itv} - \sum_{j \in \mathcal{N}} \sum_{t' \in \mathcal{T}^{SE}_{itjv}} x_{itjt'v} = 0, \quad i \in \mathcal{N}, t \in \mathcal{T}, v \in \mathcal{V} \tag{4.2}$$

$$\sum_{j \in \mathcal{N}} \sum_{t' \in \mathcal{T}^{SE}_{otjv}} x_{ot^*jt'v} = 1, \quad v \in \mathcal{V} \tag{4.3}$$

$$\sum_{i \in \mathcal{N}} \sum_{t \in \mathcal{T}^{S}_{ijv}} \sum_{t' \in \mathcal{T}^{SE}_{itdv}} x_{itdt'v} = 1, \quad v \in \mathcal{V} \tag{4.4}$$

Constraints (4.5) to (4.8) handle servicing of orders. Constraints (4.5) ensure that all mandatory delivery orders are serviced exactly once, while constraints (4.6) ensure that all optional delivery and pickup orders are serviced at most once during the planning horizon. Constraints (4.7) ensure that an order is serviced equally many times as it is visited. Constraints (4.8) ensure that a vessel must visit an order if it is to service that order. From the binary constraints (4.17) on u_{jv}, these constraints further ensure that a vessel may only service the same order once.

$$\sum_{v \in \mathcal{V}} u_{iv} = 1, \quad i \in \mathcal{N}^{MD} \tag{4.5}$$

$$\sum_{v\in\mathcal{V}} u_{iv} \leq 1, \quad i \in \mathcal{N}^{OD} \cup \mathcal{N}^{OP} \tag{4.6}$$

$$\sum_{i\in\mathcal{N}} \sum_{t\in\mathcal{T}^S_{ijv}} \sum_{t'\in\mathcal{T}^{SE}_{itjv}} \sum_{v\in\mathcal{V}} x_{itjt'v} = \sum_{v\in\mathcal{V}} u_{jv}, \quad j \in \mathcal{N} \tag{4.7}$$

$$\sum_{i\in\mathcal{N}} \sum_{t\in\mathcal{T}^S_{ijv}} \sum_{t'\in\mathcal{T}^{SE}_{itjv}} x_{itjt'v} = u_{jv}, \quad j \in \mathcal{N}, v \in \mathcal{V} \tag{4.8}$$

Constraints (4.9) to (4.15) handle the load and capacity of the PSVs. Constraints (4.9) set the delivery load of a given cargo type for all vessels equal to the sum of all order sizes associated with the delivery orders to be serviced. Constraints (4.10) ensure that the sum of the loads for both delivery and pickup of a given cargo type is contained within the capacity for that cargo type in a given vessel. Constraints (4.11) and (4.12) control the load continuity of delivery loads between installations i and j for all vessels, while Constraints (4.13) and (4.14) do the same for pickup loads. Constraints (4.15) set the pickup load for all vessels equal to the sum of all order sizes associated with the pickup orders to be serviced.

$$l^D_{ov} = \sum_{i\in\mathcal{N}^{MD}\cup\mathcal{N}^{OD}} S_i u_{iv}, \quad v \in \mathcal{V} \tag{4.9}$$

$$l^D_{iv} + l^P_{iv} \leq Q_v u_{iv}, \quad i \in \mathcal{N}, v \in \mathcal{V} \tag{4.10}$$

$$l^D_{jv} \leq l^D_{iv} - S_j u_{jv} + Q_v \left(1 - \sum_{t\in\mathcal{T}^S_{ijv}} \sum_{t'\in\mathcal{T}^{SE}_{itjv}} x_{itjt'v}\right), \quad i \in \mathcal{N}, j \in \mathcal{N}^{MD}\cup\mathcal{N}^{OD}, v \in \mathcal{V} \tag{4.11}$$

$$l^D_{jv} \leq l^D_{iv} + Q_v \left(1 - \sum_{t\in\mathcal{T}^S_{ijv}} \sum_{t'\in\mathcal{T}^{SE}_{itjv}} x_{itjt'v}\right), \quad i \in \mathcal{N}, j \in \mathcal{N}^{OP}, v \in \mathcal{V} \tag{4.12}$$

$$l^P_{jv} \geq l^P_{iv} + S_j u_{jv} - Q_v \left(1 - \sum_{t\in\mathcal{T}^S_{ijv}} \sum_{t'\in\mathcal{T}^{SE}_{itjv}} x_{itjt'v}\right), \quad i \in \mathcal{N}, j \in \mathcal{N}^{OP}, v \in \mathcal{V} \tag{4.13}$$

$$l_{jv}^{P} \geq l_{iv}^{P} - Q_v \left(1 - \sum_{t \in \mathcal{T}_{ijv}^{S}} \sum_{t' \in \mathcal{T}_{itjv}^{SE}} x_{itjt'v} \right), \quad i \in \mathcal{N}, j \in \mathcal{N}^{MD} \cup \mathcal{N}^{OD}, v \in \mathcal{V} \tag{4.14}$$

$$l_{dv}^{P} = \sum_{i \in \mathcal{N}^{OP}} S_i u_{iv}, \quad v \in \mathcal{V} \tag{4.15}$$

Constraints (4.16) to (4.19) specify the domains of the four decision variables. Constraints (4.16) and (4.17) put binary requirements on whether an arc is used or not and whether an order is serviced or not, respectively. Constraints (4.18) and (4.19) impose non-negativity constraints on the loads of delivery and pickup, respectively.

$$x_{itjt'v} \in \{0,1\}, \quad \{(i,t),(j,t')\} \in \mathcal{A}_v, v \in \mathcal{V} \tag{4.16}$$

$$u_{iv} \in \{0,1\}, \quad i \in \mathcal{N} \cup \{o,d\}, v \in \mathcal{V} \tag{4.17}$$

$$l_{iv}^{D} \geq 0, \quad i \in \mathcal{N} \cup \{o,d\}, v \in \mathcal{V} \tag{4.18}$$

$$l_{iv}^{P} \geq 0, \quad i \in \mathcal{N} \cup \{o,d\}, V \in \mathcal{V} \tag{4.19}$$

Chapter 5

The Supply Vessel Speed Optimization Problem

The *Supply Vessel Speed Optimization Problem* (SVSOP) is a subproblem to the Adaptive Large Neighborhood Search (ALNS) heuristic presented in Chapter 6. The problem entails the assignment of weather-adjusted sailing speeds for a specific vessel on sailing legs along a predetermined voyage and determining idling and servicing times associated with the orders serviced. Providing sailing, idling, and servicing times along a voyage, the SVSOP aims to provide an accurate estimation of the costs associated with a voyage, and consequently, the quality of the voyages provided by the ALNS heuristic.

As discussed when outlining the arc-flow formulation in Chapter 4, the weather conditions during which a vessel operates vary with time and add an operational aspect to the TDVRPSO. In order to handle the non-linearities introduced by the weather dependency and speed optimization, time is made discrete in the same manner as in Chapter 4.

Section 5.1 explains how the SVSOP can be formulated as a *Shortest Path Problem* (SPP), while Section 5.2 outlines the mathematical formulation of the considered problem.

5.1 The SVSOP as a Shortest Path Problem

Assigning sailing speeds on a voyage is equivalent to assigning start and end times to the distances sailed. When discretizing time, both the start and end times of feasible arcs are known, and consequently, the weather forecast for the arc. As the weather forecast at a given time along the arc is provided, sailing speeds and servicing times at installations may be adjusted to the weather conditions. Based on the feasible start times, feasible arrival times resulting from feasible speed alternatives may be calculated. Further, once arrival times are known, idling and servicing times may be calculated according to the procedure

described in Section 4.2.3. Note that according to this procedure, the departure time is always equal to the start time of an arc.

The subproblem may be formulated as a *Shortest Path Problem* (SPP) by generating all feasible arcs. As the sequences of orders are provided to the problem, the resulting network of arcs and nodes, which like for the arc-flow formulation in Chapter 4 represent orders at specific points in time, form a *Directed Acyclic Graph* (DAG). In the network, each arc has a time-dependent cost associated with it. These costs can be viewed as distances between nodes, in which case the problem is indeed an SPP. Thus, the network may be structured as exemplified in Figure 5.1, where each structural level represents servicing an order or returning to the depot. Each level will generate a variety of nodes representing different time points and consequently different associated costs. The notation used in Figure 5.1 is presented in Section 5.2.

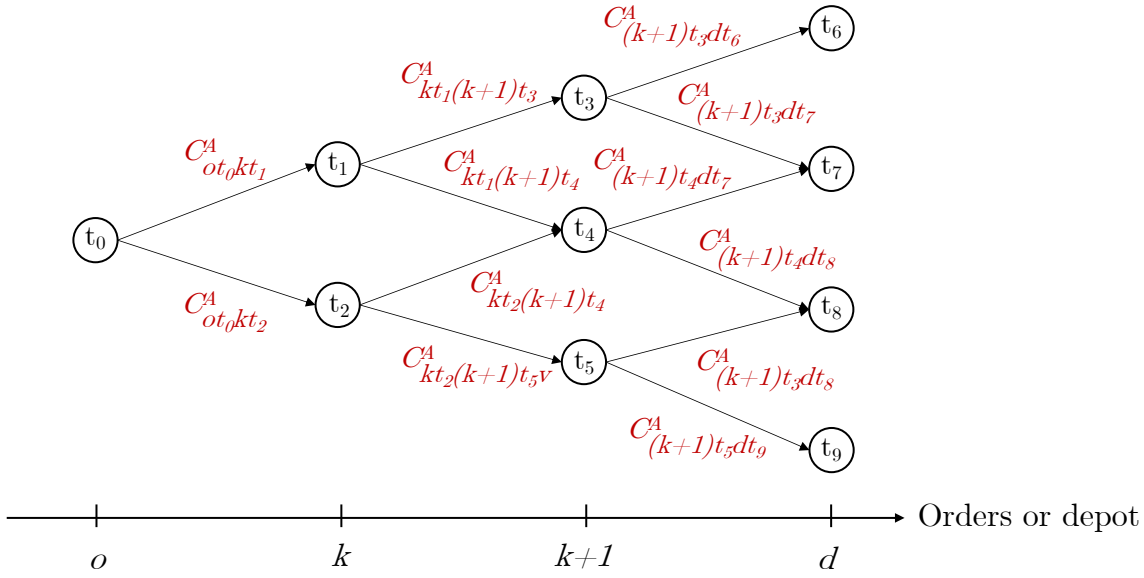


Figure 5.1: Visualization of a *Directed Acyclic Graph* for a network where two orders k and $k+1$ are serviced from the start depot (o), ending at the end depot (d). Each node in the network has two outgoing arcs, meaning that the vessel in question may choose two speed alternatives. The cost of the arcs are indicated in red using notation introduced in Section 5.2.

5.2 Mathematical Formulation

Based on the mathematical formulation in Section 4.3.1 and Section 4.3.2, this section presents a shortest path formulation of the SVSOP. Note that constraints handling order selection and of load are all taken care of in the ALNS presented in Chapter 6, and that the generation of feasible arcs between nodes is done using the same arc-generation procedure detailed in Section 4.2.3. The order in which the orders on a voyage are serviced is also given. The problem is solved for each vessel.

Sets and Indices

All sets provided are zero-indexed.

$\mathcal{N}$	-	set of orders serviced on the predetermined voyage, indexed by $k = 1, ..., N-1$, or the start ($k = 0$) or end ($k = N$) depot
$\mathcal{A}$	-	set of arcs $((k,\ t),\ ((k+1),\ t'))$
$\mathcal{T}$	-	set of all discrete time points t in the planning horizon
$\mathcal{T}_k^S$	-	subset of $\mathcal{T}$ with start times for arcs from order k
$\mathcal{T}_{kt}^{SS}$	-	subset of $\mathcal{T}$ with specific start times for arcs from order k with end time t at order $k+1$
$\mathcal{T}_{tk}^{SE}$	-	subset of $\mathcal{T}$ with specific end times for arcs from order k with start time t at order $k-1$

Parameters

$C^A_{kt(k+1)t'}$	-	fuel consumption and potential chartering cost on arc $((k,\ t),\ ((k+1),\ t'))$
t^*	-	time at which vessel preparation ends and the vessel starts sailing

Decision Variables

$x_{kt(k+1)t'}$ - $\begin{cases} 1, & \text{if arc } \{(k,t),((k+1),t')\} \text{ is used} \\ 0, & \text{otherwise} \end{cases}$

Objective Function

The objective function (5.1) in the shortest path formulation minimizes the sum of the monetary costs from fuel consumption and chartering of spot vessels when completing a voyage. Note that since order selection is handled in the ALNS heuristic presented in Chapter 6, penalty costs for postponing orders are not considered.

$$\min \sum_{\{(k,t),((k+1),t')\} \in \mathcal{A}} C^A_{kt(k+1)t'} x_{kt(k+1)t'} \tag{5.1}$$

Constraints

Equivalently to in Section 4.3.2, Constraints (5.2) ensure flow conservation, i.e. that a node having an arc entering it also has an arc leaving it. In practice, when a vessel has finished servicing an order at a node, it is required to leave that node at that exact time. Constraints (5.3) and (5.4) ensure that there is exactly one arc leaving the start depot node and one arc entering the end depot node for the given vessel. Consequently, we ensure that all orders to be serviced are on the shortest path and that the path both starts and ends at the specified depot.

$$\sum_{t' \in \mathcal{T}^{SS}_{kt}} x_{(k+1)t'kt} - \sum_{t' \in \mathcal{T}^{SE}_{t(k+1)}} x_{kt(k+1)t'} = 0, \quad k = 1, ..., N-1, \quad t \in \mathcal{T} \tag{5.2}$$

$$\sum_{t' \in \mathcal{T}^{SE}_{t0}} x_{0t^*1t'} = 1 \tag{5.3}$$

$$\sum_{t \in \mathcal{T}^{S}_{(N-1)}} \sum_{t' \in \mathcal{T}^{SE}_{tN}} x_{(N-1)tNt'} = 1 \tag{5.4}$$

Constraints (5.5) specify the domains of the decision variables, putting binary requirements on the use of an arc.

$$x_{kt(k+1)t'} \in \{0, 1\}, \quad k \in \mathcal{K} \setminus N, \quad \{(k, t), (k+1, t')\} \in \mathcal{A} \tag{5.5}$$

Based on this mathematical formulation, the SPP can be solved using Djikstra's algorithm (Dijkstra et al., 1959) or a simple tree search.

Chapter 6

Adaptive Large Neighborhood Search

This chapter presents an Adaptive Large Neighborhood Search (ALNS) for solving the Time-Dependent Vessel Routing Problem with Speed Optimization (TDVRPSO). The heuristic detailed is based on the work of Ropke and Pisinger (2006) who address the pickup and delivery problem with time windows using an ALNS. Extending their work, the solution method presented in this chapter adds both destroy and repair heuristics considered relevant for the TDVRPSO. Further extensions include performing a local search on the solutions found by the ALNS heuristic and solving a set partitioning problem for combining voyages. This chapter explains the solution method in its entirety.

Section 6.1 outlines the ALNS for the TDVRPSO, while Section 6.2 explains how a solution to the problem is represented. Section 6.3 discusses how we ensure feasibility in the solutions produced by the ALNS, and Section 6.4 introduces the heuristic for creating an initial feasible solution to the problem. Section 6.5 describes the various destroy and repair heuristics used in the large neighborhood search, while Section 6.6 explains what influences the choice of these heuristics in each iteration of the ALNS. Section 6.7 details the local search and how this is integrated with the ALNS heuristic. Finally, Section 6.8 presents the set partitioning problem, called the Voyage Combination Problem (VCP), used to generate new solutions from combining previously obtained voyages in the ALNS.

6.1 Overview of the ALNS

The ALNS procedure is presented in Algorithm 2. The algorithm initializes the current solution x by constructing a feasible solution as described in Section 6.4. Successively, a destroy and a repair heuristic are selected as presented in Section 6.6 and used to generate a candidate solution x' from x. If the candidate solution is promising, a local search attempting to find improving solutions close to the candidate solution is performed (Section 6.7). Provided that x' is accepted when applying the simulated annealing-based acceptance criterion described in Section 6.5.4, the current solution x is set to be x'.

Moreover, the VCP presented in Section 6.8 is solved for every I^{VCP} iterations. The VCP evaluates voyages saved up until this point and generates a new candidate solution, x''. This solution is accepted as the current solution by the same simulated annealing-based acceptance criterion as for evaluating x'. For each iteration in the ALNS, if the current solution is better than the global best solution found, x^*, x and x^* are set to be the new best solution. Furthermore, the total number of ALNS iterations is divided into *segments* of I^S iterations. Each time I^S iterations have been completed, the weights of the destroy and repair heuristics, as well as the cooling temperature used in the simulated annealing-based acceptance criterion, are updated. The ALNS terminates after a maximum of I^{ALNS} iterations, returning the global best solution found. The algorithm is presented below.

Algorithm 2: ALNS

Input : Total number of ALNS iterations (I^{ALNS}), number of segment iterations (I^S), number of iterations after which the VCP is solved (I^{VCP})

Output: Global best solution, x^*

set current solution x by constructing a feasible solution *(Section 6.4)*
set the global best solution, $x^* \leftarrow x$
set the current segment, $m \leftarrow 1$
for *iteration* $= 1$ to I^{ALNS} **do**
    select a destroy heuristic and a repair heuristic using the adaptive weights w_{dm} in the current segment m *(Section 6.6)*
    generate a candidate solution x' from the current solution x using the selected destroy and repair heuristics *(Section 6.5)*
    apply local search for improving candidate solution x' if promising *(Section 6.7)*
    if x' is accepted by the simulated annealing criterion (*Section 6.5.4*) **then**
        $x \leftarrow x'$
    end
    if I^{VCP} iterations have passed since the VCP was solved **then**
        solve VCP and generate a new candidate solution x'' *(Section 6.8)*
        if x'' is accepted by the simulated annealing criterion *(Section 6.5.4)* **then**
            $x \leftarrow x''$
        end
    end
    if $f(x) < f(x^*)$ **then**
        $x^* \leftarrow x$
    end
    update scores π_d of the destroy and repair heuristics *(Section 6.6)*
    if I^S iterations have passed since last weight update **then**
        update weight $w_{d,m+1}$ for method d to be used in segment $m+1$ based on the scores π_d obtained for each method in segment m *(Section 6.6)*
        update current segment, $m \leftarrow m + 1$
    end
end

6.2 Solution Representation

A solution to the TDVRPSO is denoted x and has an associated objective function value of $f(x)$. This solution consists of a set of voyages for fleet vessels (PSVs) and potentially a voyage for the spot vessel, as well as a set of postponed orders. Recall that postponed orders are orders which are not serviced within the considered planning period. A voyage consists of a sequence of orders as is demonstrated for the vessels *PSV1*, *PSV2* and *Spot* in Figure 6.1. The set of postponed orders hold the orders not included in a voyage and are hence not serviced in the given solution. These orders can only be optional. Each order, i, in a voyage is associated with a position, s, as is illustrated in Figure 6.1. For a specific installation, the orders must be visited in the order MD $\rightarrow$ OD $\rightarrow$ OP, consistent with what is presented in Section 4.2.3. These considerations are further addressed in Section 6.3. In general, an order is associated with an installation, I, and is of a specific order type, T. Considering the example in Figure 6.1, order 9 is an optional delivery (OD) order associated to installation 7, denoted 7_{OD}. A full solution representation is exemplified in Figure 6.1, demonstrating a solution with voyages for PSV1 and PSV2, an unused spot vessel, and a set of postponed orders.

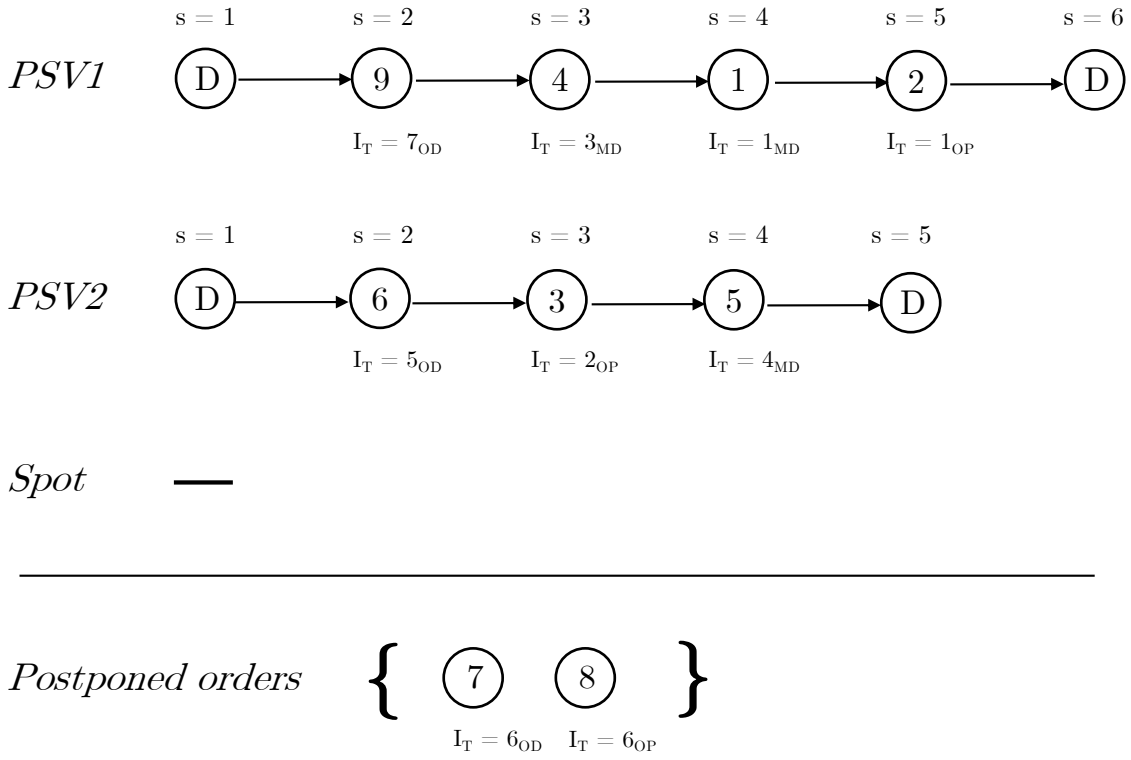


Figure 6.1: An example representation of a solution to the TDVRPSO produced by the ALNS. The solution includes voyages for two fleet vessels, i.e. PSV1 and PSV2, as well as a set of postponed orders. No voyage is performed by the spot vessel.

6.3 Ensuring Feasible Solutions

In order to handle dependencies between orders to the same installations and ensure that restrictions regarding load and voyage duration are met, we add feasibility checks to secure feasible voyages after they have been altered.

When an order is inserted into a partial solution, i.e., a solution for which there are orders in the problem that are neither part of a voyage nor postponed, several characteristics of the resulting voyage change. First, we consider the total duration of the voyage and the time spent between servicing the order preceding and the orders following the inserted order. The time spent performing the vessel activities required is recalculated to verify feasibility. The minimum sailing time between two installations is calculated by finding the maximum allowable speed when adjusting for weather conditions and applying this speed throughout the sailing leg. This is done for all installations in the new solution. Further, the idling time at an installation before servicing an order is determined by the arrival time at the installation, the calculated service duration, and the earliest possible service end time. If the total duration of the new voyage does not exceed the latest return time of the vessel, the voyage is feasible.

Secondly, the insertion of an order must be load feasible. If the inserted order is a delivery order, the start load increases. The new load will be higher than for the original voyage until the inserted order is serviced. If the load is still below the vessel capacity throughout the new voyage, the voyage is load feasible. If the inserted order is a pickup order, the start load remains unchanged. In this case, the load in the new voyage will be higher than for the original voyage from the point of picking up the order and on. If the load is still below the vessel capacity throughout the new voyage, the voyage is load feasible.

The third feasibility consideration concerns the order in which orders to the same installation are visited. If the order to be inserted is to an installation with other orders in the current solution, the order must be inserted into the same voyage. As presented in Section 4.2.3, the mandatory delivery order must be serviced first at an installation if it exists, then the optional delivery order, and finally, the optional pickup order. The order must be placed so that this ordering is upheld. Further, the order must be placed so that there are no orders to other installations separating the orders to the same installation. Such an ordering will require multiple visits to the same installation, which is not allowed.

6.4 Constructing the Initial Solution

When constructing the initial solution, all orders are initially unplaced. The orders are then inserted sequentially until all orders are either part of a vessel's voyage or postponed. The insertion is done using a *basic greedy insertion* heuristic, which is detailed in Sec-

tion 6.5.2. All feasible insertions are evaluated for each order. Evaluating an insertion involves solving the subproblem detailed in Chapter 5 for the voyage under consideration after inserting the order and then comparing the new cost with the previous cost. Alternatively, suppose the order is optional, and we want to evaluate insertion into the set of postponed orders. In that case, the additional cost will be equal to the penalty cost of postponing that order. Finally, the insertion contributing the lowest additional cost is chosen. The order associated with this insertion is then inserted, and the process is repeated. Note that all insertions are evaluated to ensure feasibility according to what is detailed in the previous section.

Algorithm 3: Construction Heuristic

Input : Set of unplaced orders ($\mathcal{U}$)
Output: Feasible solution (x)

initialize empty set of vessel voyages ($\mathcal{V}$) and empty set of postponed orders ($\mathcal{P}$)
while $\mathcal{U}$ is not empty **do**
    initialize *least increase* to positive infinity
    for order i in $\mathcal{U}$ **do**
        initialize empty set of insertions $\mathcal{S}_i$
        for v in $\mathcal{V}$ **do**
            find all feasible insertions for i in v and add to $\mathcal{S}_i$
        end
        if order i is optional and there exists an MD order to i's installation **then**
            continue
        end
        if order i is optional **then**
            add insertion into $\mathcal{P}$ to $\mathcal{S}_i$
        end
        if $\mathcal{S}_i$ is not empty **then**
            for insertion s in $\mathcal{S}_i$ **do**
                evaluate cost increase of insertion s
                if cost increase of insertion s is lower than *least increase* **then**
                    update *least increase*, set s to be the *best insertion* and i to be the *best order to insert*
                end
            end
        end
    end
    insert *best order to insert* in *best insertion* and remove *best order to insert* from $\mathcal{U}$
    update $\mathcal{V}$ and $\mathcal{P}$
end
create new solution, x, from $\mathcal{V}$, $\mathcal{P}$ and $\mathcal{U}$ (empty)

6.5 Large Neighborhood Search

In the Adaptive Large Neighborhood Search (ALNS) heuristic, a set of destroy heuristics and repair heuristics are applied. In each iteration of the heuristic, we choose one destroy and one repair heuristic. The choice of heuristic is made through a roulette wheel selection, defined by adaptive weights as described in Section 6.6.

6.5.1 Destroy Heuristics

This section details the destroy heuristics used in the ALNS. For all destroy heuristics, if a mandatory order to an installation is removed, the optional orders to that installation are also removed. This ensures that when orders are re-inserted into the partial solution using a repair heuristic, there are more feasible insertions than simply re-inserting the mandatory order into its previous position. Removing an optional order without removing the remaining orders to an installation is permitted, as a possible new solution might include postponing the order in question. The first four destroy heuristics presented are inspired by Shaw (1998) and Ropke and Pisinger (2006), while the spread and spot vessel removal heuristics are problem-specific and introduced in this thesis.

Random Removal

The random removal method randomly chooses an order to remove from all orders in the existing voyages or the set of postponed orders until at least q^{ALNS} orders are removed.

Related Removal

The idea behind the related removal method is to find q^{ALNS} orders that have similar characteristics and remove these. The method chooses the first order to remove randomly. The remaining orders are sorted by relatedness, defined by the relative distances between the orders' installations. Using mathematical notation, we have that

$$r_{ij} = d_{ij}, \tag{6.1}$$

where r_{ij} denotes the relatedness and d_{ij} the distance between orders i and j. The next order to remove is then chosen as the element at position R in the sorted list of orders, corresponding to the random number

$$R = y^p \times |\text{TasksSortedByRelatedness}|, \tag{6.2}$$

where y is a random number between 0 and 1, and the exponent $p > 1$ is a determinism parameter deciding the degree of randomness in the selection process. Increasing p decreases y^p, thus decreasing the degree of randomness, vice versa. Recall that if we want to remove a mandatory order and there exists orders to the same installation in the considered voyage, all orders to the installation are removed. The sorting of the remaining orders is left unchanged. The algorithm for related removal is presented below.

Algorithm 4: Related Removal

Input : Solution (x), determinism parameter (p), number of orders to remove (q)
Output: Partial solution (x')

initialize empty set for unplaced orders ($\mathcal{U}$)
choose a random order i from a random v in $\mathcal{V}$
add order i to $\mathcal{U}$
while $|\mathcal{U}| < q^{ALNS}$ **do**
    choose a random order i in $\mathcal{U}$
    sortedOrders $\leftarrow$ sorted list of orders not in $\mathcal{U}$ based on relatedness to order i
    choose random number $y \in [0, 1]$
    $R = y^P \times$ |sortedOrders|
    add the order at place R in sortedOrders, j, to $\mathcal{U}$
    if order j is a mandatory delivery order **then**
        add all orders to the same installation as j to $\mathcal{U}$
    end
    remove all orders in $\mathcal{U}$ from the relevant set of vessel voyages ($\mathcal{V}$) or the set of postponed orders ($\mathcal{P}$)
end
create partial solution, x', from $\mathcal{V}$, $\mathcal{P}$ and $\mathcal{U}$

Worst Removal

The worst removal method searches for the removal that contributes the highest additional cost and the largest contribution to the objective function. The contribution is calculated by solving the subproblem detailed in Chapter 5 for the relevant voyage after removing the order and then comparing the new cost with the previous cost. Alternatively, suppose the order is optional, and we want to evaluate removal from the set of postponed orders. In that case, the contribution will be equal to the penalty cost of postponing that order. The orders are sorted by their relative contribution, ranging from greatest to smallest. After sorting the orders, the order to remove is chosen by the same selection process as for related removal, introducing randomness to the process.

Cluster Removal

Like the related removal method, the cluster removal heuristic aims to remove orders of close geographical proximity. As the name suggests, the heuristic separates orders in the same voyage into two clusters and removes one of these clusters. Removing a collection of orders, re-insertion of the same orders into the same positions will be less likely, as we remove all orders in the cluster. This method will also prevent orders from clusters being left on the original voyage, which may occur when applying related removal.

In order to cluster the orders serviced on a given voyage, we apply the *k-means* algorithm with $k = 2$. This algorithm first chooses the geographical coordinates of two random orders as centroids and assigns the orders serviced on the voyage to the closest centroid. The algorithm then starts iterating towards the best clusters. The following three points sum up an iteration:

1. The sum of squared distances between all orders and both centroids are calculated.
2. All orders are assigned to the closest centroid, resulting in two new clusters.
3. New centroids are computed by finding the average position of all orders that belong to the given clusters.

The algorithm terminates when there is no change to the two centroids in an iteration. This means that we have found the same clusters twice in a row. One of the two clusters is then removed at random. If fewer than q^{ALNS} orders are removed, a new voyage is chosen at random. If there are no more voyages to remove from, the algorithm terminates and returns the partial solution. Hence, the heuristic may return a partial solution where either more or fewer than q^{ALNS} orders are removed.

Spread Removal

The spread removal heuristic removes orders that are far from all other orders in a voyage. It starts by finding the minimum distance from each order to another order in a voyage. Next, it removes the order which has the longest minimum distance to another order. The heuristic then finds new distances for the remaining orders and removes the next order based on the same criterion. This process is repeated until q^{ALNS} orders are removed.

Spot Vessel Removal

The spot vessel removal heuristic removes all orders serviced by the spot vessel in the solution. If no orders are serviced by the spot vessel in the given solution, the heuristic cannot be chosen.

6.5.2 Repair Heuristics

This section details the repair heuristics used in the ALNS. These heuristics receive a partial solution consisting of partial voyages, a set of postponed orders, and a set of unplaced orders. The unplaced orders are those that have been removed by one of the destroy heuristics. The repair heuristic chosen inserts the unplaced orders into a partial voyage or the set of postponed orders. The result is a feasible solution consisting of voyages and postponed orders, as illustrated in Figure 6.1. The first two repair heuristics presented are inspired by Shaw (1998), while the maximum penalty cost and maximum order size insertion heuristics are problem-specific and introduced in this thesis.

Basic Greedy Insertion

The basic greedy insertion method evaluates the effect of inserting an order from the set of unplaced orders in all the feasible insertions for that order. This process is completed for all the unplaced orders. The order to insert is then chosen to be the order with the smallest contribution to the objective function, referred to as the *best order to insert*. As described in Section 6.4, evaluating an insertion involves solving the subproblem detailed in Chapter 5 for the relevant voyage after inserting the order and then comparing the new cost with the previous cost. Alternatively, suppose the order is optional, and we want to evaluate insertion into the set of postponed orders. In that case, the additional cost will be equal to the penalty cost of postponing that order. If the best order to insert is not mandatory but is from an installation with a mandatory order yet to be placed, the next best order is chosen. This process is repeated until all unplaced orders are inserted. The order in which the orders are inserted is solely determined by the best insertion given the available insertions for the various orders.

In mathematical terms, the order leading to the smallest increase in the objective function when inserted at its best position is given in Equation (6.3). In this equation, i is an order, s a feasible insertion in the partial solution considered, $\mathcal{S}_i$ the set of feasible insertions for order i and $\mathcal{U}$ the set of unplaced orders. Consequently, $i(s)$ denotes the insertion of order i in position s.

$$\underset{i \in \mathcal{U},\, s \in \mathcal{S}_i}{\operatorname{argmin}} \; \Delta\text{Cost}(i(s)) \tag{6.3}$$

.

Regret Insertion

In the basic regret method, i.e., regret-2, the regret value is the difference between the change in the objective function caused by the best insertion of an order and the second best insertion of the same order. Finding the change in objective function value from inserting an order is done equivalently to what is described for the basic greedy insertion heuristic. Generalizing for more than two insertions, regret–k insertion is defined as an insertion method that compares the k best insertions of an order and calculates the regret value as the sum of the cost difference between the k best insertions. As for the basic greedy insertion method, evaluating an insertion involves solving the subproblem detailed in Chapter 5 for the relevant voyage after inserting the order and then comparing the new cost with the previous cost. Here, the insertion with the largest regret value is chosen to be inserted. If this order, i.e., the best order to insert, is not mandatory but is from an installation with a mandatory order, the next best order to insert is chosen. This process is then repeated until all unplaced orders are inserted.

Denoting $i(s_1)$ the best and $i(s_{k'})$ the k'-th best insertion of order i, respectively, the regret value of the insertion is given by Equation (6.4). Note that s_1 and $s_{k'}$ may be insertions in different vessels or potentially into the set of postponed orders if the orders are optional.

$$\underset{i \in U}{\operatorname{argmin}} \left\{ \sum_{k'=2}^{k} \Delta\text{Cost}(i(s_1)) - \Delta\text{Cost}(i(s_{k'})) \right\} \tag{6.4}$$

.

Maximum Penalty Cost Insertion

The maximum penalty cost insertion heuristic aims to prevent situations in which orders with high penalty costs must be postponed. The heuristic first sorts the orders so that all mandatory orders are first. It then sorts the optional orders by descending penalty costs for postponing the orders. Further, the heuristic inserts the orders sequentially according to the sorted list of orders. The insertions are done greedily following what is presented for the *basic greedy insertion* method, i.e., the order under consideration is inserted in its best position based on the available feasible insertions for that order. However, the ordering of insertion is determined before the orders' insertion costs are evaluated, making the heuristic different from basic greedy insertion.

Maximum Order Size Insertion

The maximum order size insertion heuristic prioritizes orders that are of larger sizes. Thus, the heuristic is equivalent to the maximum penalty cost insertion, except that the optional orders are sorted by descending size.

6.5.3 Applying Noise in the Insertion Methods

Having identified the insertion heuristics as being quite myopic, Ropke and Pisinger (2006) introduce a noise term to the objective function so that the methods do not always make the move that is locally optimal. This is also done for the ALNS heuristic presented in this chapter. Every time the cost C of an insertion of an order is calculated, a random number *noise* is calculated. This is found in the interval $[-\min N, \max N]$, where $\max N = \eta \times \max\{d_{ij}\}$ due to the importance of relative distances in the problem. The modified cost of insertion is given by $C' = \max\{0, C + noise\}$.

6.5.4 Acceptance and Stopping Criteria

Inspired by Ropke and Pisinger (2006), a simulated annealing-based acceptance criterion is used for the ALNS heuristic. A candidate solution, x' is always accepted if the objective value created by applying the chosen removal and insertion heuristics, $f(x')$, is better than the current objective value, $f(x)$. If this is not the case, a candidate solution may still be accepted with probability $e^{-(f(x')-f(x))/T}$, where $T > 0$ is referred to as the *temperature*. This implies that we sometimes choose to accept worse solutions than the current solution, thus introducing randomness and lowering the chance of getting stuck in a local optimum. The cooling temperature, T is set to an initial value, T_{start}, and for each iteration updated by scaling the current temperature with a cooling rate, $\xi \in (0, 1)$, such that $T = T \times \xi$. This implies that we decrease T throughout the search, allowing for more exploration in the early stages of the search and more exploitation towards the final stages. To avoid a temperature that is so low that we get stuck in a local optimum, we set a maximum of θ iterations. If θ iterations are completed, the next new solution is accepted. The algorithm stops after completing a maximum number of iterations, I^{ALNS}.

6.6 Choosing a Destroy and Repair Heuristic

To generate a candidate solution, one destroy and one repair heuristic must be selected. In order to identify well-performing heuristics and increase the probability of selecting these, each heuristic is assigned a weight. As the search progresses, the weights are updated relative to their contribution to exploring new and promising areas of the search space. The heuristics to apply in an iteration are selected by the roulette wheel selection principle, where heuristics with larger weights have a higher probability of being selected.

The search is divided into segments. Inspired by Ropke and Pisinger (2006), the weights are updated after each segment as defined by Equation (6.5). w_{dm} denotes the weight of heuristic d in segment m.

$$w_{d,m+1} = (1 - r)w_{dm} + r\frac{\pi_d}{\theta_d} \tag{6.5}$$

Initially, all weights are assigned equal non-zero weights. At the beginning of a segment, the *score* of heuristic d, π_d is set to zero. The score reflects the performance of a heuristic during a segment and is the sum of its received rewards throughout the segment. The rewards used in the search are shown in Table 6.1. Also, θ_d counts the number of times the heuristic d has been selected in the current segment. r is called the *reaction* parameter and determines the influence of the scores from the current segment on the weight update.

Table 6.1: Overview of the various parameters by which the score of a heuristic may increase, and what reward criterion must be fulfilled in order for the increase to be added to the score.

Parameter	Reward Criterion
σ_1	Candidate solution resulted in global improvement
σ_2	Candidate solution has not been accepted before and the solution is better than the current solution
σ_3	Candidate solution has not been accepted before, the solution is worse than the current solution, but is accepted by the simulated annealing criterion

The probability of choosing heuristic d in the roulette wheel selection is given by Equation (6.6). $\hat{d} \in I$ denotes heuristic $\hat{d}$ in the set of either destroy or repair heuristics I.

$$P(d) = \frac{w_{dm}}{\sum_{\hat{d} \in I} w_{\hat{d}m}} \tag{6.6}$$

6.7 Local Neighborhood Search

To assist the ALNS heuristic in finding high-quality solutions faster, we implement a local neighborhood search. Local search operators (LSOs) can be used to improve solutions produced by the ALNS heuristic. The search pursues improving a solution by investigating the *neighbors* of the current solution, using operators that marginally alter the solution. To ensure feasibility, the considerations presented in Section 6.3 are taken. Section 6.7.1 presents the LSOs implemented, while Section 6.7.2 describes the setup of the ALNS heuristic supported by the local search.

6.7.1 Local Neighborhood Search Operators

This section presents the local search operators (LSOs) used for the local search. The operators are inspired by the LSOs used by Gendreau et al. (1992) and Korsvik et al. (2011).

Intra-Voyage Relocate

The intra-voyage relocate operator moves one order from position s_1 to s_2 within the same vessel voyage v. Note that when moving an order associated to an installation with other orders, all orders to this installation are moved. This is to ensure feasibility according to what is discussed in Section 6.3. Such a move is exemplified in Figure 6.2.

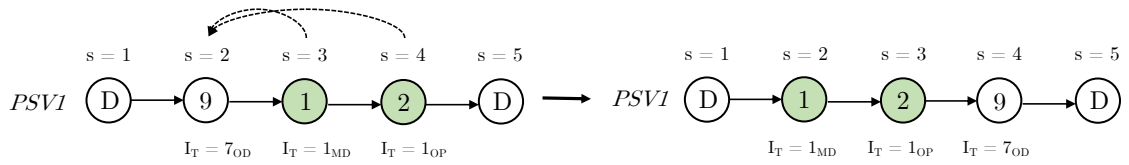


Figure 6.2: An example of a move resulting from using the intra-voyage relocate operator. Order 1 and 2 are relocated from positions s = 3 and s = 4 to s = 2 and s = 3, respectively. Both orders are moved because they belong to the same installation.

Inter-Voyage Relocate

The inter-voyage relocate operator moves the order in position s_1 in one vessel voyage to position s_2 in another vessel voyage. Such a move is exemplified in Figure 6.3.

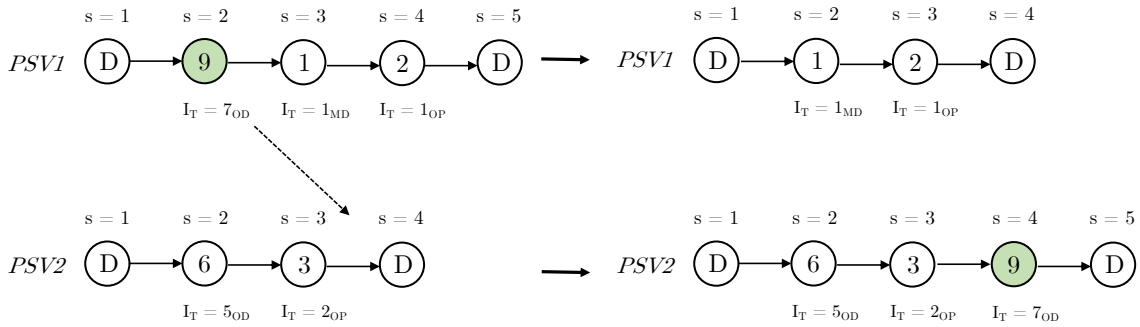


Figure 6.3: An example of a move resulting from using the inter-voyage relocate operator. Order 9 is moved from s = 2 in *PSV1* to s = 4 in *PSV2*.

Intra-Voyage Exchange

The intra-voyage exchange operator exchanges the order in position s_1 in one vessel voyage with the order in position s_2 in the same vessel voyage. Such a move is exemplified in Figure 6.4

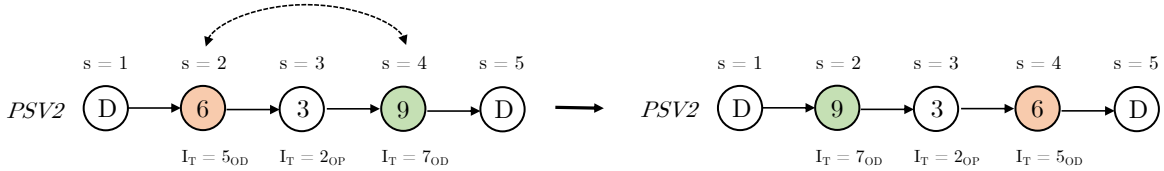


Figure 6.4: An example of a move resulting from using the intra-voyage exchange operator. Order 9 is moved from s = 4 to s = 2, while order 6 moves the opposite way.

Inter-Voyage Exchange

The inter-voyage exchange operator exchanges the order in position s_1 in one voyage with the order in position s_2 in another voyage. Such a move is exemplified in Figure 6.5.

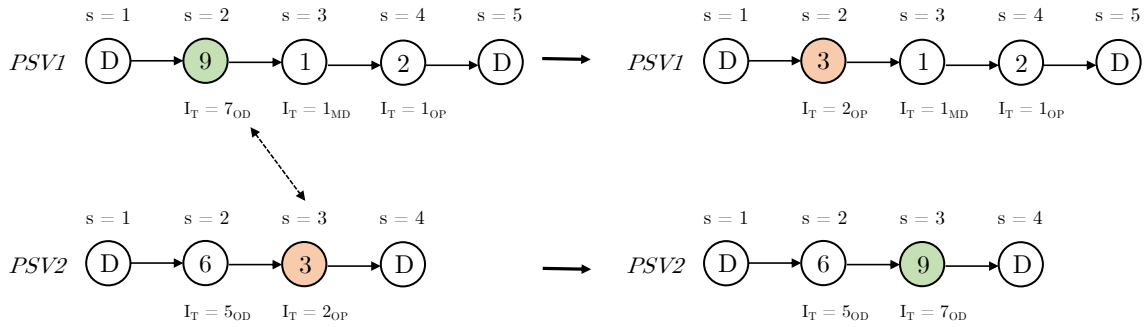


Figure 6.5: An example of a move using the inter-voyage exchange operator. Order 9 is moved from s = 2 in *PSV1* to s = 4 in *PSV2*, while order 3 does the opposite move.

Schedule Postponed

The schedule postponed operator moves an order in the set of postponed orders and inserts it into position s in a vessel voyage. Such a move is exemplified in Figure 6.6.

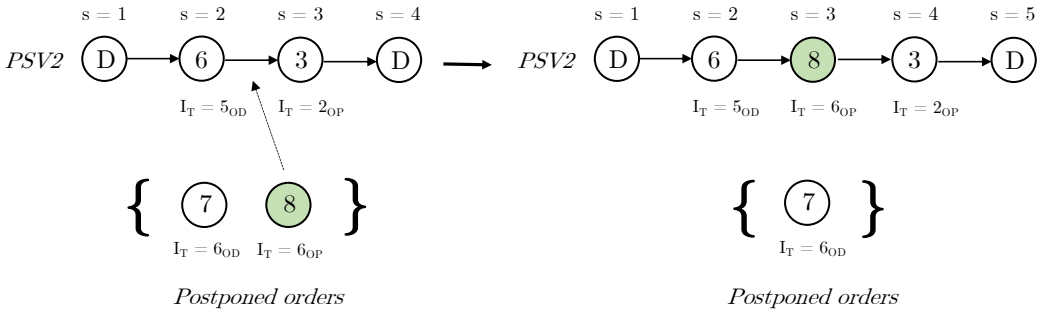


Figure 6.6: An example of a move resulting from using the schedule postponed operator. Order 8 is from the postponed order set and inserted at s = 3 in PSV2.

Postpone Scheduled

The postpone scheduled operator moves an order from a position s in a vessel voyage to the set of postponed orders. Such a move is exemplified in Figure 6.7.

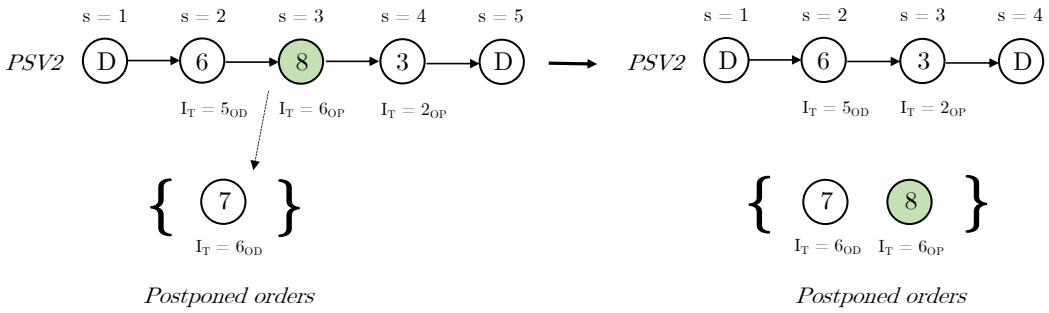


Figure 6.7: An example of a move resulting from using the postpone scheduled operator. Order 8 is removed from s = 3 in *PSV2* and added to the postponed order set.

Voyage Exchange

The voyage exchange operator swaps the voyage sailed by one PSV with the voyage sailed by another. This operator is introduced to take advantage of a situation where two voyages in a solution may be sailed by two different PSVs. This requires that the two voyages are load feasible for both PSVs. Since different PSVs provide different fuel consumption costs, solutions resulting from a voyage swap will likely change the total costs, and consequently, the objective value of a solution. The voyage exchange move is exemplified in Figure 6.8.

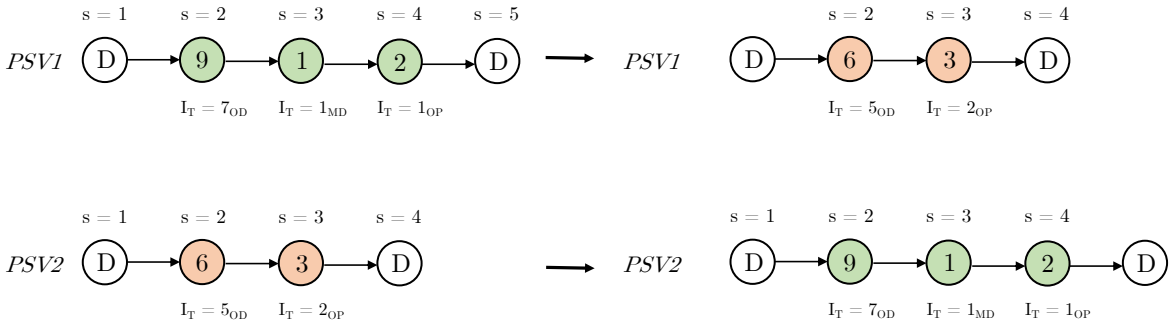


Figure 6.8: An example of a move resulting from using the voyage exchange operator. The entire voyage sailed by *PSV1* is swapped with that sailed by *PSV2*.

6.7.2 Local Neighborhood Search Strategy

The local neighborhood search works as an addition to the ALNS heuristic. This means that the LSOs are used after the heuristic has produced a new candidate solution, attempting to further improve the objective value by reducing costs. A local search is performed when an iteration of the ALNS heuristic has resulted in a candidate solution with an objective value that is within β % of the current best solution. The LSOs are then applied sequentially, making all possible moves for each of the operators. The solution improving the input solution to an LSO the most is chosen and used as input to the next LSO. If the input solution is not improved by an LSO, it is passed to the next. After all LSOs are applied, the solution provided by the final LSO is returned, and a new iteration of the ALNS begins. If no LSO finds an improving solution, the solution outputted from the local search is just the candidate solution provided as input from the ALNS iteration.

6.8 The Voyage Combination Problem

As described in Section 6.2, solutions produced by the ALNS heuristic consist of one voyage for each vessel. This suggests that a relatively poor solution may still contain individual voyages that perform well. In searching for improving solutions, we introduce and solve the Voyage Combination Problem (VCP), a set partitioning problem using voyages previously encountered in the search. The problem setup is inspired by Homsi et al. (2020) who use

a set partitioning formulation when solving the Industrial and Tramp Ship Routing and Scheduling Problem (ITRSP). Solving the VCP involves finding the best combination of voyages such that all mandatory orders are serviced, and all optional orders are either serviced or postponed. The problem is solved every I^{VCP} iterations of the ALNS. The resulting best solution is accepted based on the same criterion as a solution produced by a destroy/repair perturbation. The model formulation uses notation introduced in Section 4.3, as well as additional notation relevant to the problem. The new notation, model objective function, and constraints are presented below. As previously introduced, orders are classified as mandatory delivery (MD), optional delivery (OD), or optional pickup (OP).

Sets and Indices

$\mathcal{K}$ - set of installations that have both OD and OP orders, but no MD order

$\mathcal{R}_v$ - set of available voyages for vessel v, including all previously found voyages

$\tilde{\mathcal{R}}_{kv}$ - subset of $\mathcal{R}_v$ of available voyages for vessel v that visit installation k, but only services one of the optional orders

Parameters

C_{rv} - cost of using voyage r for vessel v

A_{irv} - $\begin{cases} 1, & \text{if order } i \text{ is serviced on voyage } r \text{ by vessel } v \\ 0, & \text{otherwise} \end{cases}$

Decision Variables

y_{rv} - $\begin{cases} 1, & \text{if voyage } r \text{ is used by vessel } v \\ 0, & \text{otherwise} \end{cases}$

z_i - $\begin{cases} 1, & \text{if order } i \text{ is not serviced} \\ 0, & \text{otherwise} \end{cases}$

Objective Function

The objective function (6.7) in the model for the VCP minimizes the sum of the monetary costs associated with a specific vessel choosing a specific voyage and penalty costs assigned

to optional delivery or optional pickup orders that are postponed.

$$\min \sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} C_{rv} y_{rv} + \sum_{i \in \mathcal{N}} C_i^P z_i \tag{6.7}$$

Constraints

Constraints (6.8) ensure that a voyage may be chosen at most once by a vessel. Further, the voyages generated in the solutions produced by the ALNS introduce a new concern. Suppose an installation has two optional orders associated with it. In that case, we must ensure that a voyage containing only one of the optional orders cannot be combined with a voyage containing only the other. This situation may arise because servicing the OD order and postponing the OP order can be feasible in one solution, while the opposite can be feasible in another. These considerations are handled by Constraints (6.9). Constraints (6.10) secure that all orders are either serviced on a voyage by a vessel or postponed, and Constraints (6.11) disable the opportunity of postponing mandatory orders.

$$\sum_{r \in \mathcal{R}_v} y_{rv} \leq 1, \quad v \in \mathcal{V} \tag{6.8}$$

$$\sum_{v \in \mathcal{V}} \sum_{r \in \tilde{\mathcal{R}}_{kv}} y_{rv} \leq 1, \quad k \in \mathcal{K} \tag{6.9}$$

$$\sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} A_{irv} y_{rv} + z_i = 1, \quad i \in \mathcal{N} \tag{6.10}$$

$$z_i = 0, \quad i \in \mathcal{N}^{MD} \tag{6.11}$$

Constraints (6.12) and (6.13) specify the domains of the decision variables, putting binary requirements on the use of voyages for vessels and the servicing of orders, respectively.

$$y_{rv} \in \{0,1\}, \quad r \in \mathcal{R}_v, v \in \mathcal{V} \tag{6.12}$$

$$z_i \in \{0,1\}, \quad i \in \mathcal{N} \tag{6.13}$$

Chapter 7

Implementation and Test Instances

This chapter specifies details in the implementation of the solution methods proposed in this thesis. Furthermore, it presents the test instances which have been generated based on data provided by Equinor. Section 7.1 presents a vessel voyage planning case and its relevant data. Section 7.2 categorizes significant wave heights into *weather states*, and explains how these weather states impact PSV operations. Next, Section 7.3 outlines three *weather scenarios* representing different weather situations in the planning horizon, before Section 7.4 explains how the test instances are generated. Section 7.5 presents how the PSV fuel consumption is approximated in this thesis. Finally, Section 7.6 describes how the time discretization and penalty cost parameters introduced in Chapter 4 are set.

7.1 The Mongstad Case

Mongstad is an industrial site in Vestland county in Norway housing a supply depot, the Mongstad supply base, servicing 27 offshore installations located in the North Sea at eight different oil-and-gas fields. The harbor at Mongstad is one of the largest oil-and-gas product harbors in Europe in terms of tonnage produced (Equinor, 2020a). Equinor has provided access to data relevant for servicing the installations from the Mongstad supply base. This data is the basis of our interpretation of the problem supply vessel planners at Equinor face when scheduling voyages for PSVs and will be referred to as the Mongstad case. Figure 7.1 presents the installations and the supply depot in the Mongstad case.

Data concerning the installations are provided in Appendix B. The data consists of location, opening hours, and standard order sizes of each order type for all installations. The locations are used to calculate the distances between installations and distances from the Mongstad supply base to the installations. The standard order sizes reflect the typical size of an order type from the installations. These are estimated based on the activity level at each installation. The order sizes are measured in *cargo units*. One cargo unit is the space an offshore container occupies on the PSV decks, which is typically eight square

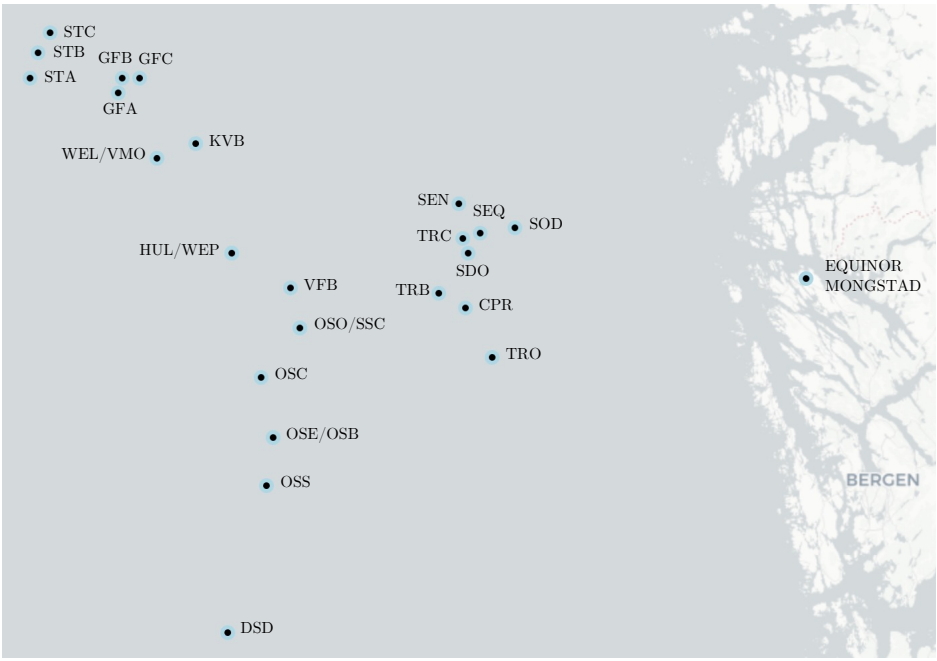


Figure 7.1: Map of the 27 offshore installations serviced from the Mongstad supply base.

meters (HACON Containers, 2020). Out of the 27 installations, four have limited opening hours. These installations are open from 07:00 to 19:00. Also, vessel preparation at the depot begins at 08:00 and lasts until 16:00.

The contracted fleet used in this thesis consists of five vessels. Additionally, one vessel from the spot market can be chartered. The deck cargo capacities of these vessels are based on the capacities of five vessels operating from Mongstad with long-term contracts with Equinor and one vessel from the spot market in the North Sea. Appendix C outlines the capacity data of these vessels.

Furthermore, data about energy and charter costs, speed limits, and service time are provided. The energy resource consumed by the PSVs is assumed to be marine diesel oil. Thus "energy consumption" will be referred to as "fuel consumption" from this point on. The service time is an estimate of the time required to transfer one cargo unit from a vessel to an installation. Table 7.1 presents the outlined data.

Table 7.1: Fuel, chartering, speed, and service time data for the PSVs. Note that the charter cost only applies to the spot vessel as this is a sunk cost for the fleet vessels.

Parameter	Fuel cost	Charter cost	Speed interval	Service time per cargo unit
Value	276 USD/ton	608 USD/hour	10-14 kts	10 minutes

A voyage for a PSV servicing installations from the Mongstad supply base normally lasts up to three days. Furthermore, the vessel preparation for the next voyage starts at 08:00 at the supply depot. Therefore, the planning horizon starts at 08:00 on day 0 and ends at 08:00 on day 3.

7.2 Weather Impact

As the TDVRPSO is weather-dependent, the weather impact on PSV operations must be modeled. Modeling this weather impact accurately is a complex exercise in which wave height and direction and wind strength and direction are among essential factors that may complicate and possibly prevent activities performed by the PSVs. As mentioned in Section 4.2.1, the handling of weather in this thesis is inspired by Halvorsen-Weare and Fagerholt (2011), simplifying the weather impact to be a univariate stochastic variable of significant wave height, as this is considered the most significant factor by supply vessel planners. We categorize the weather conditions into four weather states (WS) based on significant wave height (SWH), as shown in Table 7.2. Note that the solution methods presented can handle more complex descriptions of the weather conditions, potentially achieved by simulating the routing of PSVs under various weather conditions. However, this would involve advanced hydrodynamics, which is beyond the scope of this thesis.

Rough weather states add more resistance to PSV operations than less rough weather states. This resistance is realized in four ways. Firstly, the maximum allowed sailing speed is decreased in weather states 2 and 3. Secondly, an offset in sailing speed is considered when calculating the fuel consumed when sailing in weather states 2 and 3. Denoting the offset Δs_w given weather state w and the sailing speed in perfect weather s, the fuel consumption is calculated by inserting s and Δs_w into the fuel consumption function presented in Section 7.5. Thirdly, the service time increases in weather states 1 and 2, while servicing is prohibited in weather state 3. The installation is then closed for service. Finally, the fuel consumption for idling and servicing increases in weather states 1, 2, and 3. Columns 3 to 6 in Table 7.2 present these impact parameters.

Table 7.2: Impact on vessel speed and fuel consumption in different weather states, i.e. for different significant wave heights.

WS	SWH (m)	Decrease sailing speed (kts)	Offset sailing speed (kts)	Increase service time (%)	Increase fuel consumption idling and servicing (%)
0	≤ 2.5	0	0	0	0
1	$(2.5, 3.5]$	0	0	20	20
2	$(3.5, 4.5]$	2	2	30	30
3	≥ 4.5	3	3	*No service*	100

7.3 Weather Scenarios

As demonstrated in Section 7.2, weather conditions have considerable effects on how a vessel may perform its activities during a voyage. Hence, based on the work of Ulsrud and Vandvik (2020), three *weather scenarios* are introduced to capture different characteristics

of the performance and solutions of the arc-flow model and the ALNS in Chapters 8 and 9. These weather scenarios are visualized in Figure 7.2. Essentially, a weather scenario is a sequence of the weather states presented in Section 7.2 through the planning horizon. The scenario names *Perfect*, *Mixed*, and *Critical* describe the weather conditions experienced by a vessel in a given scenario.

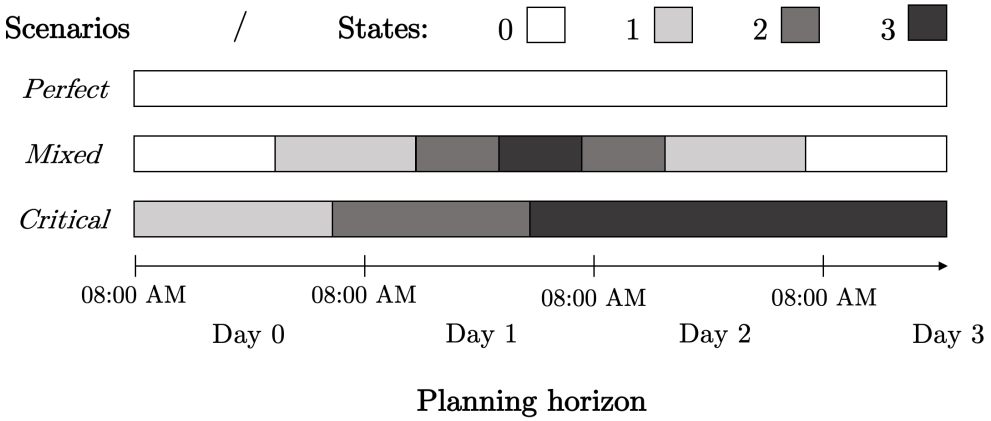


Figure 7.2: Visualization of the applied weather scenarios, detailing which weather states are experienced at what times during the planning horizon.

The scenario *Perfect* refers to the situation where the weather conditions have no impact on vessel activities as weather state 0 is forecast for the entire planning horizon. The scenario *Mixed* presents a situation where all weather states occur during the planning horizon. Finally, in scenario *Critical* weather conditions deteriorate over the course of the planning horizon. The increasing weather impact makes vessel activities harder and more expensive to perform before weather state 3 is reached, resulting in closed installations for the remainder of the planning horizon.

7.4 Test Instances

Based on the presented data, instances of the TDVRPSO are generated as described in Algorithm 5. First, the specified number of installations to have orders, I^{orders}, are sampled randomly from the set of 27 installations. Then, Lines 2 to 4 sample the number of MD, OD, and OP orders such that at least 50% and at most 70% of the I^{orders} installations with orders will have an MD order, at least 20% and at most 40% will have an OD order, and at least 20% and at most 40% will have an OP order. Line 5 distributes the orders randomly among the I^{orders} installations. Note that one installation cannot have multiple orders of the same type but may have multiple orders of different types. Lines 6 and 7 set the sizes of each order by sampling from a continuous uniform distribution ranging from 50% to 150% of the standard order size of the appropriate order type and installation. Finally, Line 9 ensures that the total capacity of all available vessels, including the spot vessel, covers the total order size.

Algorithm 5: Test Instance Generation Procedure

Input : Set of installations ($\mathcal{I}$), set of vessels ($\mathcal{V}$), number of installations to have orders (I^{orders}), order size deviation (δ), standard order sizes for each order type and installation $n \in \mathcal{I}$ (OS_n^{MD}, OS_n^{OD}, OS_n^{OP})

Output: Test instance

create subset of installations of size I^{orders} by sampling randomly from $\mathcal{I}$
sample number of MD orders randomly in the interval $[0.5I^{orders}, 0.7I^{orders}]$
sample number of OD orders randomly in the interval $[0.2I^{orders}, 0.4I^{orders}]$
sample number of OP orders randomly in the interval $[0.2I^{orders}, 0.4I^{orders}]$
distribute MD, OD, and OP orders randomly among the subset of installations
for o in the set of orders of type T ($T \in \{MD, OD, OP\}$) **do**
    set order size $\in [(1 - \delta)OS_n^T, (1 + \delta)OS_n^T]$
end
assign fleet vessels and spot vessel such that their capacity covers the total order size

The generated test instances are presented in Appendix D and summarized in Table 7.3. The instances are divided into 12 *instance groups* based on the number of installations. Further, all instances in an instance group have the same number of fleet vessels. Each instance group has five instances, so a total of 60 instances are made. The notation I-O-V-X is used to identify the various instances, where I denotes the number of installations, O the number of orders, and V the number of available fleet vessels. Finally, X is used to distinguish instances with the same number of installations, orders, and vessels.

Table 7.3: Test instances used in Chapters 8 and 9. The instances are grouped based on the number of installations. The number of orders vary within a group, but the number of available fleet vessels is the same. Chartering is always possible.

Installations	Orders	MD	OD	OP	Fleet vessels	Notation
5	5-7	3	1-2	1-2	1	5-(5-7)-1-(x)
7	8-9	4-5	2	2	1	7-(8-9)-1-(x)
9	9-11	5-6	2-3	2-3	1	9-(9-11)-1-(x)
11	12-15	6-7	3-4	3-4	2	11-(12-15)-2-(x)
13	14-18	7-9	3-5	3-5	2	13-(14-18)-2-(x)
15	15-21	8-10	3-6	3-6	2	15-(15-21)-2-(x)
17	18-23	9-11	4-6	4-6	3	17-(18-23)-3-(x)
19	19-25	10-13	4-7	4-7	3	19-(19-25)-3-(x)
21	23-28	11-14	5-8	5-8	3	21-(23-28)-3-(x)
23	24-31	12-16	5-9	5-9	4	23-(24-21)-4-(x)
25	26-33	13-17	5-10	5-10	4	25-(26-33)-4-(x)
27	28-37	14-18	6-10	6-10	5	27-(28-37)-5-(x)

By varying the number of installations with orders, order types, order sizes, and the number of vessels available, the instances represent different planning situations faced by the planners at Equinor. This way, the performance and solutions of the proposed solution methods can be evaluated and compared. All test instances are assigned the *Perfect* weather scenario. In order to study the effects of weather in Chapter 9, some instances are assigned a different weather scenario.

7.5 Modeling Fuel Consumption

Each vessel activity has a unique fuel consumption. The fuel consumption for idling and servicing is modeled as linear functions of the time spent performing the activity. The fuel consumption in kilograms per hour for these activities are given in Table 7.4. Idling and servicing will require more fuel in rougher weather conditions, according to the rightmost column in Table 7.2. Note that the fuel consumption for servicing will increase on top of this, as the service time will increase in weather states 1 and 2.

Table 7.4: Fuel consumption per hour for idling and servicing.

Vessel activity	**Fuel consumption (kg/hour)**
Idling	120
Servicing	170

Following up on the discussion about fuel consumption in Section 4.2.1, the sailing fuel consumption in kilograms at a given distance is assumed to be a cubic function of the vessel sailing speed, identical to the fuel consumption function presented by Norlund and Gribkovskaia (2017). As emphasized by Lindstad et al. (2013), a vessel will consume more fuel in rough weather compared to perfect weather due to the added resistance. It is assumed that engine power output is the same in all weather states. The impact from waves will slow down the vessels, causing the same engine power output to yield different speeds depending on the weather state. Hence, more weather impact will increase the vessel fuel consumption at a given speed. In this thesis, weather impact on fuel consumption is accounted for by adjusting for a speed loss Δs_w, as described in Section 7.2. The fuel consumption is thus dependent on the planned sailing speed s in perfect weather, and the weather conditions determining the speed loss Δs_w, and is given by

$$FC^w(s,\ \Delta s_w,\ d,\ v) = \frac{d}{s - \Delta s_w} FC_v^{s_0} \left(\frac{s}{s_0}\right)^3, \tag{7.1}$$

where d is the distance sailed and $FC_v^{s_0}$ is the fuel consumption from sailing at the design speed s_0 with vessel v. The design speed is set to 12 knots for all vessels, and each vessel's fuel consumption per hour in design speed is given in Appendix C. Applying the weather states introduced in Section 7.2, the weather conditions may change from one time interval to the next. Consequently, fuel consumption is also time-dependent.

Equation (7.1) outputs the fuel consumption in kilograms when the speed, distance, and weather are provided as input. Aiming to study the fuel efficiency of sailing at different speeds, the fuel consumption in kilograms per nautical mile (kg/nmi) for the different weather states is plotted in Figure 7.3. Observe that the cost-optimal speed is the lowest allowable speed for all weather states. Provided that problem restrictions imposed by opening hours, return times, and weather conditions do not incentivize an increase in

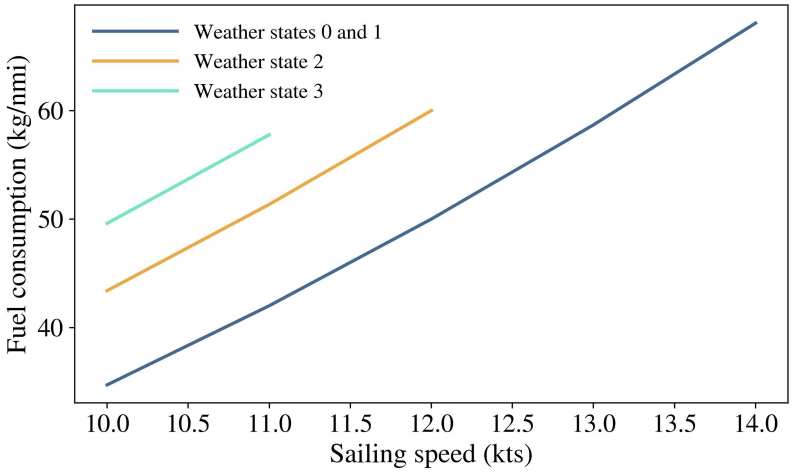


Figure 7.3: Fuel consumption as a function of sailing speed showing curves for the different weather states presented in Section 7.2.

sailing speed, this speed will always be chosen. Finally, recall that the solution methods proposed in this thesis can handle any non-linear fuel consumption function. Hence, Equation (7.1) can easily be replaced by any other fuel consumption approximation method.

7.6 Determining Time Discretization and Penalty Cost

Section 7.6.1 explains the reasoning behind the chosen time discretization parameter, and Section 7.6.2 does the same for the penalty cost of not servicing an order.

7.6.1 Time Discretization

The time discretization parameter γ sets the length of the discrete time intervals in a given planning horizon. The parameter denotes the number of discrete time periods per hour and is defined accordingly in Equation (7.2). Thus, the larger the discretization parameter, the shorter the time intervals.

$$\gamma = \frac{1 \text{ hour}}{\text{Length of time intervals (hours)}} \tag{7.2}$$

Discretizing time in the arc-generation procedure used in Chapters 4 and 5 yields discrete start and end times for the arcs. The larger the discretization parameter, the more feasible start and end times exist. The start and end times of an arc define the sailing time included in the arc. In other words, they determine the sailing speed. Consequently, the larger the discretization parameter, the more speed alternatives. Since the cost of fuel consumed by a vessel is a function of speed, this further implies that a larger discretization parameter provides more cost alternatives. Therefore, providing the mathematical models from Chapters 4 and 5 with more cost alternatives will yield more accurate solutions. However, as each cost alternative is equivalent to an arc, a larger discretization parameter

introduces more feasible arcs, increasing the complexity and solution times. Hence, there is a trade-off between the size of the discretization parameter and the solution times.

When deciding on a time discretization parameter, discretized time intervals of lengths shorter than the minimum time it takes to travel from one installation to another are desired. A too large time interval between two consecutive discrete time points may result in a vessel arriving at the same discrete point in time, regardless of whether it sails at the minimum or maximum speed. A time discretization parameter of $\gamma = 4$ results in a time interval of 15 minutes, shorter than the sailing time on 97.5% of the distances between installations at the maximum allowable speed in the TDVRPSO. For these installations, the start and end times of the associated arcs will be different. This will also be the case for the remaining installations, provided that the total time taken to perform the vessel activities of the arc in question exceeds that of one discrete time interval. Hence, $\gamma = 4$ is chosen as the time discretization parameter in this thesis.

7.6.2 Penalty Cost of Postponing Optional Orders

Postponing an optional order means that the order must be serviced at a later time. Hence, this action must be penalized in a meaningful way. This is done using the penalty cost described in Chapters 4 and 6. In order to provide sensible solutions, the penalty cost should reflect a realistic cost of postponing the order. Two ideas are crucial in the setting of the penalty cost in this thesis:

1. An optional order should be serviced if a PSV on a voyage has available capacity and is able to service the order and make it back to the depot by the return time.
2. A spot vessel should not be chartered to service optional orders.

Each optional order is assigned a non-zero penalty cost as described in Algorithm 6. The penalty cost is set as the cost of sailing a fleet vessel in design speed from the depot to the order, servicing the order, and returning to the depot. Chapter 9 provides an analysis of how the penalty costs influence the ALNS solutions.

Algorithm 6: Penalty Cost Calculation

```
initialize the penalty costs of all orders to zero
for o in the set of all optional orders do
    calculate the design speed sailing and servicing cost of order o
    assign the sum of these costs as the penalty cost of order o
end
```

Chapter 8

Computational Study

This chapter contains a computational study of the Adaptive Large Neighborhood Search (ALNS) heuristic's performance when solving instances of the Time-Dependent Vessel Routing Problem with Speed Optimization (TDVRPSO). Section 8.1 describes the test environment used for the computational study and technical details regarding the implementation. Section 8.2 explains how the parameter tuning is done. Section 8.3 compares the heuristic with a Large Neighborhood Search (LNS), tests the local search and Voyage Combination Problem presented in Chapter 6 to determine the best ALNS configuration and studies the convergence of this configuration. Finally, Section 8.4 compares the performance of the heuristic to that of a commercial solver.

8.1 Test Environment

The ALNS, local search extension, and subproblem are implemented in Java 11.0.2. The arc-flow model and model of the Voyage Combination Problem are implemented in Gurobi 9.1.1 using the Python and Java interfaces, respectively. The arc-generation preprocessing procedure of the arc-flow model is implemented in Python 3.8.5. The tests are run on a computing node in the *Solstorm* computing cluster at the Norwegian University of Science and Technology. The computing node is a Lenovo ThinkSystem SD530 computer running version 7 of the operating system Linux CentOS. It has two 3.6GHz Intel Xeon Gold 6244 processors with 8 cores and a 384GB RAM.

According to Equinor, waiting for a solution for more than 600 seconds would not be tolerable given the time pressure the planners face. Hence, the ALNS has two stopping criteria: (1) I^{ALNS} iterations have been performed, or (2) the runtime has reached 600 seconds. The maximum runtime of the commercial solver is set to 3600 seconds. However, for comparability with the ALNS heuristic, the results after 600 seconds are discussed.

Since the ALNS heuristic includes randomness, each test instance is run more than once

to approximate the algorithm's average performance. When presenting the results from running the ALNS heuristic on a test instance, the average results from five runs are presented. When tuning parameters in Section 8.2 five runs are performed per test instance *and* parameter setting. The results from the commercial solver are deterministic, so one run per instance is sufficient.

The subproblem detailed in Chapter 5 is a bottleneck for the runtime of an ALNS iteration. As many of the heuristics in the ALNS algorithm involve solving the subproblem multiple times, parallel computing is utilized in the implementation. The insertions and removals are evaluated in parallel by solving the subproblem for each insertion or removal. Furthermore, when the subproblem is solved for a particular vessel and voyage, the solution is stored in a *cache*. This ensures that the same subproblem is only solved once. Finally, the β described in Section 6.7.2 is set to 20%, requiring the candidate solution to be within 20% of the current best solution for a local search to be performed.

8.2 Parameter Tuning

The parameters used for the ALNS heuristic are presented and motivated in Sections 6.5 and 6.6. A tuning process is performed to determine appropriate values for the parameters. Ten tuning instances are generated as described in Section 7.4 and are only used for tuning purposes. The instances range from 9 to 27 installations, allowing us to test various features of the ALNS heuristic while not exceeding the time limit of 600 seconds.

Table 8.1 lists the systematically tuned parameters used in the ALNS heuristic. $\hat{q}$ represents the percentage of the number of orders in the test instance that is to be removed in each iteration. The adaptive weights are adjusted by the reward parameters σ_1, σ_2 and σ_3, and the reaction parameter r. The amount of noise added to the objective function is decided by the noise control parameter η. For the destroy heuristics worst removal and related removal, randomness is introduced through the determinism parameter p.

Table 8.1: Overview of the systematically tuned parameters in the ALNS heuristic. The parameters are listed in the order they were tuned. The initial value of a parameter is the value that was used before it was tuned.

Parameter	Initial Value	Final Value	Description
$\hat{q}$	$[5\%, 15\%]$	$[15\%, 50\%]$	Percentage of orders to remove, picked uniformly at random in the interval
σ_1	33	33	ALNS score for finding new globally optimal solution
σ_2	9	9	ALNS score for finding new locally optimal solution
σ_3	13	1	ALNS score for finding new solution
r	0.1	0.1	ALNS reaction parameter
η	0.250	0.025	ALNS noise control parameter
p	5	7	Determinism parameter

The ten parameter tuning instances are run five times for all parameter settings. The tuning is performed by only changing one parameter at a time. After tuning a parameter, the tuned value is used when tuning the next parameter. Initial parameter values are set as reported in Table 8.1. These are inspired by Ropke and Pisinger (2006) and Liu et al. (2019) and adjusted using an ad-hoc approach while testing the ALNS heuristic for instances of various sizes. Five different settings are tested for each parameter, and the tuning process is performed once for each parameter. Deciding on which parameters to choose is guided by comparing average objective values and runtimes. See Appendix E for details on the tuning of each parameter and associated results.

From inspecting Table 8.1 we see that several parameters have final values which are different to their initial values. Comparing the final total average objective value with all parameters tuned to the value found when tuning the first parameter, $\hat{q}$, we observe a decrease from 5685.4 to 5670.7, corresponding to a 0.26% decrease. Hence, we see a slight improvement after tuning all parameters but note that the ALNS heuristic can also find good solutions when applying the initial parameters.

The ALNS heuristic also includes parameters that are not systematically tuned, but in most cases, determined by a trial-and-error process when developing the ALNS heuristic. These parameters are presented in Table 8.2.

Table 8.2: Overview of the ALNS parameters that are not systematically tuned.

Parameter	Value	Description
k	3	Regret parameter set for the regret insertion heuristic (*Section 6.5.2*)
κ	0.2	Lower threshold for the adaptive weights.
ξ	$0.2\% \times T_{start}$	Simulated annealing cooling rate.
T_{start}	-	Simulated annealing temperature, set so that the probability of accepting a candidate solution is 50% if the candidate solution is less than 5% worse than the current solution.
I^{ALNS}	5000	Number of iterations for the ALNS heuristic in the parameter tuning process.
I^{S}	100	Number of iterations in one ALNS segment.
I^{VCP}	-	Number of iterations between each time the Voyage Combination Problem (*Section 6.8*) is solved. The parameter is set to 200 for the first five completions, 500 for the next two and then to 1000. Hence, the VCP is solved ten times within 5000 ALNS iterations.
θ	20	Maximum number of consecutive iterations for the same current solution. After θ iterations have passed, a new random solution is constructed and set as the new current solution. (*Section 6.5.2*).

The regret parameter, k, used in the regret insertion heuristic described in Section 6.5.2 is set to 3. The lower threshold of the adaptive weights, κ, is set to 0.2. Further, the parameters ξ and T_{start} must be determined and provided as input to the simulated an-

nealing acceptance criterion. Liu et al. (2019) set the starting temperature, T_{start} so that the probability of accepting a new candidate solution is 50% if the candidate solution is less than 5% worse compared to the current solution. The cooling rate, ξ, is set so that the final temperature in the search is equal to $0.2\% \times T_{start}$. Based on the success of these strategies, the same values are used in this thesis. The maximum number of iterations I^{ALNS} is set to 5000 and the number of segment iterations, I^S, to 50. Initial testing of the VCP demonstrates promising results, and the problem is solved at iterations 200, 400, 600, 800, 1000, 1500, 2000, 3000, 4000, and 5000, amounting to ten times. Hence, the problem is solved more frequently earlier in the search. This is due to trial-and-error testing showing that the best solution is often found within the first 1000 iterations of the ALNS. Finally, for the ALNS heuristic to not get stuck on the same current solution, a maximum number of θ consecutive iterations with the same current solution is set to 20.

8.3 Performance Testing of the ALNS Framework

There are several aspects to consider when evaluating the performance of the ALNS heuristic developed. In Section 8.3.1, we explore the effects of adaptive weights when selecting destroy and repair heuristics in the ALNS heuristic. In Section 8.3.2 we look at the extensions to the ALNS provided by the local search and the VCP, and examine their effects on the performance of the heuristic. In Section 8.3.3, the best configuration of the ALNS heuristic is used when investigating the convergence of the heuristic from the starting point of the solution created by the construction heuristic. Finally, the performance of this configuration is compared to that of a commercial solver in Section 8.4.

In all the proceeding sections, the data presented is for the *instance groups* described in Section 7.4. Each instance in a given instance group is run five times. Hence, the results for each instance group are based on the average of 25 runs for the given ALNS configuration.

8.3.1 Comparing ALNS to LNS

In each iteration of the ALNS heuristic, one destroy and one repair heuristic are chosen. This choice is guided by the use of adaptive weights, as is described in Section 6.6. This section investigates the effects of using weights by comparing the ALNS heuristic to a Large Neighborhood Search (LNS) heuristic. Apart from the use of adaptive weights, the LNS heuristic is set up equivalently to the ALNS. This means that the choice of destroy and repair heuristics in each iteration of the LNS heuristic is random. The ALNS heuristic and the LNS heuristic are run five times on each test instance, each run completing 5000 iterations. The results are presented in Table 8.3. In addition to the average objective value and solution time for all instance groups, the table presents a *gap*. For each instance

in an instance group, we record the best run across all configurations. The gap for each configuration is then defined as the gap between the average of the five runs for the instance and the best run. *Gap* is the average of these gaps across all instances in an instance group. *Gap* is defined equivalently for the remainder of Section 8.3.

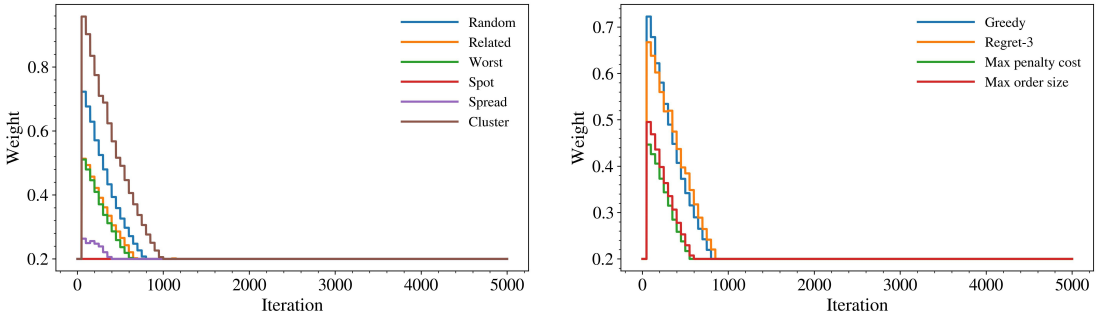
Table 8.3: Comparison of the ALNS and LNS, i.e., of the heuristic with and without the use of adaptive weights for selection of destroy and repair heuristics. The column $\overline{Obj.}$ presents the average objective values of the instance groups. The column *CV* presents the average coefficient of variation for the runs in the instance groups. The column *Gap* presents the average gaps between the best runs across all configurations and the average best solutions produced in each instance group, and the column *Time[s]* specifies the average time for the runs in the instance group in seconds.

Instance	**ALNS** (adaptive weights)				**LNS** (no weights)			
Group	$\overline{\textbf{Obj.}}$	**CV**	**Gap**	**Time[s]**	$\overline{\textbf{Obj.}}$	**CV**	**Gap**	**Time[s]**
5	2217.6	0.00%	0.00%	0.9	2217.6	0.00%	0.00%	0.9
7	2095.0	0.00%	0.00%	2.5	2095.0	0.00%	0.00%	2.5
9	5627.5	0.00%	0.00%	4.5	5627.5	0.00%	0.00%	4.6
11	3517.5	0.00%	0.00%	18.9	3517.5	0.00%	0.00%	18.6
13	3973.4	0.15%	0.07%	32.3	3971.3	0.04%	0.02%	32.1
15	8404.3	0.16%	0.34%	37.2	8405.2	0.19%	0.35%	36.5
17	4974.7	0.21%	0.15%	56.7	4978.5	0.30%	0.23%	57.3
19	5164.1	0.20%	0.21%	70.9	5168.7	0.24%	0.29%	71.4
21	9322.4	0.29%	0.65%	84.8	9321.2	0.44%	0.63%	82.6
23	6001.0	0.45%	0.69%	131.7	5992.7	0.56%	0.55%	129.3
25	6649.0	0.26%	0.29%	141.8	6650.8	0.28%	0.32%	145.3
27	8518.4	0.80%	1.48%	141.8	8509.4	1.02%	1.38%	138.4
Average tot.	5538.7	0.21%	0.44%	61.1	5538.0	0.26%	0.43%	60.0

From the results presented in Table 8.3 we observe that the ALNS heuristic and the LNS choosing destroy and repair heuristics randomly perform similarly. The total average results for the twelve instance groups demonstrate a difference in running time of just 1.6 seconds, with the LNS choosing destroy and repair heuristics randomly yielding a 0.01% better total average objective value. As we have a non-deterministic heuristic and only an insignificant difference in objective value, we cannot conclude that one outperforms the other. In the following, we choose to continue using adaptive weights.

The adaptive weights in the ALNS heuristic are controlled by the heuristic's performance in each ALNS segment, as is explained in Section 6.6. The development of the weights for destroy and repair heuristics, respectively, are exemplified for instance 25-29-4-1 in Figure 8.1. This instance is from the instance group with 25 installations, demonstrating results from a large test instance. Examples from the remaining instance groups are provided in Appendix F. The weights are all initialized to their lower threshold of 0.2. Upon inspection we see that the the shapes of the graphs in Figure 8.1a and 8.1b are somewhat similar. This is explained by the destroy and repair heuristics being rewarded

equally in iterations where they are paired. However, the weights of the destroy and repair heuristics differ due to variations in the pairings of heuristics and the number of times each heuristic is applied in the ALNS heuristic. Observing the shapes of the graphs in Figure 8.1, we see that all destroy and repair heuristics peak within the first 500 iterations. After the heuristics peak, improvements do not happen frequently, and all heuristics drop to the set threshold value within the first 1000 iterations. Hence, a large number of iterations are performed at the lower threshold. Looking back to Table 8.3, this could explain the similarity of the average objective values observed for the *ALNS (adaptive weights)* and the *LNS (no weights)* configurations. Furthermore, based on the test instance presented in this section and the instances in Appendix F, no destroy heuristic performs consistently and significantly better than another. For many instances, the spot vessel removal heuristic is never used, as follows naturally when the spot vessel is not required. Inspecting the repair heuristics used in the search, we make similar observations as for the destroy heuristics. No repair heuristic consistently dominates the search for all instances. Hence, we do not conclude that one destroy or repair heuristic is notably better or worse than all others, but we appreciate that they are all used for applicable instances and result in improvements.



(a) Destroy heuristics **(b)** Repair heuristics

Figure 8.1: Development of the weights associated to the destroy and repair heuristics. The plots represent a run of the test instance 25-29-4-1, and values are sampled at every update of the weights.

8.3.2 Testing of the ALNS Extensions

This thesis presents two primary extensions to the ALNS heuristic. The first is a local search that uses local search operators (LSOs) to discover neighboring solutions that improve the candidate solution found in an ALNS iteration, as is described in Section 6.7. The second extension is the Voyage Combination Problem (VCP) described in Section 6.8. Using voyages from solutions found in previous iterations of the ALNS heuristic, this problem combines voyages and finds new and improving solutions to the TDVRPSO. This section examines the benefits of the aforementioned extensions, separately and in combination. The results of all configurations are presented in Table 8.4. Along with the coefficient of

variation and average runtimes for the various configurations, the results include a *gap* which is defined equivalently to that in Section 8.3.1. Average objective values are not reported, but can be found for the simplest ALNS configuration in the previous section.

Table 8.4: Comparison of results from running the ALNS with its extensions, i.e., local search (LS) and the Voyage Combination Problem (VCP). The column *CV* presents the average coefficient of variation for the runs in the instance groups. The column *Gap* presents the average gaps between the best runs across all configurations and the average best solutions produced in each instance group, and the column *Time[s]* specifies the average time for the runs in the instance group in seconds.

Instance	**ALNS**			**ALNS + LS**			**ALNS + VCP**			**ALNS + LS + VCP**		
Group	**CV**	**Gap**	**Time[s]**	**CV**	**Gap**	**Time[s]**	**CV**	**Gap**	**Time[s]**	**CV**	**Gap**	**Time[s]**
5	0.00%	0.00%	0.9	0.00%	0.00%	1.4	0.00%	0.00%	1.0	0.00%	0.00%	1.6
7	0.00%	0.00%	2.5	0.00%	0.00%	4.1	0.00%	0.00%	1.1	0.00%	0.00%	4.5
9	0.00%	0.00%	4.5	0.00%	0.00%	7.5	0.00%	0.00%	5.5	0.00%	0.00%	8.3
11	0.00%	0.00%	18.9	0.00%	0.00%	42.0	0.00%	0.00%	23.2	0.00%	0.00%	50.3
13	0.15%	0.07%	32.3	0.00%	0.00%	72.2	0.15%	0.07%	47.1	0.00%	0.00%	84.0
15	0.16%	0.34%	37.2	0.02%	0.02%	76.1	0.10%	0.13%	56.7	0.00%	0.00%	91.1
17	0.21%	0.17%	56.7	0.03%	0.01%	149.5	0.19%	0.12%	95.5	0.00%	0.00%	167.7
19	0.20%	0.21%	70.9	0.04%	0.03%	189.6	0.15%	0.11%	132.0	0.00%	0.00%	224.3
21	0.29%	0.80%	84.8	0.20%	0.48%	174.2	0.21%	0.21%	161.2	0.13%	0.19%	195.2
23	0.45%	0.69%	131.7	0.17%	0.16%	333.0	0.28%	0.28%	225.7	0.02%	0.05%	363.9
25	0.26%	0.35%	141.8	0.04%	0.10%	391.5	0.20%	0.19%	273.6	0.03%	0.07%	442.0
27	0.80%	1.81%	141.8	0.58%	0.92%	304.0	0.36%	0.36%	223.8	0.23%	0.31%	387.8
Average tot.	0.23%	0.51%	61.1	0.09%	0.20%	145.4	0.14%	0.17%	104.0	0.03%	0.07%	168.4

After each iteration that produces a candidate solution within 20% of the current best solution, a local search is performed. Investigating the results for ALNS with local search in Table 8.4, we observe an average gap of 0.20%, which is considerably lower than for ALNS without local search (0.51%). The coefficient of variation is also reduced from 0.23% to 0.09%. The average solution time increases from 61.0 seconds to 145.4 seconds, and all instance groups are solved within 600 seconds. Further, we find that the ALNS without local search produces on average 10.3 updates of the current best solution throughout the search. Naturally, these updates are caused solely by destroy/repair perturbations. The ALNS with local search provides an average of 9.5 updates for the instance groups. 91.6% of these updates are caused by the local search. Thus, we see that destroy/repair perturbations are responsible for significantly fewer of the new best solutions after introducing the local search. This can be explained by the local search improving the solutions found by the perturbations, finding high-quality solutions faster, and making new improvements in the next destroy/repair perturbation less probable. When examining which LSOs improve the candidate solutions provided by the destroy/repair perturbations, we observe that the voyage exchange operator performs best. This could be a result of the fleet having vessels with heterogeneous costs. Thus, exchanging voyages could lead to a significant change in

costs, provided that the vessels can handle the load associated with their new voyages. While the standard operators such as intra-voyage relocate and intra-voyage exchange perform well, we observe that the postpone scheduled and schedule postponed operators rarely lead to new best solutions. The first could be attributed to the high penalty costs of postponing already scheduled orders. The latter may be due to a limited number of cases with orders in the set of postponed orders to be scheduled.

Investigating the best solutions found by the ALNS heuristic with local search, we see that the local search finds the best solution in 70% of the instances, while destroy/repair perturbations alone find the best solution in 25% of the instances. In the remaining 5% of the instances, the best solution is found by the construction heuristic. This highlights the value of introducing the local search in the ALNS heuristic. Further, adding the local search leads to greater improvements in average total objective values for the larger instance groups, as is supported by the lower gaps in Table 8.4. By comparing the average objective values to those found by a commercial solver (Table 8.6), we see that optimal solutions for instance groups 5 to 11 are found by both ALNS configurations. Excluding these instance groups, we observe a gap reduction from 0.56% to 0.21% and a 1% improvement in average total objective value when including the local search. Thus, we conclude that the local search significantly improves the ALNS heuristic's performance.

As described in Section 8.2, the VCP is first solved for every 200 iterations of the ALNS heuristic, then every 500 iterations, and finally every 1000 iterations. Considering all instance groups, including the VCP leads to an average gap of 0.17%, which is lower than that of the ALNS heuristic with local search. The coefficient of variation is 0.14%, which is slightly higher than that of the ALNS heuristic with local search, but still considerably lower than for the heuristic without extensions. The average solution time increases from 61.0 seconds for the ALNS heuristic without extensions to 104.0 seconds. The longest average solution time is 273.6 seconds for instance group 25, which is well below the set time limit. Investigating the instance groups 5 to 11 further, we observe that the optimal solution is found within the first 200 iterations for all instances. Like when considering the local search extension, we know these solutions to be optimal by comparing them to those found by a commercial solver. Consequently, the VCP does not find new best solutions for these instances. Investigating instance groups 13 to 27 we observe that solving the VCP results in a new current best solution for an average of 18% of the times it is solved. For these instances, we also have a gap reduction from 0.56% to 0.18%, and a 1% improvement in average total objective value. Based on these findings, we conclude that introducing the VCP improves the ALNS heuristic performance for large, real-sized instances significantly.

Finally, we investigate the configuration of the ALNS heuristic incorporating both the local search and the VCP. When considering this configuration, we find a gap of 0.07% and a coefficient of variation of 0.03%. Both are the lowest across all configurations. Further, we observe that both the coefficient of variation and the gap is 0.00% for instance

groups 5 to 19, demonstrating that it finds the best solution across all configurations for all runs and all instances within these instance groups. The average solution time increases from 61.0 seconds to 168.4 seconds, and all instances are solved within 600 seconds. Improvements in the current best solution throughout the search are found by the destroy/repair perturbations and both extensions. Considering instance groups 13 to 27, i.e., the instance groups for which the best solution is not found before solving the VCP, we observe that on average 70% of the updates of the best solution are caused by the LSOs. Approximately 10% of these updates are found from solving the VCP during the search. The remaining 20% are caused by destroy/repair perturbations. Considering all instance groups, we see that the local search finds the best solution in 70% of the instances, while destroy/repair perturbations alone find the best solution in 25% of the instances. In the remaining 5% of the instances, the construction heuristic finds the best solution. As for the previous configurations, we conclude that adding the local search and the VCP in combination significantly improves the ALNS heuristic's performance.

Based on the findings presented in this section, we observe from the calculated gaps that the ALNS heuristic including both the local search and the VCP provides the best average total objective values. Furthermore, since this configuration also provides the most consistent results in satisfactory solution times, it will be used when investigating convergence and comparing with a commercial solver in the proceeding analyses. Henceforth, this configuration is referred to as the ALNS heuristic.

8.3.3 ALNS Convergence

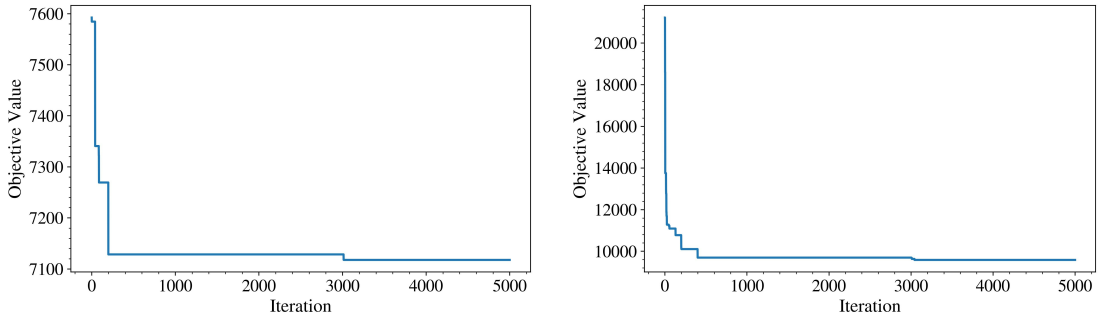
In order to investigate the convergence of the ALNS heuristic towards solutions with satisfactory objective values, the solutions from running the best configuration of the ALNS heuristic are studied and compared to the initial solutions provided by the construction heuristic presented in Section 6.4. These results are presented in Table 8.5. The construction heuristic results are based on one run as it is deterministic. Consequently, its coefficient of variation is 0.00% for all instance groups. The ALNS results are reported based on the average value from running each instance in an instance group five times.

Table 8.5 shows that the ALNS heuristic improves the average objective values for all instance groups significantly. The construction heuristic finds optimal solutions for the instances 5-6-1-1, 7-9-1-1, and 7-9-1-2. For the other instances in instance groups 5 and 7, the construction heuristic produces initial solutions that are close to the optimal solution, as is supported by the small gaps to the best solutions produced. Thus, these instance groups show the smallest average improvement between the construction heuristic and the ALNS heuristic solutions. For the larger instance groups, the construction heuristic is unable to produce initial solutions close to the best solutions found by the ALNS heuristic. This suggests that placing orders greedily is too simple for larger instances.

Table 8.5: Comparison of the construction heuristic and the ALNS heuristic. The column $\overline{Obj.}$ presents the average objective values of the instance groups. The column *CV* presents the average coefficient of variation for the runs in the instance groups. The column *Gap* presents the gaps between the best runs from both configurations and the average best solutions produced in each instance group, and the column *Time[s]* specifies the average time for the runs in the instance group in seconds.

Instance	Construction Heuristic				ALNS + LS + VCP			
Group	$\overline{\text{Obj.}}$	CV	Gap	Time[s]	$\overline{\text{Obj.}}$	CV	Gap	Time[s]
5	2324.8	0.00%	3.84%	0.0	2217.6	0.00%	0.00%	1.6
7	2140.0	0.00%	2.09%	0.0	2095.0	0.00%	0.00%	4.5
9	13335.9	0.00%	46.43%	0.0	5627.5	0.00%	0.00%	8.3
11	4367.3	0.00%	19.03%	0.0	3517.5	0.00%	0.00%	50.3
13	6182.5	0.00%	33.56%	0.0	3970.5	0.15%	0.00%	84.0
15	13152.5	0.00%	36.02%	0.0	8375.4	0.16%	0.00%	91.1
17	7152.6	0.00%	28.42%	0.0	4966.1	0.21%	0.00%	167.7
19	8712.7	0.00%	39.17%	0.0	5153.5	0.20%	0.00%	224.3
21	15413.9	0.00%	37.61%	0.0	9265.8	0.29%	0.10%	195.2
23	8823.0	0.00%	30.28%	0.0	5962.1	0.45%	0.04%	363.9
25	11761.4	0.00%	42.12%	0.0	6630.5	0.26%	0.05%	442.0
27	17177.5	0.00%	44.70%	0.0	8390.0	0.80%	0.25%	387.8
Average tot.	9212.0	0.00%	30.27%	0.0	5514.3	0.21%	0.04%	168.4

Figure 8.2 illustrates the convergence of the best solution's objective value when running the ALNS heuristic on the instances 23-31-4-1 and 27-37-5-1. Observing these plots, we see that most of the improvement occurs in the first iterations of the search. The graph then stabilizes with minor improvements occurring later in the search. This trend is evident for all the test instances presented in Section 7.4. These results imply that operational supply vessels planners may set a more conservative stopping criterion, allowing the algorithm to run fewer iterations, and achieve high-quality solutions in significantly less time.



(a) 23-31-4-1 **(b)** 27-37-5-1

Figure 8.2: Convergence of the best solution's objective value throughout the ALNS heuristic when run on two large test instances. Most improvements from the initial solution happens in the first iterations of the search, with minor improvements later in the search.

8.4 Comparison of the ALNS to a Commercial Solver

This section presents a comparison between the solutions produced by the ALNS heuristic and the commercial optimization solver Gurobi. The solver uses exact methods to solve the mathematical arc-flow model presented in Chapter 4 and is run for 3600 seconds (one hour) on all instances from the instance groups presented in Section 7.4. The results are presented in Table 8.6. In addition to average objective values and runtimes, the table presents gaps. The gap referred to as $Gap^{Obj.}$ is the gap between the average objective value of the instance group found by the ALNS heuristic and that found by the commercial solver. Calculating this gap is done by dividing the difference between the average objective value found by the ALNS heuristic and Gurobi by the average objective value found by Gurobi. Gap^{LB} is the gap between the average objective value of the instance group found by Gurobi and the lower bound found by Gurobi. Calculating this gap is done by dividing the difference between the average objective value found by Gurobi and the lower bound by the average objective value found by Gurobi. As before, we report the average value of the five instances in an instance group for the ALNS heuristic. The commercial solver is also run with a 600 seconds (ten minutes) time limit, and these results are discussed towards the end of the section.

Table 8.6: Comparison of a commercial solver (Gurobi) run for 3600 seconds to the ALNS heuristic. The column $\overline{Obj.}$ presents the average objective value of the instance group. For instance groups in which Gurobi does not find feasible solutions, no objective values are reported. The column Gap^{LB} presents the gap between the objective value and the lower bound provided by Gurobi. The column CV presents the average coefficient of variation for the runs in the instance groups. The column $Gap^{Obj.}$ presents the gap between the best objective value found by Gurobi and the average objective value from the ALNS heuristic in the instance group. The column *Time[s]* specifies the average time for the runs in the instance group in seconds.

Instance	**Gurobi** ($\leq$ 3600 **s**)			**ALNS + LS + VCP**			
Group	$\overline{\textbf{Obj.}}$	$\textbf{Gap}^{LB}$	**Time[s]**	$\overline{\textbf{Obj.}}$	**CV**	$\textbf{Gap}^{Obj.}$	**Time[s]**
5	2217.6	0.00%	5.9	2217.6	0.00%	0.00%	1.6
7	2095.0	0.00%	32.9	2095.0	0.00%	0.00%	4.5
9	5627.5	0.00%	48.5	5627.5	0.00%	0.00%	8.3
11	3517.5	0.00%	2417.2	3517.5	0.00%	0.00%	50.3
13	3994.7	7.87%	3567.7	3970.5	0.00%	-0.60%	84.0
15	12577.9	25.83%	3600.0	8375.5	0.00%	-32.88%	91.1
17	8567.2	61.00%	3600.0	4966.1	0.00%	-42.03%	167.7
19	8988.4	61.13%	3600.0	5153.5	0.00%	-42.66%	224.3
21	13325.4	44.54%	3600.0	9265.8	0.13%	-30.47%	195.2
23	-	-	3600.0	5962.1	0.02%	-	363.9
25	-	-	3600.0	6630.5	0.03%	-	442.0
27	-	-	3600.0	8389.9	0.23%	-	387.8
Average tot.	-	-	2306.02	5514.3	0.03%	-	168.4

The commercial solver is able to reach optimality for instance groups 5 to 11 within 3600 seconds. Within this limit, the instance 13-16-2-1 from instance group 13 is the largest it is able to solve to optimality. Comparing with the values found by the ALNS heuristic, we see that the heuristic finds the optimal solutions for the same instance groups. From instance group 13 and on, the solver does not reach optimality for all instances in any instance group, generating a gap. As previously shown, the ALNS heuristic solves all instances in all instance groups within the time limit of 600 seconds. The coefficient of variation is low for all instance groups, reaching a maximum value of 0.23% for instance group 27. For instance groups 13 to 19, the ALNS heuristic outperforms the solutions found by the commercial solver, resulting in the negative gaps presented in the $Gap^{Obj.}$ column. For instance groups 15 to 19, the gap ranges from -43.80% to -74.42%, demonstrating that the ALNS heuristic produces solutions of substantially higher quality than the commercial solver. For instance groups 23 to 27, the ALNS heuristic continues to produce high-quality solutions where the commercial solver is unable to find any feasible solution.

Considering the commercial solver runs with a 600 seconds time limit, the solver is able to reach optimality for all instances in instance groups 5 to 9. Within this limit, the instance 11-13-2-1 from instance group 11 is the largest instance it solves to optimality. However, most instances from this group are not solved within this limit.

Comparing the ALNS heuristic to the commercial solver, it is clear that where the solver cannot solve real-sized instances of the TDVRPSO, the ALNS heuristic finds high-quality solutions for these instances. Consequently, the heuristic can provide operational support to supply vessel planners for problems of realistic complexities.

Chapter 9

Managerial Insights

Currently, the operational part of the supply vessel voyage planning at Equinor relies on the expertise of experienced planners. Each day the planners adjust the voyages and schedules from a tactical master plan for weather conditions, miscalculations in cargo size estimates, unforeseen orders, and other operational aspects. This chapter demonstrates the operational insights provided by the ALNS heuristic. Section 9.1 visualizes the solution to a real-sized instance of the TDVRPSO, exemplifying the complexity of the problem faced by supply vessel planners. The proceeding sections present new test instances and investigate the value of different aspects of the solutions in more detail. Section 9.2 demonstrates the value of performing speed optimization and taking weather into account. Next, Section 9.3 explains how order selection contributes to forming voyages that better reflect the urgency of the orders. Finally, Section 9.4 shows how considering pickup orders can lead to voyages that can bring more supplies back to the supply depot.

9.1 A Solution to a Real-Sized Instance of the TDVRPSO

The test instance 21-28-3-1 represents a real-sized instance of the TDVRPSO. In this case, the supply vessel planners at the Mongstad supply base are faced with a demand of 28 orders from 21 installations. These orders are comprised of mandatory delivery, optional delivery, and optional pickup orders. The planners have three fleet vessels available but may charter a spot vessel if this is necessary. Applying the ALNS heuristic developed in this thesis, the best solution to the situation faced is the one visualized in Figure 9.1. The solution includes voyages for all three fleet vessels. Further, the ALNS heuristic found chartering a spot vessel necessary to fulfill all mandatory orders. This is due to the capacity shortage of the available fleet vessels. Thus, a schedule for the spot vessel is provided. These voyages also include optimized speeds for all sailing distances, taking the weather conditions and installation opening hours into account. The ALNS heuristic also specifies which orders that should be postponed to the next planning horizon. This

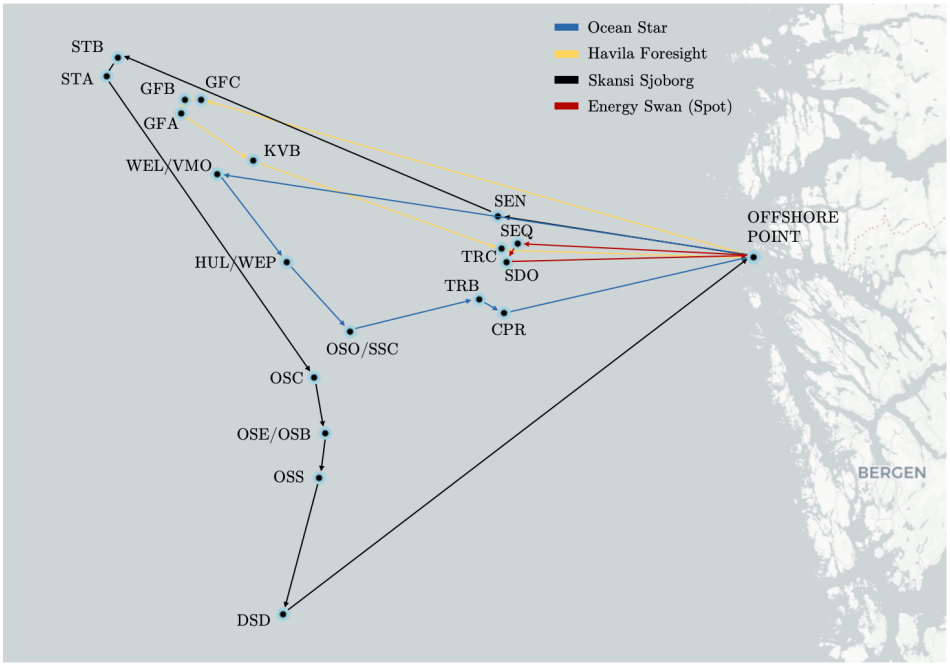


Figure 9.1: A solution to a real-sized instance of the TDVRPSO, including voyages for three fleet vessels and one spot vessel from an offshore point outside the Mongstad base.

solution postpones an optional delivery order from the installation HUL, as Ocean Star does not have enough capacity to include this order on its voyage. Thus, the solution produced can provide the supply vessel planners at Equinor with operational assistance when handling logistical challenges.

As mentioned above, the ALNS heuristic provides the supply vessel planners with optimized speeds and specifies which optional orders to postpone. These two aspects are highlighted in more detail in the following sections.

9.2 Value of Speed Optimization and Weather Dependency

By introducing time-dependent speed optimization, sailing speeds may be adjusted to reflect the weather conditions experienced during a voyage. The monetary costs in the TDVRPSO are associated with fuel consumption and chartering of PSVs from the spot market, both of which may be reduced by addressing disruptions caused by weather. In demonstrating the value of speed optimization and planning based on weather forecasts, we consider a test instance where a single vessel services five orders of standard sizes. Each order is made mandatory and assigned to an installation, as specified in Table 9.1.

Table 9.1: Order composition in the test instance used in this section.

Installations	CPR	SDO	SSC	STC	GFC
Orders	MD	MD	MD	MD	MD

To demonstrate how performing speed optimization and considering weather can reduce

costs, we start by applying the *Perfect* weather scenario (WP) to the test instance. Then, the test instance is solved with and without speed optimization. Without speed optimization, the PSV will only be assigned the fixed design speed. Regardless of whether speed optimization is performed, the voyage in the optimal solution is the one given in Figure 9.2.

Figure 9.2: Optimal voyage for the instance in Table 9.1 with perfect weather conditions.

The speed profiles in Figure 9.3 clarify the difference between the solutions. Since the voyage is unaffected by closed installations due to weather or opening hours, no idling is performed. Disregarding speed optimization, the ALNS suggests that the vessel should sail at 12 knots. However, performing speed optimization shows that the vessel should sail at 10 knots. This is the most fuel-efficient speed, as shown in Figure 7.3.

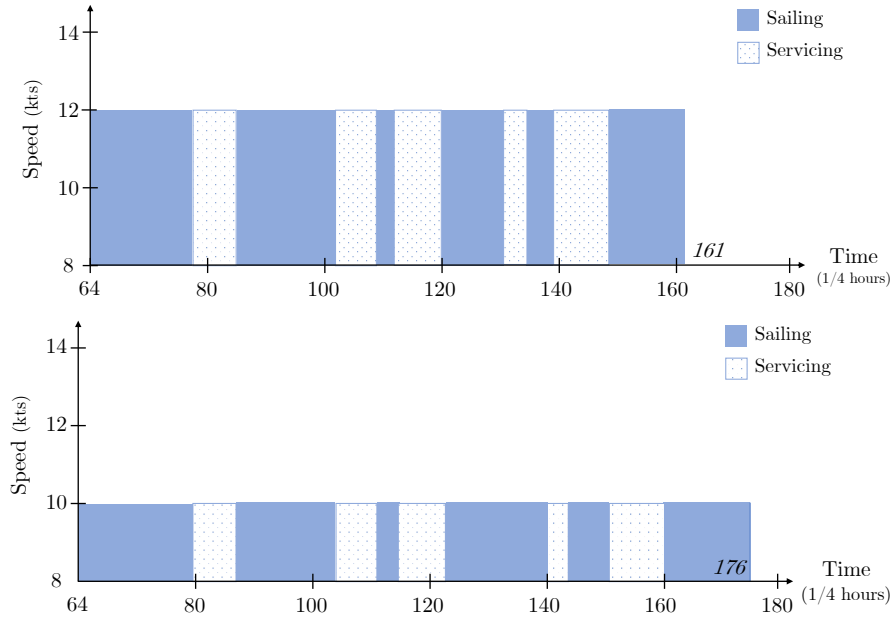


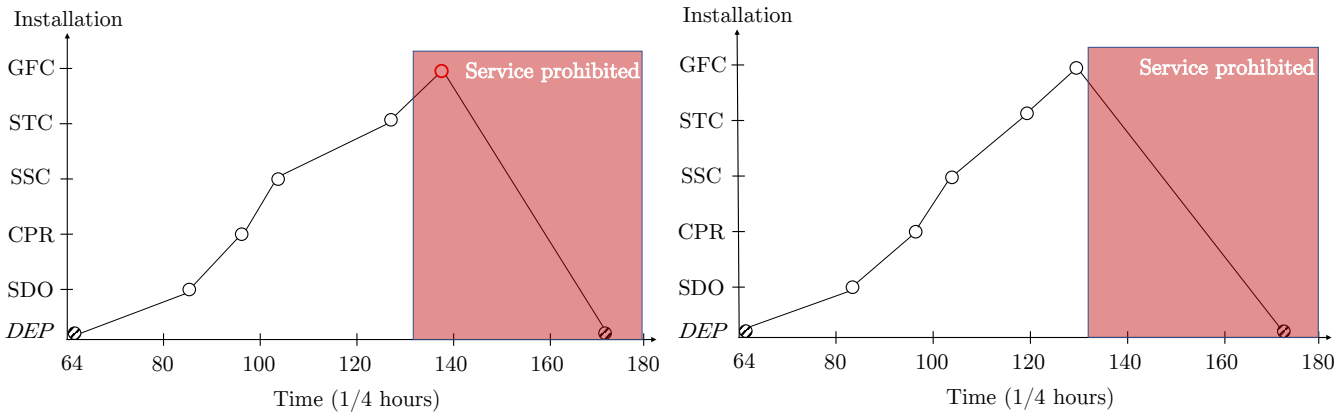
Figure 9.3: Voyage speed profiles using design speed (top) and optimized speeds (bottom) under perfect weather conditions. Performing speed optimization has a clear impact on proposed speeds. Time 64 corresponds to 16:00 on the departure day.

Without performing speed optimization, the sailing speed of the PSV is fixed at the design speed of 12 knots throughout the voyage, resulting in a return time of 161 time units. With speed optimization, the sailing speed is changed to reflect the cost-optimal speed in terms of fuel consumption, still ensuring that the orders are served within the planning horizon. This results in a constant sailing speed of 10 knots and a return time of 176 time units. For this instance, speed optimization results in a cost reduction of 25% and a reduction in fuel consumption of 2869 kg. As CO_2 emissions from crude oil range between 2.940 and 3.212 grams of CO_2 per gram of crude oil (Krantz, 2016), this reduction is equivalent to approximately a 8607 kg decrease in CO_2 emissions.

Furthermore, speed optimization enables planning for disruptive weather conditions. Consider the weather scenario *Critical* (WC) presented in Section 7.3. This weather scenario enters *weather state 3* at an early stage in the planning horizon, and servicing of orders from that point on is prohibited. However, the PSVs may sail in this weather state, so they will be able to return to the depot if they are finished servicing their orders by the time the weather scenario enters *weather state 3*. When applying this scenario to the test instance described at the beginning of this section, the voyage chosen by the PSV changes. The new voyage is visualized in Figure 9.4.

Figure 9.4: Optimal voyage for the instance in Table 9.1 with WC weather conditions.

Applying weather scenario *Critical* (WC) and choosing the PSV's design speed for sailing, servicing at the final installation along the route (i.e., the installation GFC) within the planning horizon will not be possible. Consequently, this will require the chartering of a spot vessel, adding both chartering and fuel consumption costs. This situation is demonstrated in the time-space diagram in Figure 9.5a.



(a) Design speed

(b) Speed optimization

Figure 9.5: Time-space diagram when **(a)** applying design speed and when **(b)** performing speed optimization to the instance in Table 9.1. While the vessel in **(a)** is unable to service the order before GFC closes due to weather, the vessel in **(b)** is able to do so.

Introducing time-dependent speed optimization, the PSV may adjust its speed so that the servicing of the order at installation GFC will be done before the installation closes due to weather. A time-space diagram for this situation is demonstrated in Figure 9.5b.

Since servicing times are identical in both situations and there is no idling at the installations, we see that the average speed used between the installations is faster until the last installation visit when applying speed optimization. Further, note that speed optimization causes the PSV to return to the fuel-minimizing speed after servicing the final order, ensuring that it returns to the depot within the planning horizon. This is further emphasized in the speed profile given in Figure 9.6.

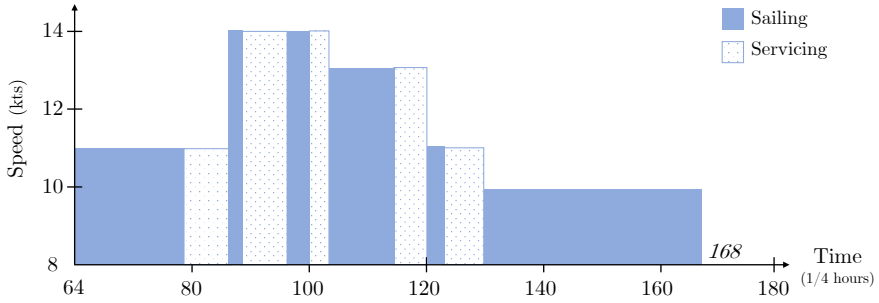


Figure 9.6: Speed profile for the speed-optimized voyage solving the instance in Table 9.1 with weather scenario WC.

Planning without performing time-dependent speed optimization and considering weather forecasts is common to supply vessel planning. This approach avoids the complexities accounted for in this thesis. However, the effects of speed optimization and planning for weather are significant, as is presented in Table 9.2 and Table 9.3, respectively.

To quantify the effects of speed optimization in varying weather conditions, the ALNS heuristic is run on five large test instances from Section 7.4 assigned the different weather scenarios from Section 7.3. For each test instance and weather scenario, the ALNS heur-

istic is run five times with and without speed optimization. Without speed optimization, the vessels are assigned the design speed of 12 knots. Given the average objective values and CO_2 emissions for each instance and weather scenario, the cost, and emission reductions from performing speed optimization are calculated and summarized by Table 9.2. A 20% reduction in costs means that the average objective value from the five ALNS runs was 20% lower when doing speed optimization. Similarly, a 17% reduction in CO_2 emissions means that the average CO_2 emissions were reduced by 17% by performing speed optimization. The average cost and CO_2 emission reductions across all instances and weather scenarios are 19.89% and 19.12%, respectively. These significant reductions imply that speed-optimized voyages can save costs and make vessel operations more sustainable.

Table 9.2: The reductions in costs and CO_2 emissions by performing speed optimization on five of the largest test instances from Section 7.4 in the weather scenarios from Section 7.3. The results indicate that speed optimization cuts both costs and CO_2 emissions compared to planning with the constant design speed of 12 knots.

Instance	**Perfect**		**Mixed**		**Critical**	
	Cost Red.	**CO_2 Red.**	**Cost Red.**	**CO_2 Red.**	**Cost Red.**	**CO_2 Red.**
19-21-3-2	24.01%	18.12%	20.90%	18.78%	18.52%	13.67%
21-24-3-1	22.52%	20.37%	19.18%	17.25%	16.28%	17.81%
23-27-4-1	22.87%	21.77%	19.07%	18.74%	16.32%	22.93%
25-29-4-1	23.63%	24.31%	19.95%	19.03%	17.06%	21.53%
27-32-5-1	22.41%	19.76%	18.90%	15.13%	16.78%	17.57%
Average tot.	23.09%	20.87%	19.60%	17.79%	16.99%	18.70%

To quantify the value of taking weather forecasts into account when performing speed optimization, we study how solutions planned with the *Perfect* weather scenario when the actual weather scenario is *Mixed* or *Critical* are affected by these weather conditions. Falsely assuming *Perfect* weather, solutions will often include voyages servicing orders when the installations are actually closed due to bad weather. They will also underestimate fuel costs, as these increase with the wave height. Therefore, we calculate the *realized* costs of these voyages, i.e., the fuel costs in the actual weather scenario and the costs suffered when vessels attempt to service orders at closed installations. If an order is optional, it does not need to be serviced in the current planning horizon. Thus, the order's penalty cost is calculated as in Section 7.6 and added to the realized costs. If the order is mandatory, the order must be serviced in the current planning horizon. Hence, the fuel costs of idling at the installation until it reopens are added. Note that the calculations of realized costs are based on the assumption that the realized weather is as forecasted.

Table 9.3 compares the realized costs and the number of *missed orders* (MO) of solutions planned with the *Perfect* weather scenario to those planned with the actual weather scenario (*Mixed* or *Critical*). By missed orders, we refer to the orders that vessels attempt to service when the associated installations are closed. As the solutions assuming perfect

weather include voyages arriving at closed installations, the realized costs are high due to many postponed orders and idling while waiting for installations to open. When the ALNS heuristic takes weather into account, the solutions avoid arriving at closed installations by optimizing sailing speeds. This reduces the average estimated costs by 40.06%. Even though the costs presented in this analysis are tentative, they indicate that including weather forecasts in the voyage planning reduces operational costs.

Further, note that the solutions from planning with the *Perfect* weather scenario when the actual weather scenario is *Mixed* or *Critical* are also speed-optimized. However, as the weather scenario is falsely assumed to be *Perfect*, speed optimization yields sailing speeds causing the vessels to arrive at closed installations. Including weather forecasts, speed optimization leads to speeds avoiding this. Hence, it is the *combination* of speed optimization and weather dependency that provides the most value for supply vessel planners.

Table 9.3: The realized costs (**Costs**) in USD and number of missed orders (**#MO**) of solutions to five large test instances when planning with and without weather forecasts. Including weather forecasts in combination with speed optimization yields solutions adapting to the weather conditions in the planning horizon.

Instance	**Mixed (Actual)**				**Critical (Actual)**			
	Perfect (Plan)		**Mixed (Plan)**		**Perfect (Plan)**		**Critical (Plan)**	
	Costs	**#MO**	**Costs**	**#MO**	**Costs)**	**#MO**	**Costs**	**#MO**
19-21-3-2	10602.72	5	6062.91	0	15896.32	5	7035.51	0
21-24-3-1	12740.57	6	6595.69	0	18238.28	6	7648.18	0
23-27-4-1	9951.77	4	7052.73	0	11798.97	4	8100.48	0
25-29-4-1	11599.26	3	8277.93	0	9504.63	1	9445.63	0
27-32-5-1	18084.40	8	10807.54	0	20991.80	5	12308.46	0
Average tot.	12595.74	5.2	7759.36	0	15286.00	4.2	8907.65	0

9.3 Value of Order Selection

Order selection is implemented in the ALNS heuristic to reflect that orders are of different importance and should be prioritized differently in the planning. Prioritization is done by categorizing orders as either optional or mandatory, as explained in Chapter 2. To demonstrate the gained insights from taking order selection into account, two instances where seven installations have one delivery order are considered. In reality, two of these orders are optional. In the first instance, all orders from the installations are considered mandatory. In the second instance, the two optional orders are considered optional. There is only one available vessel from the contracted fleet in both instances. This means that it might not be possible to service all orders with this vessel, either due to time restrictions or vessel capacity. In order to impose stricter time restrictions, weather scenario *Critical* (WC) is applied. Table 9.4 presents the order composition in the two instances.

Table 9.4: Order composition in the test instances used to demonstrate the value of order selection. Instance 2 converts two mandatory orders to optional.

	STB	KVB	VMO	GFC	HUL	CPR	DSD
Instance 1	MD	MD	MD	MD	MD	MD	MD
Instance 2	MD	MD	MD	MD	MD	OD	OD

Running the ALNS heuristic on Instance 1 with one fleet vessel and weather scenario WC yields the optimal voyages visualized in Figure 9.7. As the order from DSD is far away from the orders from the other installations, the fleet vessel is unable to service this order before the installations close due to bad weather. Because the order from DSD is considered mandatory, a spot vessel is chartered. The total costs of the voyages amount to 16 230 USD, of which 9 576 USD are chartering costs. Furthermore, the spot vessel fuel costs amount to 3 029 USD, compared to the fleet vessel fuel costs of 3 625 USD. Hence, 77% of the costs of meeting the demand are from servicing a non-critical order. This is an extreme case, but it demonstrates the value of including order selection in the problem.

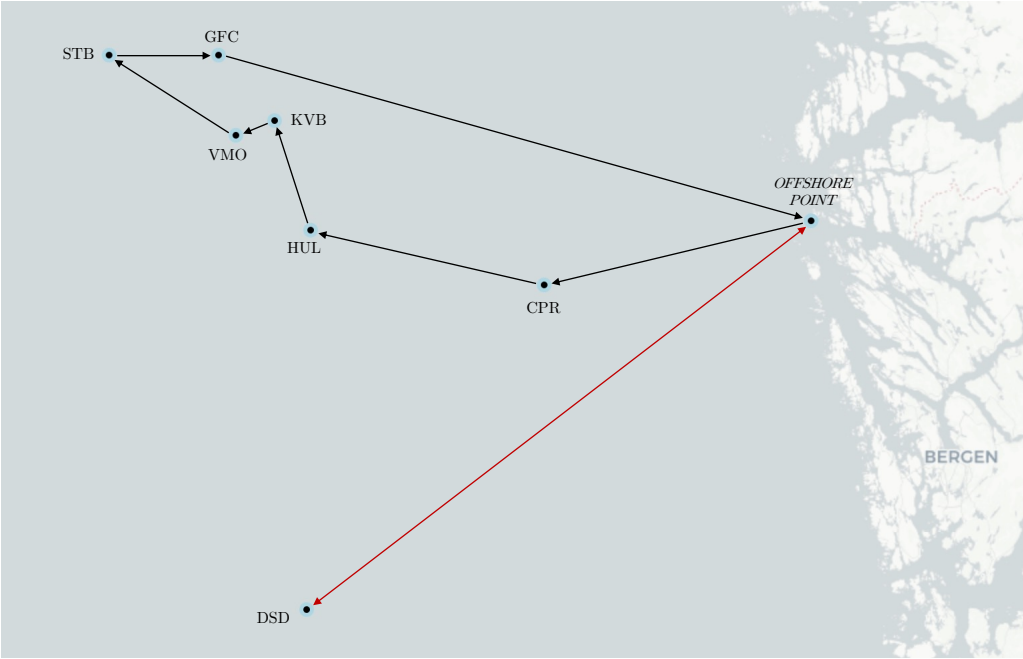


Figure 9.7: Optimal voyages for Instance 1. The red arrow is the spot vessel's voyage. Here, a spot vessel is chartered to service a non-critical order at a high cost.

The optimal voyage resulting from solving Instance 2, where the orders from CPR and DSD are considered optional, is shown in Figure 9.8. The available fleet vessel services all mandatory orders and the optional order from CPR, while the optional order from DSD is postponed. Hence, no chartering is done to service the optional order from DSD. The total fuel costs in this solution are then only 3 625 USD, and there are no chartering costs. However, a penalty cost of 2 305 USD is incurred. As discussed in Section 7.6.2, the penalty cost of an optional order is set to the fuel costs of sailing from the depot to the installation with the order, servicing the installation, and sailing back to the depot.

This is lower than the fuel costs of the spot vessel in Instance 1 because the spot vessel sped up to reduce the chartering costs since these costs increase with time.

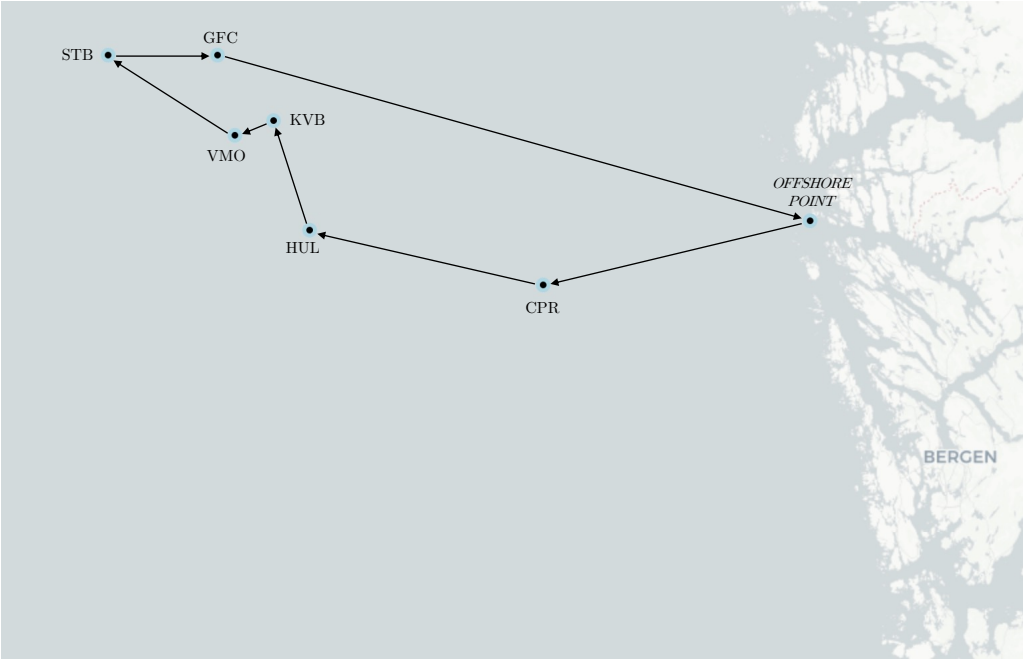


Figure 9.8: Optimal voyages for Instance 2. No spot vessel is chartered to service the optional order from DSD, resulting in saved costs.

The setting of the penalty cost determines how the ALNS heuristic prioritizes orders in the order selection. Correct behavior is defined in Section 7.6.2 as servicing as many optional orders as possible with vessels from the contracted fleet while avoiding chartering spot vessels to service optional orders. The solution for Instance 2 is consistent with this behavior, indicating that the penalty cost setting is sensible. The penalty costs for the optional orders in Instance 2 are presented in Table 9.5. Comparing the penalty cost of DSD's optional order at 2 305 USD to the chartering and fuel costs of servicing this order at 12 605 USD clarifies why postponing the order is more cost-efficient than servicing it.

Table 9.5: Penalty costs of the optional orders of Instance 2. These are set as the average cost of servicing each order separately, as described in Section 7.6.2.

Order	CPR (OD)	DSD (OD)
Penalty cost (USD)	1 227	2 305

Recall that optional orders should be serviced as long as available fleet vessels have the capacity and time to service them. As stated above, the optional order from CPR is serviced by the available fleet vessel in Instance 2. This means that the additional cost of servicing this order must be smaller than the penalty cost of postponing it. To verify this, consider the following calculations: In the optimal solution in Instance 2, the vessel sails at 10 knots between the depot and CPR and at 11 knots between CPR and HUL. Also, the size of the order from CPR is 20 units. Utilizing Equation (7.1) and the constants in

Table 7.1 and Table 7.4 we get the following fuel costs:

- Depot to HUL at 11 knots: 830 USD
- Depot to CPR at 10 knots: 426 USD
- CPR to HUL at 11 knots: 335 USD
- Servicing CPR's order: 94 USD

The additional cost of servicing CPR's order is the difference between sailing costs from the depot to HUL and the sailing and servicing costs from visiting CPR before HUL. This is 25 USD. As the penalty cost of postponing the order is 1 227 USD, the ALNS heuristic favors the solution servicing rather than postponing the order.

Different decisions can be made with different penalty costs. For example, suppose the penalty cost of servicing DSD's order is set to higher than the chartering cost. In that case, the ALNS heuristic favors chartering a spot vessel instead of postponing the order. Alternatively, if the penalty cost of CPR's order is set lower than the additional cost of servicing the order, the ALNS heuristic would favor solutions postponing the order, even though it is cheaper to service the order now than to service the order on a separate voyage. Both these behaviors should be avoided. Setting the penalty cost as the cost of using a fleet vessel to service only this order seems to produce sensible order selection choices.

9.4 Value of Including Pickup Orders

By including pickup orders in the demand from the installations, the voyages will adapt to service these orders. To demonstrate the value of including pickup orders when creating voyages with the ALNS heuristic, a simple instance where two installations have orders is considered. To make the argument as illustrative as possible, we let the installations have unusually large orders. The instance is solved with one available fleet vessel with a capacity of 100 units.

Table 9.6: Order composition in the test instance used to demonstrate the value of pickup. The order sizes are unusually large for illustrative purposes.

Installation	Type	Size
KVB	MD	30
KVB	OP	40
VFB	MD	70

If the ALNS heuristic is run on this instance and the pickup order from KVB is ignored, it finds the voyage in Figure 9.9 as the optimal voyage. In this case, the vessel sails to KVB first to deliver the 30 delivery units and frees up space such that there are 30 units of free

capacity on the vessel deck. Even though the planning process has ignored the demand for pickup, KVB still demands pickup. In reality, the vessel would then try to bring as much of this as possible. However, there would only be room for 30 of the 40 units that need to be picked up.

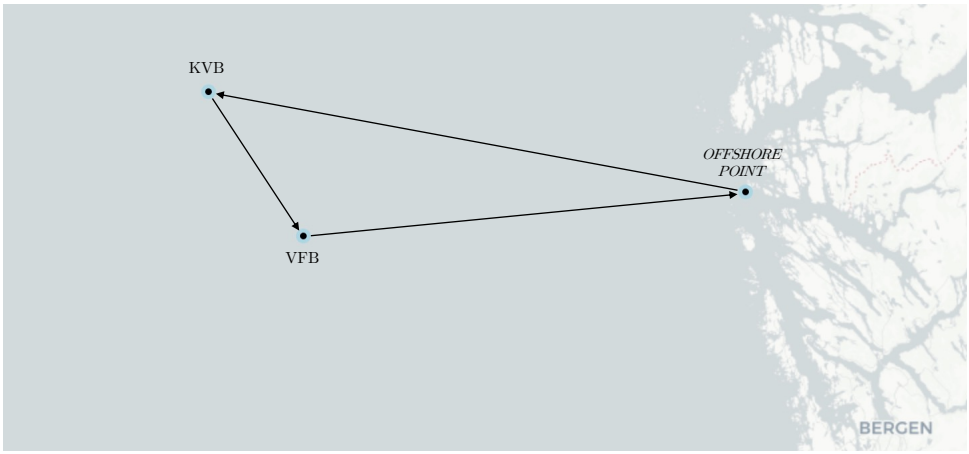


Figure 9.9: Optimal voyage when not considering the pickup order in Table 9.6. As KVB is visited first, not all pickup units can be brought due to capacity restrictions.

Running the ALNS heuristic and including the pickup order in the order selection yields a better solution, shown in Figure 9.10. Here, the vessel sails to VFB first, delivers the 70 units, and sails to KVB to deliver the 30 units. Then, it can pick up all of the 40 pickup units and return to the depot having serviced all orders.

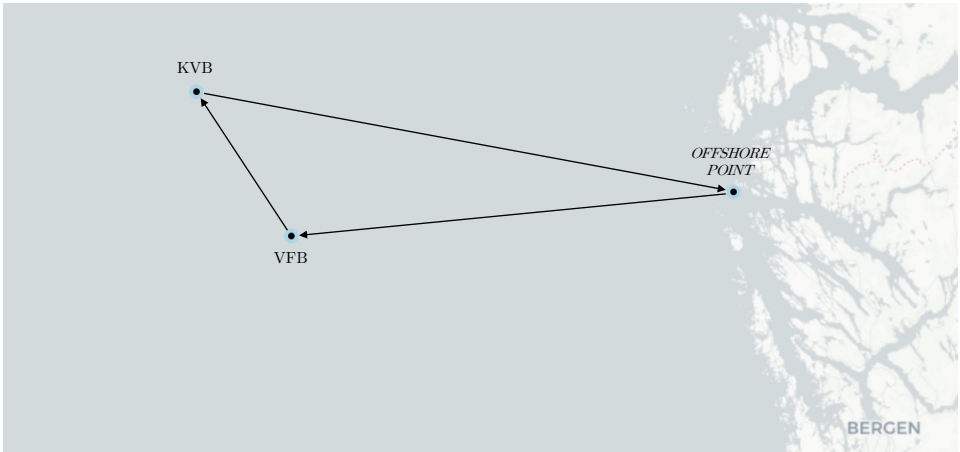


Figure 9.10: Optimal voyage resulting when considering the pickup order in Table 9.6. As VFB is visited first, there is enough capacity to bring back all pickup units from KVB.

Even though this example is simple, it illustrates a bigger picture: By considering pickup orders in the planning, the voyages will be put together in a different way that services more of the actual demand from the installations.

Chapter 10

Concluding Remarks

This thesis has discussed an operational approach to routing platform supply vessels (PSVs) from an onshore depot to offshore installations, servicing orders, and returning to the depot within the planning horizon. The problem is formulated as the Time-Dependent Vessel Routing Problem with Speed Optimization (TDVRPSO), addressing a routing problem with both pickup and delivery orders, order selection, and speed optimization. Supply vessel planning problems are well studied, but mainly from the strategic and tactical levels. Consequently, studies on offshore routing problems from an operational viewpoint are limited, motivating an investigation of operational aspects related to such problems.

Due to tactical plans being vulnerable to variations in weather conditions and order sizes, supply vessel planners have identified the need to address these issues to mitigate the risk of added costs associated with fuel consumption and chartering of spot vessels. These variations are addressed in the TDVRPSO by formulating an operational problem, including time-dependent speed optimization. In order to assist planners in making optimal decisions, orders are prioritized by relative importance. Further, the inclusion of pickup orders leads to better utilization of the capacity of the PSVs, as the voyages are adapted to service more of these orders. The decisions in the problem detail which fleet vessels to use, whether to charter spot vessels, which orders to bring on which vessel, optimal voyages and schedules, and optimal sailing speeds.

Realistic test instances of the TDVRPSO are generated based on a realistic vessel voyage planning case faced by Equinor. Sixty test instances of different complexities are generated and grouped based on their size. The instances are used to evaluate the performance of the proposed solution methods. Since the time the supply vessel planners can wait for a solution from an operational tool is restricted, the running time limit is set to ten minutes.

An arc-flow model is implemented with an exact commercial solver. The model solves all test instances with 13 orders distributed among 11 installations to optimality within one

hour and some with 16 orders distributed among 13 installations. For test instances with 13 to 21 installations, the solver finds solutions with a significant optimality gap. For the largest instances, the solver is not able to find any feasible solutions within one hour. Given the ten-minute time limit, the solver can solve some instances with 13 orders distributed among 11 installations to optimality. As the realistic vessel voyage planning case could involve orders from a total of 27 installations, the exact solution method developed in this thesis is not sufficient as an operational tool for Equinor.

In response to this, an Adaptive Large Neighborhood Search (ALNS) heuristic is implemented. The heuristic presented is based on the work of Ropke and Pisinger (2006) and is adapted to fit the problem-specific extensions relevant to the TDVRPSO. The ALNS heuristic presented incorporates both destroy and repair heuristics introduced by Shaw (1998) Ropke and Pisinger (2006) as well as heuristics first introduced in this thesis. Several of the new heuristics perform well, improving the overall search. Further, a local search for improving the solutions generated by the ALNS is added. The local search operators lead to frequent improvements of the best solution in the search, resulting in better final solutions and faster convergence for the considered test instances. The ALNS heuristic also incorporates a set partitioning problem referred to as the Voyage Combination Problem (VCP) for combining voyages previously encountered in the search. By analyzing the heuristic's performance, the configuration in which both the local search and the VCP are incorporated provides the best results. Finally, to evaluate the quality of the solutions produced by the ALNS heuristic, a subproblem for performing speed optimization and calculating costs of voyages is introduced and solved.

The results generated show that the ALNS heuristic finds solutions that consider the operational aspects of the problem, including the extensions regarding pickup orders and order selection. Performing speed optimization yields qualitatively different voyages at adjusted sailing speeds to adhere to variations in weather. Further, when possible, the sailing speed is reduced to the most fuel-efficient level in order to save costs and lower emissions. Considering the effects of speed optimization in varying weather conditions, we find reductions of up to 24% in both CO_2 emissions and costs. Furthermore, investigating the effects of taking weather forecasts into account when performing speed optimization, operating cost reductions of up to 40% are calculated. Planning for pickup orders, the resulting voyages are put together in a way that services more of the actual demand from the installations. Also, considering order selection, the results demonstrate that when orders were not required to be serviced, and the only alternative is to charter spot vessels, postponing the orders is chosen, consequently avoiding expensive chartering costs.

The ALNS heuristic, including the proposed extensions, provides high-quality solutions for all generated test instances within the ten-minute time limit. Consequently, the heuristic enables us to solve the TDVRPSO for the real case considered. Hence, it shows a clear potential as an operational tool for the supply vessel planners at Equinor.

The work presented in this thesis is focused on addressing the operational challenges in the real-life situation faced by supply vessel planners. The resulting VRP includes time-dependent speed optimization and accounts for selective orders and the possibility of delivering and picking up orders at installations. Consequently, extensions that are often avoided due to their added complexity are addressed. Our contributions may thus be summarized in two main points. Firstly, we introduce essential operational aspects to the routing of platform supply vessels. Secondly, we propose a well-functioning ALNS heuristic providing high-quality solutions in satisfactory time for time-dependent vessel routing problems with mandatory delivery and both optional pickup and optional delivery orders.

Chapter 11

Future Research

Evaluating the work on the TDVRPSO, we identify three main areas for future research, addressing the problem itself, modeling issues, and the use of heuristic solution methods.

First, new extensions to the operational problem faced by supply vessel planners may be investigated. These may include multiple depots, varying departure times from the depot, and multiple visits to installations with associated collision avoidance. Furthermore, the model presented is based on the assumption that unloading bulk cargo is faster than unloading deck cargo. Modeling without this assumption could be of interest. These considerations could make for a more accurate representation of the problem faced by the planners at Equinor and provide additional value.

Second, modeling weather and fuel consumption in more detail than what is done in this thesis may be interesting. As discussed, the presented model is equipped to handle complex models of weather and fuel consumption. A more complex weather model will enable a more accurate estimation of fuel consumption. Furthermore, the fuel consumption can be estimated more precisely by using more complex fuel consumption functions.

Third, improving the performance of the ALNS heuristic presented is of interest. This thesis presents an ALNS heuristic tailored to solve the TDVRPSO and provides satisfactory results for all real-sized test instances. Consequently, the ALNS may work as an operational decision support tool. Nevertheless, improving the performance of a heuristic for solving VRPs with selective pickup and delivery orders and time-dependent speed optimization could be interesting. When surveying and synthesizing heuristics for multi-attribute VRPs, Vidal et al. (2013) highlight the potential of combining a range of methods into hybrid algorithms and parallel cooperative methods, benefiting from different search space exploration techniques. In addition to the ALNS, these methods may include metaheuristic families and hybrid genetic algorithms. As previously indicated, increasing the number of attributes considered in the TDVRPSO is of interest. Consequently, solution methods capable of addressing all these attributes will be appropriate.

Bibliography

Albjerk, N., Danielsen, T., Krey, S., Stålhane, M. & Fagerholt, K. (2016). A vessel pickup and delivery problem from the disruption management in offshore supply vessel operations. *International Conference on Computational Logistics*, 50–64.

Andersson, H., Fagerholt, K. & Hobbesland, K. (2015). Integrated maritime fleet deployment and speed optimization: Case study from roro shipping. *Computers & Operations Research*, *55*, 233–240.

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

Borthen, T., Loennechen, H., Fagerholt, K., Wang, X. & Vidal, T. (2019). Bi-objective offshore supply vessel planning with costs and persistence objectives. *Computers & Operations Research*, *111*, 285–296.

Borthen, T., Loennechen, H., Wang, X., Fagerholt, K. & Vidal, T. (2018). A genetic search-based heuristic for a fleet size and periodic routing problem with application to offshore supply planning. *EURO Journal on Transportation and Logistics*, *7*(2), 121–150.

Bruck, B. P., dos Santos, A. G. & Arroyo, J. E. C. (2012). Hybrid metaheuristic for the single vehicle routing problem with deliveries and selective pickups. *2012 IEEE Congress on Evolutionary Computation*, 1–8.

Christiansen, M., Fagerholt, K., Nygreen, B. & Ronen, D. (2007). Maritime transportation. *Handbooks in operations research and management science*, *14*, 189–284.

Christiansen, M., Fagerholt, K., Rachaniotis, N. P. & Stålhane, M. (2017). Operational planning of routes and schedules for a fleet of fuel supply vessels. *Transportation Research Part E: Logistics and Transportation Review*, *105*, 163–175.

Coelho, I., Munhoz, P. L. A., Haddad, M. N., Souza, M. J. F. & Ochi, L. S. (2012). A hybrid heuristic based on general variable neighborhood search for the single vehicle routing problem with deliveries and selective pickups. *Electronic Notes in Discrete Mathematics*, *39*, 99–106.

Coelho, I., Munhoz, P., Ochi, L., Souza, M., Bentes, C. & Farias, R. (2016). An integrated cpu–gpu heuristic inspired on variable neighbourhood search for the single vehicle routing problem with deliveries and selective pickups. *International Journal of Production Research*, *54*(4), 945–962.

Cuesta, E. F., Andersson, H., Fagerholt, K. & Laporte, G. (2017). Vessel routing with pickups and deliveries: An application to the supply of offshore oil platforms. *Computers & Operations Research*, *79*, 140–147.

Dabia, S., Ropke, S., Van Woensel, T. & De Kok, T. (2013). Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science*, *47*(3), 380–396.

Dijkstra, E. W. et al. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), 269–271.

Equinor. (2020a). *Equinor Mongstad*. Retrieved 10th December 2020, from https://www.equinor.com/en/what-we-do/terminals-and-refineries/mongstad.html

Equinor. (2020b). *Greening our shipping: Replacing the workhorses of the ocean*. Retrieved 10th December 2020, from https://www.equinor.com/en/magazine/greening-our-shipping.html

Equinor. (2020c). *Where we are - Norway*. Retrieved 10th December 2020, from https://www.equinor.com/en/where-we-are/norway.html

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

Fagerholt, K., Laporte, G. & Norstad, I. (2010). Reducing fuel emissions by optimizing speed on shipping routes. *Journal of the Operational Research Society*, *61*(3), 523–529.

Fagerholt, K. & Lindstad, H. (2000). Optimal policies for maintaining a supply service in the Norwegian Sea. *Omega*, *28*(3), 269–275.

Franceschetti, A., Demir, E., Honhon, D., Van Woensel, T., Laporte, G. & Stobbe, M. (2017). A metaheuristic for the time-dependent pollution-routing problem. *European Journal of Operational Research*, *259*(3), 972–991.

Gendreau, M., Hertz, A. & Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, *40*(6), 1086–1094.

Golden Energy Offshore. (2005). *Skansi Sjoborg PSV*. Retrieved 20th May 2021, from https://www.geoff.no/fleet/energy-swan

Green Yard Kleven. (2012). *Skandi Mongstad PSV*. Retrieved 20th May 2021, from https://www.skansi.fo/fleet/sjoborg

Green Yard Kleven. (2014). *Ocean Star PSV*. Retrieved 20th May 2021, from https://www.kleven.no/referansar/ocean-star

Gribkovskaia, I., Halskau sr, Ø., Laporte, G. & Vlček, M. (2007). General solutions to the single vehicle routing problem with pickups and deliveries. *European Journal of Operational Research*, *180*(2), 568–584.

Gribkovskaia, I., Laporte, G. & Shlopak, A. (2008a). A tabu search heuristic for a routing problem arising in servicing of offshore oil and gas platforms. *Journal of the Operational Research Society*, *59*(11), 1449–1459.

Gribkovskaia, I., Laporte, G. & Shyshou, A. (2008b). The single vehicle routing problem with deliveries and selective pickups. *Computers & Operations Research*, *35*(9), 2908–2924.

Gutiérrez-Jarpa, G., Desaulniers, G., Laporte, G. & Marianov, V. (2010). A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows. *European Journal of Operational Research*, *206*(2), 341–349.

Gutiérrez-Jarpa, G., Marianov, V. & Obreque, C. (2009). A single vehicle routing problem with fixed delivery and optional collections. *IIE Transactions*, *41*(12), 1067–1079.

HACON Containers. (2020). *10ft DNV 2.7-1 Offshore container*. Retrieved 13th December 2020, from https://hacon-containers.com/containers/10ft-dnv-2-7-1-offshore-container

Halvorsen-Weare, E. E. & Fagerholt, K. (2011). Robust supply vessel planning. *International Conference on Network Optimization*, 559–573.

Halvorsen-Weare, E. E. & Fagerholt, K. (2017). Optimization in offshore supply vessel planning. *Optimization and Engineering*, *18*(1), 317–341.

Halvorsen-Weare, E. E., Fagerholt, K., Nonås, L. M. & Asbjørnslett, B. E. (2012). Optimal fleet composition and periodic routing of offshore supply vessels. *European Journal of Operational Research*, *223*(2), 508–517.

Havila Shipping. (2007). *Havila Foresight PSV*. Retrieved 20th May 2021, from https://www.havilashipping.no/fleet/psv/havila-foresight

Homsi, G., Martinelli, R., Vidal, T. & Fagerholt, K. (2020). Industrial and tramp ship routing problems: Closing the gap for real-scale instances. *European Journal of Operational Research*, *283*(3), 972–990.

Ichoua, S., Gendreau, M. & Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European journal of operational research*, *144*(2), 379–396.

Jahre, M., Persson, G., Aas, B., Gribkovskaia, I., Halskau, Ø. & Shlopak, A. (2007). Routing of supply vessels to petroleum installations. *International Journal of Physical Distribution & Logistics Management*.

Kisialiou, Y., Gribkovskaia, I. & Laporte, G. (2018a). The periodic supply vessel planning problem with flexible departure times and coupled vessels. *Computers & Operations Research*, *94*, 52–64.

Kisialiou, Y., Gribkovskaia, I. & Laporte, G. (2018b). Robust supply vessel routing and scheduling. *Transportation Research Part C: Emerging Technologies*, *90*, 366–378.

Kisialiou, Y., Gribkovskaia, I. & Laporte, G. (2019). Supply vessel routing and scheduling under uncertain demand. *Transportation Research Part C: Emerging Technologies*, *104*, 305–316.

Korsvik, J. E., Fagerholt, K. & Laporte, G. (2011). A large neighbourhood search heuristic for ship routing and scheduling with split loads. *Computers & Operations Research*, *38*(2), 474–483.

Krantz, G. (2016). *CO_2 and sulphur emissions from the shipping industry*. Retrieved 11th December 2020, from https://www.egcsa.com/wp-content/uploads/CO2-and-sulphur-emissions-from-the-shipping-industry.pdf

Lindstad, H., Asbjørnslett, B. E. & Jullumstrø, E. (2013). Assessment of profit, cost and emissions by varying speed as a function of sea conditions and freight market. *Transportation Research Part D: Transport and Environment*, *19*, 5–12.

Liu, R., Tao, Y. & Xie, X. (2019). An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. *Computers & Operations Research*, *101*, 250–262.

Ma, Z.-J., Wu, Y. & Dai, Y. (2017). A combined order selection and time-dependent vehicle routing problem with time widows for perishable product delivery. *Computers & Industrial Engineering*, *114*, 101–113.

Maisiuk, Y. & Gribkovskaia, I. (2014). Fleet sizing for offshore supply vessels with stochastic sailing and service times. *Procedia Computer Science*, *31*, 939–948.

Malandraki, C. & Daskin, M. S. (1992). Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, *26*(3), 185–200.

Moan, A. & Ødeskaug, P. (2020). *A hybrid genetic approach to the operational supply vessel planning problem with speed optimization* (Master's thesis). Norwegian University of Science and Technology.

Norlund, E. K. & Gribkovskaia, I. (2013). Reducing emissions through speed optimization in supply vessel operations. *Transportation Research Part D: Transport and Environment*, *23*, 105–113.

Norlund, E. K. & Gribkovskaia, I. (2017). Environmental performance of speed optimization strategies in offshore supply vessel planning under weather uncertainty. *Transportation Research Part D: Transport and Environment*, *57*, 10–22.

Norlund, E. K., Gribkovskaia, I. & Laporte, G. (2015). Supply vessel planning under cost, environment and robustness considerations. *Omega*, *57*, 271–281.

Norstad, I., Fagerholt, K. & Laporte, G. (2011). Tramp ship routing and scheduling with speed optimization. *Transportation Research Part C: Emerging Technologies*, *19*(5), 853–865.

Norwegian government. (2020). *Norway's oil history in 5 minutes*. Retrieved 10th December 2020, from https://www.regjeringen.no/en/topics/energy/oil-and-gas/norways-oil-history-in-5-minutes/id440538/

Psaraftis, H. N. & Kontovas, C. A. (2014). Ship speed optimization: Concepts, models and combined speed-routing scenarios. *Transportation Research Part C: Emerging Technologies*, *44*, 52–69.

Ropke, S. & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *International conference on principles and practice of constraint programming*, 417–431.

Shyshou, A., Gribkovskaia, I., Laporte, G. & Fagerholt, K. (2012). A large neighbourhood search heuristic for a periodic supply vessel planning problem arising in offshore oil and gas operations. *INFOR: Information Systems and Operational Research*, *50*(4), 195–204.

Skansi Offshore. (2012). *Skansi Sjoborg PSV*. Retrieved 20th May 2021, from https://www.skansi.fo/fleet/sjoborg

Sopot, E. & Gribkovskaia, I. (2014). Routing of supply vessels to with deliveries and pickups of multiple commodities. *Procedia Computer Science*, *31*, 910–917.

Stålhane, M., Albjerk, N., Danielsen, T., Krey, S. & Fagerholt, K. (2019). A variable neighbourhood search heuristic for disruption management in offshore oil and gas logistics. *Journal of the Operational Research Society*, *70*(4), 588–600.

Süral, H. & Bookbinder, J. H. (2003). The single-vehicle routing problem with unrestricted backhauls. *Networks: An International Journal*, *41*(3), 127–136.

The J.J. Ugland Companies. (2014). *Mv Juanita PSV*. Retrieved 20th May 2021, from https://www.jjuc.no/bilder/Fleets/Offshore%20service%20vessels/Dokumenter/JUANITA%20BROCHURE%20-%20Sept%202020.pdf

Ting, C.-K. & Liao, X.-L. (2013). The selective pickup and delivery problem: Formulation and a memetic algorithm. *International Journal of Production Economics*, *141*(1), 199–211.

Ulsrud, K. P. & Vandvik, A. H. (2020). *The time-dependet vessel routing problem with speed optimization*. Norwegian University of Science and Technology.

Vidal, T., Crainic, T. G., Gendreau, M. & Prins, C. (2013). Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, *231*(1), 1–21.

Vroon Offshore Services. (2014). *Vos Purpose PSV*. Retrieved 20th May 2021, from https://www.vroon.nl/Files/VesselParticulars/VOS%20PURPOSE20210505041523.pdf

Appendix A

Mathematical Model

Sets and Indices

$\mathcal{V}$	-	set of available vessels v
$\mathcal{N}$	-	set of orders i
$\mathcal{N}^{MD}$	-	subset of $\mathcal{N}$ consisting of mandatory delivery orders
$\mathcal{N}^{OD}$	-	subset of $\mathcal{N}$ consisting of optional delivery orders
$\mathcal{N}^{OP}$	-	subset of $\mathcal{N}$ consisting of optional pickup orders
$\mathcal{A}_v$	-	set of arcs $((i,\ t),\ (j,\ t'))$ for vessel v
$\mathcal{T}$	-	set of all discrete time points, t, in the planning horizon
$\mathcal{T}^{S}_{ijv}$	-	subset of $\mathcal{T}$ with start times for arcs between i and j for vessel v
$\mathcal{T}^{SS}_{ijtv}$	-	subset of $\mathcal{T}$ with specific start times for arcs between i and j with end time t for vessel v
$\mathcal{T}^{SE}_{itjv}$	-	subset of $\mathcal{T}$ with specific end times for arcs between i and j with start time t for vessel v

Parameters

S_i	-	size of order i
Q_v	-	maximum load capacity of vessel v
$C^A_{itjt'v}$		fuel consumption and potential chartering cost for vessel v on arc $((i,\ t),\ (j,\ t'))$
C^P_i	-	penalty cost of not servicing optional order i
o	-	supply depot at the beginning of a voyage, modeled as an origin node
d	-	supply depot at the end of a voyage, modeled as a destination node
t^*	-	time at which vessel preparation ends

Decision Variables

$x_{itjt'v}$ - $\begin{cases} 1, & \text{if arc } \{(i,t),(j,t')\} \text{ is used by vessel } v \\ 0, & \text{otherwise} \end{cases}$

u_{iv} - $\begin{cases} 1, & \text{if order } i \text{ is serviced by vessel } v \\ 0, & \text{otherwise} \end{cases}$

l^D_{iv} - delivery load for vessel v after servicing order i

l^P_{iv} - pickup load for vessel v after servicing order i

Objective Function

$$\min \sum_{v\in\mathcal{V}} \sum_{\{(i,t),(j,t')\}\in\mathcal{A}_v} C^A_{itjt'v} x_{itjt'v} + \sum_{i\in\mathcal{N}^{OD}\cup\mathcal{N}^{OP}} C^P_i (1 - \sum_{v\in\mathcal{V}} u_{iv}) \tag{A.1}$$

Constraints

$$\sum_{j\in\mathcal{N}} \sum_{t'\in\mathcal{T}^{SS}_{ijtv}} x_{jt'itv} - \sum_{j\in\mathcal{N}} \sum_{t'\in\mathcal{T}^{SE}_{itjv}} x_{itjt'v} = 0, \quad i\in\mathcal{N}, t\in\mathcal{T}, v\in\mathcal{V} \tag{A.2}$$

$$\sum_{j\in\mathcal{N}} \sum_{t'\in\mathcal{T}^{SE}_{otjv}} x_{ot^*jt'v} = 1, \quad v\in\mathcal{V} \tag{A.3}$$

$$\sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}^S_{ijv}}\sum_{t'\in\mathcal{T}^{SE}_{itdv}} x_{itdt'v}=1,\quad v\in\mathcal{V} \tag{A.4}$$

$$\sum_{v\in\mathcal{V}} u_{iv}=1,\quad i\in\mathcal{N}^{MD} \tag{A.5}$$

$$\sum_{v\in\mathcal{V}} u_{iv}\leq 1,\quad i\in\mathcal{N}^{OD}\cup\mathcal{N}^{OP} \tag{A.6}$$

$$\sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}^S_{ijv}}\sum_{t'\in\mathcal{T}^{SE}_{itjv}}\sum_{v\in\mathcal{V}} x_{itjt'v}=\sum_{v\in\mathcal{V}} u_{jv},\quad j\in\mathcal{N} \tag{A.7}$$

$$\sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}^S_{ijv}}\sum_{t'\in\mathcal{T}^{SE}_{itjv}} x_{itjt'v}=u_{jv},\quad j\in\mathcal{N},v\in\mathcal{V} \tag{A.8}$$

$$l^D_{ov}=\sum_{i\in\mathcal{N}^{MD}\cup\mathcal{N}^{OD}} S_i u_{iv},\quad v\in\mathcal{V} \tag{A.9}$$

$$l^D_{iv}+l^P_{iv}\leq Q_v u_{iv},\quad i\in\mathcal{N},v\in\mathcal{V} \tag{A.10}$$

$$l^D_{jv}\leq l^D_{iv}-S_j u_{jv}+Q_v\left(1-\sum_{t\in\mathcal{T}^S_{ijv}}\sum_{t'\in\mathcal{T}^{SE}_{itjv}} x_{itjt'v}\right),\quad i\in\mathcal{N},j\in\mathcal{N}^{MD}\cup\mathcal{N}^{OD},v\in\mathcal{V} \tag{A.11}$$

$$l^D_{jv}\leq l^D_{iv}+Q_v\left(1-\sum_{t\in\mathcal{T}^S_{ijv}}\sum_{t'\in\mathcal{T}^{SE}_{itjv}} x_{itjt'v}\right),\quad i\in\mathcal{N},j\in\mathcal{N}^{OP},v\in\mathcal{V} \tag{A.12}$$

$$l^P_{jv}\geq l^P_{iv}+S_j u_{jv}-Q_v\left(1-\sum_{t\in\mathcal{T}^S_{ijv}}\sum_{t'\in\mathcal{T}^{SE}_{itjv}} x_{itjt'v}\right),\quad i\in\mathcal{N},j\in\mathcal{N}^{OP},v\in\mathcal{V} \tag{A.13}$$

$$l^P_{jv}\geq l^P_{iv}-Q_v\left(1-\sum_{t\in\mathcal{T}^S_{ijv}}\sum_{t'\in\mathcal{T}^{SE}_{itjv}} x_{itjt'v}\right),\quad i\in\mathcal{N},j\in\mathcal{N}^{MD}\cup\mathcal{N}^{OD},v\in\mathcal{V} \tag{A.14}$$

$$l_{dv}^{P} = \sum_{i \in \mathcal{N}^{OP}} S_i u_{iv}, \quad v \in \mathcal{V} \tag{A.15}$$

$$x_{itjt'v} \in \{0,1\}, \quad \{(i,t),(j,t')\} \in \mathcal{A}_v, v \in \mathcal{V} \tag{A.16}$$

$$u_{iv} \in \{0,1\}, \quad i \in \mathcal{N} \cup \{o,d\}, v \in \mathcal{V} \tag{A.17}$$

$$l_{iv}^{D} \geq 0, \quad i \in \mathcal{N} \cup \{o,d\}, v \in \mathcal{V} \tag{A.18}$$

$$l_{iv}^{P} \geq 0, \quad i \in \mathcal{N} \cup \{o,d\}, V \in \mathcal{V} \tag{A.19}$$

Appendix B

Data from the Mongstad Case

Table B.1 shows the location, opening hours, and standard order size associated with each installation. The location is given as geographical coordinates, opening hours as the hours each installation opens and closes, and standard order size as the size in cargo units (HACON Containers, 2020) of a typical order from the installation. The standard order size for an installation applies to all three types of orders considered.

#	Installation	Latitude	Longitude	Opening hours	Standard order size (MD-OD-OP)
1	TRO (Troll A)	60,64	3,72	07-19	27-14-28
2	TRB (Troll B)	60,77	3,50	07-19	20-10-21
3	TRC (Troll C)	60,88	3,60	07-19	14-7-14
4	CPR (Cosl Promoter)	60,74	3,61	00-24	22-11-23
5	SEN (Songa Endurance)	60,95	3,58	00-24	23-12-24
6	SDO (Stena Don)	60,85	3,62	00-24	18-9-19
7	SEQ (Songa Equinox)	60,89	3,67	00-24	23-12-24
8	OSE (Oseberg A/D)	60,48	2,82	00-24	27-14-28
9	OSB (Oseberg B)	60,48	2,82	00-24	14-7-14
10	OSC (Oseberg C)	60,60	2,77	00-24	14-7-14
11	OSO (Oseberg Øst)	60,70	2,93	00-24	20-10-21
12	SSC (Safe Scandinavia)	60,70	2,93	00-24	17-9-18
13	OSS (Oseberg Sør)	60,38	2,79	00-24	20-10-21
14	DSD (Songa Delta)	60,08	2,63	00-24	15-8-16
15	KVB (Kvitebjørn)	61,07	2,50	00-24	27-14-28
16	VMO (Valemon)	61,04	2,34	00-24	20-10-21
17	WEL (West Elara)	61,04	2,34	00-24	22-11-23
18	VFB (Veslefrikk B)	60,78	2,89	00-24	27-14-28
19	WEP (West Epsilon)	60,85	2,65	00-24	22-11-23
20	HUL (Huldra)	60,85	2,65	00-24	20-10-21
21	STA (Statfjord A)	61,25	1,85	07-19	20-10-21
22	STB (Statfjord B)	61,20	1,82	00-24	20-10-21
23	STC (Statfjord C)	61,29	1,90	00-24	27-14-28
24	GFA (Gullfaks A)	61,17	2,18	00-24	20-10-21
25	GFB (Gullfaks B)	61,20	2,20	00-24	20-10-21
26	GFC (Gullfaks C)	61,20	2,27	00-24	27-14-28
27	SOD (Songa Dee)	60,90	3,81	00-24	22-11-23

Table B.1: Data associated with the installations in the Mongstad case.

Appendix C

Vessels

In this thesis, six PSVs are used to set the vessel capacity and fuel consumption parameters realistically. Five of the PSVs currently have long-term contracts with Equinor. These are used to set the parameters of the five *fleet vessels*. The sixth vessel is a vessel from the North Sea spot market. This vessel is used to set the parameters of the *spot vessel*. Table C.1 shows the capacities measured in cargo units and the fuel consumption in design speed measured in kg/hour. The capacities are retrieved from the sources listed in the table. The fuel consumption parameters are estimated based on the design speed fuel consumption of a vessel built in 2014 (Vroon Offshore Services, 2014). This vessel has a fuel consumption in design speed of 540 kg/hour, so the vessels in Table C.1 built in 2014 are assigned this fuel consumption. In order to get the values for the vessels with later build years, the consumption increases by 2% per build year.

PSV	**Fleet / Spot**	**Capacity**	**Fuel Consumption** s_0
Ocean Star (Green Yard Kleven, 2014)	Fleet	131	540
Havila Foresight (Havila Shipping, 2007)	Fleet	130	620
Skansi Sjoborg (Skansi Offshore, 2012)	Fleet	125	562
Skandi Mongstad (Green Yard Kleven, 2012)	Fleet	128	608
MV Juanita (The J.J. Ugland Companies, 2014)	Fleet	127	540
Energy Swan (Golden Energy Offshore, 2005)	Spot	130	645

Table C.1: Six PSVs, five with long-term contracts and one spot vessel, used to set the capacity and design speed fuel consumption parameters used in this thesis. The capacities are measured in cargo units, where one cargo unit occupies eight square meters on deck (HACON Containers, 2020). The fuel consumption is measured in kilograms per hour.

Appendix D

Test Instances

Table D.1 shows a full overview of the test instances that are used for testing the performances of the solution methods presented in this thesis. Results from the performance testing are presented in the computational study in Chapters 8 and 9.

Table D.1: A full overview of the test instances used for performance testing in the computational study in Chapters 8 and 9.

Instance	Installations	Orders	MD	OD	OP	Fleet vessels
5-5-1-1	5	5	3	1	1	1
5-5-1-2	5	5	3	1	1	1
5-6-1-1	5	6	3	2	1	1
5-7-1-1	5	7	3	2	2	1
5-7-1-2	5	7	3	2	2	1
7-8-1-1	7	8	4	2	2	1
7-8-1-2	7	8	4	2	2	1
7-8-1-3	7	8	4	2	2	1
7-9-1-1	7	9	5	2	2	1
7-9-1-2	7	9	5	2	2	1
9-9-1-1	9	9	5	2	2	1
9-9-1-2	9	9	5	2	2	1
9-10-1-1	9	10	5	3	2	1
9-11-1-1	9	11	6	2	3	1
9-11-1-2	9	11	6	2	3	1
11-12-2-1	11	12	6	3	3	2
11-12-2-2	11	12	6	3	3	2
11-13-2-1	11	13	7	3	3	2
11-14-2-1	11	14	7	4	3	2
11-15-2-1	11	15	7	4	4	2
13-14-2-1	13	15	8	3	3	2
13-15-2-1	13	15	7	4	4	2
13-15-2-2	13	15	7	3	5	2
13-16-2-1	13	16	8	4	4	2
13-18-2-1	13	18	8	5	5	2
15-15-2-1	15	15	8	4	3	2
15-17-2-1	15	17	11	3	3	2
15-18-2-1	15	18	11	4	3	2
15-18-2-2	15	18	9	5	4	2
15-21-2-1	15	21	9	6	6	2
17-18-3-1	17	18	9	4	5	3
17-19-3-1	19	19	11	4	4	3
17-19-3-2	19	19	9	5	5	3
17-22-3-1	19	22	11	5	6	3
17-23-3-1	19	23	11	6	6	3
19-19-3-1	19	19	10	4	5	3
19-21-3-1	19	21	10	5	6	3
19-21-3-2	19	21	10	4	7	3
19-22-3-1	21	22	10	5	7	3
19-25-3-1	21	25	12	6	7	3
21-23-3-1	21	23	12	6	5	3
21-23-3-2	21	23	11	5	7	3
21-24-3-1	21	24	13	6	5	3
21-26-3-1	21	26	15	5	6	3
21-28-3-1	21	28	15	6	7	3
23-24-4-1	23	24	12	5	7	4
23-25-4-1	23	25	12	6	7	4
23-27-4-1	23	27	13	5	9	4
23-30-4-1	23	30	12	9	9	4
23-31-4-1	23	31	16	9	6	4
25-26-4-1	25	26	13	6	7	4
25-28-4-1	25	28	15	6	7	4
25-29-4-1	25	29	13	9	7	4
25-30-4-1	25	30	13	8	9	4
25-33-4-1	25	33	16	8	9	4
27-28-5-1	27	28	15	7	6	5
27-29-5-1	27	29	15	6	8	5
27-32-5-1	27	32	16	9	7	5
27-35-5-1	27	35	18	8	9	5
27-37-5-1	27	37	18	9	10	5

Appendix E

Parameter Calibration

E.1 Removal Parameter

The removal parameter q^{ALNS} determines the number of orders removed in each iteration of the ALNS when this is specified for the destroy heuristic in question. The removal processes are described in Section 6.5.1. Inspired by Liu et al. (2019) we set a control parameter $\hat{q}$ and define $q^{ALNS} = \lfloor \hat{q} \times |\mathcal{N}| \rfloor$. $\hat{q}$ denotes the percentage of orders to remove in each iteration, and $|\mathcal{N}|$ the number of tasks. Setting the value of $\hat{q}$ presents a trade-off; a small value only allows the algorithm to move in proximity to the current solution, possibly getting stuck in a local optimum (Ropke and Pisinger, 2006); a large value can be tedious and may expose limitations in the repair heuristics due to new solutions being similar to those previously obtained. Similar to what is done by Liu et al. (2019) we test five intervals for $\hat{q}$ in order to enable a sensitivity analysis. These intervals are $[5\%, 15\%]$, $[15\%, 30\%]$, $[5\%, 30\%]$, $[15\%, 50\%]$ and $[30\%, 50\%]$. $\hat{q}$ is picked uniformly at random in each iteration. The results are reported in Table E.1.

The objective values, times, and the number of iterations included in Table E.1 are averages from running the various instances. Investigating the results, we see that the average time consistently increases when increasing $\hat{q}$. Looking at the total average objective values, $[5\%, 15\%]$ shows the worst value. The remaining intervals show similar values, with $[15\%, 50\%]$ providing the best out of the four. As the running times of all intervals are fair, we select $[15\%, 50\%]$ due to its superior objective value. Consequently, $\hat{q}$ is chosen randomly from the interval $[15\%, 50\%]$ for each iteration of each run of the ALNS heuristic.

Instance	$\hat{\mathbf{q}} \in [5\%, 15\%]$			$\hat{\mathbf{q}} \in [5\%, 30\%]$			$\hat{\mathbf{q}} \in [15\%, 30\%]$			$\hat{\mathbf{q}} \in [15\%, 50\%]$			$\hat{\mathbf{q}} \in [30\%, 50\%]$		
	Obj.	**Time[s]**	**Iter.**	**Obj.**	**Time[s]**	**Iter.**	**Obj.**	**Time[s]**	**Iter.**	**Obj.**	**Time [s]**	**Iter.**	**Obj.**	**Time[s]**	**Iter.**
9-9-1-1	3144.5	5.4	5000	3144.5	6.3	5000	3144.5	6.6	5000	3144.5	8.3	5000	3144.5	8.9	5000
11-13-1-1	7999.5	9.4	5000	7999.5	9.9	5000	7999.5	11.6	5000	7999.5	14.5	5000	7999.5	17.1	5000
13-16-2-1	3602.8	21.0	5000	3596.9	28.5	5000	3621.4	31.3	5000	3617.5	44.4	5000	3590.9	52.6	5000
15-17-2-1	4246.8	25.6	5000	4262.4	35.8	5000	4263.1	38.8	5000	4104.4	54.6	5000	4285.9	61.2	5000
17-21-3-1	4723.8	21.3	5000	4695.9	38.2	5000	4662.6	46.6	5000	4721.8	96.1	5000	4728.5	132.2	5000
19-22-2-1	8788.9	27.3	5000	8629.0	43.1	5000	8767.3	54.8	5000	8758.2	111.3	5000	8771.9	144.3	5000
21-27-3-1	5382.5	43.3	5000	5308.6	107.3	5000	5255.2	144.4	5000	5256.4	231.3	5000	5326.4	290.4	5000
23-27-3-1	5276.5	55.2	5000	5204.1	130.7	5000	5171.4	174.6	5000	5135.2	277.9	5000	5057.1	338.0	5000
25-31-4-1	6903.3	48.1	5000	6850.2	134.0	5000	6739.7	181.6	5000	6866.5	279.1	5000	6720.9	356.9	5000
27-34-5-1	7288.7	71.8	5000	7240.0	176.4	5000	7287.3	230.0	5000	7250.0	353.1	5000	7348.0	439.6	5000
Average	5735.7	32.8	5000	5693.1	71.0	5000	5691.2	92.0	5000	5685.4	147.0	5000	5697.4	184.1	5000

Table E.1: Results from the tuning runs of parameter $\hat{q}$. Each instance was run five times for each interval for $\hat{q}$. The column $\overline{Obj.}$ presents the average objective values of the five runs. The column *Time[s]* specifies the average time for the runs of the instance in seconds, and the column *Iter.* presents the average number of iterations completed for the five runs.

E.2 Score Parameters

Adjusting the adaptive weights used in the ALNS heuristic is done through the use of score parameters. σ_1 reward finding a new globally best solution, σ_2 a new locally improving solution and σ_3 finding a new solution. The score parameter settings tested are the following: $\sigma = [33, 9, 13]$, $\sigma = [33, 9, 1]$, $\sigma = [33, 9, 9]$, $\sigma = [9, 9, 9]$ and $\sigma = [9, 9, 1]$. These settings are chosen to test how the relationships between them could affect the test instances differently. The results for tuning the score parameters are presented in Table E.2.

The objective values, times, and the number of iterations included in Table E.2 are averages from running the various instances five times for each setting. The results demonstrate that the average runtimes for the various score parameter settings are very similar, ranging from 140.9 s to 147.1 s. Furthermore, the objective values of the settings are also similar, with $\sigma = [33, 9, 1]$ resulting in the best total average value of the five. Further, we see that the fundamentally different score parameters $\sigma = [33, 9, 1]$ and $\sigma = [9, 9, 9]$ result in the best average objective value for some instances each, suggesting that the value of σ does not have a significant effect on the results of the ALNS heuristic. Nevertheless, a setting has to be selected, and $\sigma = [33, 9, 1]$ is chosen based on it having the best total average objective value.

Instance	$\sigma = [\mathbf{33}, \mathbf{9}, \mathbf{13}]$			$\sigma = [\mathbf{33}, \mathbf{9}, \mathbf{1}]$			$\sigma = [\mathbf{33}, \mathbf{9}, \mathbf{9}]$			$\sigma = [\mathbf{9}, \mathbf{9}, \mathbf{9}]$			$\sigma = [\mathbf{9}, \mathbf{9}, \mathbf{1}]$		
	Obj.	**Time [s]**	**Iter.**	**Obj.**	**Time[s]**	**Iter.**	**Obj.**	**Time[s]**	**Iter.**	**Obj.**	**Time[s]**	**Iter.**	**Obj.**	**Time [s]**	**Iter.**
9-9-1-1	3144.5	8.1	5000	3144.5	8.5	5000	3144.5	8.6	5000	3144.5	8.5	5000	3144.5	8.3	5000
11-13-1-1	7999.5	15.0	5000	7999.5	16.2	5000	7999.5	15.9	5000	7999.5	16.1	5000	7999.5	15.6	5000
13-16-2-1	3609.6	44.9	5000	3620.4	44.8	5000	3582.4	45.9	5000	3603.0	45.5	5000	3611.3	43.2	5000
15-17-2-1	4277.9	49.8	5000	4251.7	51.3	5000	4295.4	52.6	5000	4268.0	50.1	5000	4264.1	51.4	5000
17-21-3-1	4686.1	92.9	5000	4627.8	93.9	5000	4720.4	94.5	5000	4658.9	91.9	5000	4718.7	86.7	5000
19-22-2-1	8751.4	113.2	5000	8753.3	114.0	5000	8716.4	112.4	5000	8617.0	108.2	5000	8775.8	114.3	5000
21-27-3-1	5384.6	231.7	5000	5377.1	228.3	5000	5180.4	230.9	5000	5286.1	231.9	5000	5314.2	216.5	5000
23-27-3-1	4924.3	281.4	5000	5034.5	270.3	5000	5194.6	275.7	5000	5232.7	273.2	5000	5188.3	269.5	5000
25-31-4-1	6862.9	282.4	5000	6863.4	274.3	5000	6767.3	278.7	5000	6846.5	280.2	5000	6860.2	266.3	5000
27-34-5-1	7305.9	351.4	5000	7128.3	345.8	5000	7318.4	255.6	5000	7273.0	338.9	5000	7159.8	337.6	5000
Average	5694.7	147.1	5000	5680.1	144.8	5000	5691.9	147.1	5000	5692.9	144.4	5000	5703.6	140.9	5000

Table E.2: Results from the tuning runs of parameter σ. Each instance was run five times for each setting of σ. The column $\overline{Obj.}$ presents the average objective values of the five runs. The column *Time[s]* specifies the average time for the runs of the instance in seconds, and the column *Iter.* presents the average number of iterations completed for the five runs.

E.3 Reaction Parameter

The reaction parameter r guides the adaptive of the weights at the end of each segment of the ALNS heuristic, i.e., each time I^S iterations have been run. A value of $r = 1$ means the weights will be updated exclusively on the scores collected in the last I^S iterations. $r = 0$ means the weights will not be updated at all. This process works according to what is described in Section 6.6. The results from the tuning of the reaction parameter are presented in Table E.3.

The objective values, times, and the number of iterations included in Table E.3 are averages from running the various instances five times for each setting. We observe that the average runtimes for the various reaction parameters are very similar, ranging from 142.7 s to 144.2 s. As $r = 0.10$ results in the best total average objective value, this is chosen as the reaction parameter to be used in the ALNS heuristic.

Instance	r = 0.05			r = 0.10			r = 0.20			r = 0.50			r = 1.00		
	$\overline{\text{Obj.}}$	Time[s]	Iter.	$\overline{\text{Obj.}}$	Time[s]	Iter.	$\overline{\text{Obj.}}$	Time[s]	Iter.	$\overline{\text{Obj.}}$	Time [s]	Iter.	$\overline{\text{Obj.}}$	Time[s]	Iter.
9-9-1-1	3144.5	8.5	5000	3144.5	8.3	5000	3144.5	8.5	5000	3144.5	8.3	5000	3144.5	8.4	5000
11-13-1-1	7999.5	16.3	5000	7999.5	16.2	5000	7999.5	16.2	5000	7999.5	16.0	5000	7999.5	16.1	5000
13-16-2-1	3612.3	43.7	5000	3604.3	43.3	5000	3601.8	44.8	5000	3621.4	43.5	5000	3621.4	43.9	5000
15-17-2-1	4295.3	51.7	5000	4254.2	52.4	5000	4280.0	52.7	5000	4064.4	50.4	5000	4299.8	52.3	5000
17-21-3-1	4688.9	91.7	5000	4714.3	90.4	5000	4692.2	81.5	5000	4614.8	88.3	5000	4644.4	86.2	5000
19-22-2-1	8751.5	109.6	5000	8663.3	110.5	5000	8766.8	116.3	5000	8749.7	116.5	5000	8726.8	112.1	5000
21-27-3-1	5166.1	228.1	5000	5269.5	217.3	5000	5195.1	226.5	5000	5294.1	227.1	5000	5169.9	222.8	5000
23-27-3-1	5208.0	270.9	5000	5067.5	279.5	5000	5052.9	274.5	5000	5108.4	271.9	5000	5158.8	274.6	5000
25-31-4-1	6893.3	270.8	5000	6797.7	271.0	5000	6879.6	275.2	5000	6899.7	271.7	5000	6817.6	271.3	5000
27-34-5-1	7312.5	346.1	5000	7064.1	337.9	5000	7086.2	338.5	5000	7248.4	348.1	5000	7198.6	338.9	5000
Average	5707.2	143.8	5000	5657.9	142.7	5000	5669.9	143.5	5000	5672.7	144.2	5000	5678.1	142.7	5000

Table E.3: Results from the tuning runs of parameter r. Each instance was run five times for each value of r. The column $\overline{Obj.}$ presents the average objective values of the five runs. The column *Time[s]* specifies the average time for the runs of the instance in seconds, and the column *Iter.* presents the average number of iterations completed for the five runs.

E.4 Noise Control Parameter

The noise control parameter sets the level of randomness to add when inserting orders in a partial solution to the problem. Zero noise results in the orders being inserted solely according to the principles of the relevant repair heuristic described in Section 6.5.2. Large noise will enable the repair heuristic to choose an order to insert that is not necessarily the locally best insertion. The use of noise is described in Section 6.5.3. The results of the tuning of the noise control parameter are presented Table E.4.

The objective values, times, and the number of iterations included in Table E.4 are averages from running the various instances five times for each setting. The average runtimes for the various noise control parameters are very similar, ranging from 139.7 s to 149.4 s. We see that no noise value is demonstrably better for all the test instances than another, but $\eta = 0.025$ shows the total best average objective value. The noise control parameter selected is therefore $\eta = 0.025$.

Instance	$\eta = 0.000$			$\eta = 0.025$			$\eta = 0.125$			$\eta = 0.250$			$\eta = 0.500$		
	$\overline{\textbf{Obj.}}$	**Time[s]**	**Iter.**	$\overline{\textbf{Obj.}}$	**Time[s]**	**Iter.**	$\overline{\textbf{Obj.}}$	**Time[s]**	**Iter.**	$\overline{\textbf{Obj.}}$	**Time [s]**	**Iter.**	$\overline{\textbf{Obj.}}$	**Time[s]**	**Iter.**
9-9-1-1	3144.5	6.4	5000	3144.5	7.1	5000	3144.5	7.2	5000	3144.5	8.2	5000	3144.5	9.7	5000
11-13-1-1	7999.5	14.2	5000	7999.5	17.2	5000	7999.5	15.8	5000	7999.5	16.4	5000	7999.5	16.7	5000
13-16-2-1	3621.4	41.7	5000	3614.9	42.5	5000	3614.9	41.0	5000	3609.0	44.3	5000	3614.9	46.4	5000
15-17-2-1	4295.3	50.0	5000	4068.4	51.1	5000	4097.5	48.9	5000	4260.7	51.8	5000	4288.1	57.3	5000
17-21-3-1	4701.2	92.0	5000	4696.5	87.0	5000	4704.0	88.9	5000	4636.4	89.4	5000	4706.7	104.9	5000
19-22-2-1	8732.7	118.6	5000	8713.3	112.3	5000	8665.7	99.4	5000	8712.4	107.4	5000	8745.0	112.7	5000
21-27-3-1	5231.1	219.4	5000	5276.4	216.8	5000	5327.3	217.3	5000	5287.4	219.9	5000	5188.9	235.5	5000
23-27-3-1	5230.2	267.3	5000	5151.8	259.8	5000	5108.1	269.7	5000	5217.4	267.7	5000	5069.7	286.3	5000
25-31-4-1	6748.0	267.7	5000	6830.7	267.5	5000	6853.9	267.1	5000	6706.5	277.7	5000	6943.4	283.8	5000
27-34-5-1	7223.4	333.8	5000	7232.5	341.4	5000	7309.44	341.5	5000	7177.7	342.0	5000	7128.9	340.7	5000
Average	5692.7	141.1	5000	5672.8	140.3	5000	5682.5	139.7	5000	5675.1	142.5	5000	5683.0	149.4	5000

Table E.4: Results from the tuning runs of parameter η. Each instance was run five times for each value of η. The column $\overline{Obj.}$ presents the average objective values of the five runs. The column *Time[s]* specifies the average time for the runs of the instance in seconds, and the column *Iter.* presents the average number of iterations completed for the five runs.

E.5 Determinism Parameter

The determinism parameter p guides the amount of randomness in the related and worst removal heuristics described in Section 6.5.1. Respectively, these two heuristics choose which orders to remove based on their relatedness or contribution to the objective function value. A value of $p = 1$ will make the choice of order to remove totally random. In general, the larger the p, the more random the choice will be. The use of the determinism parameter is described in Section 6.5.1. The results of the tuning of p are presented in Table E.5.

The objective values, times, and number of iterations included in Table E.5 are averages from running the various instances five times for each setting. The average runtimes for the various determinism parameters are similar, ranging from 136.0 s to 141.0 s. Observing the average objective values, we see that no determinism parameter is demonstrably better for all the test instances than another. $p = 7$ shows the total best average objective value and is therefore chosen as the determinism parameter to be used in the ALNS heuristic.

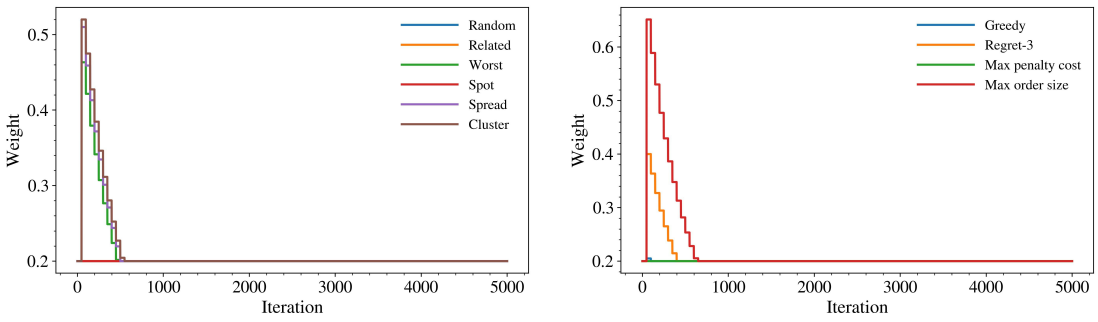
Instance	p = 3			p = 5			p = 7			p = 9			p = 11		
	$\overline{\text{Obj.}}$	Time[s]	Iter.	$\overline{\text{Obj.}}$	Time[s]	Iter.	$\overline{\text{Obj.}}$	Time[s]	Iter.	$\overline{\text{Obj.}}$	Time [s]	Iter.	$\overline{\text{Obj.}}$	Time[s]	Iter.
9-9-1-1	3144.5	7.1	5000	3144.5	6.9	5000	3144.5	7.2	5000	3144.5	6.9	5000	3144.5	7.1	5000
11-13-1-1	7999.5	15.1	5000	7999.5	16.1	5000	7999.5	15.2	5000	7999.5	15.5	5000	7999.5	15.2	5000
13-16-2-1	3616.5	41.2	5000	3615.2	41.2	5000	3612.2	40.8	5000	3621.4	41.3	5000	3615.8	39.9	5000
15-17-2-1	4263.4	50.1	5000	4280.5	46.8	5000	4268.8	48.7	5000	4281.8	49.6	5000	4278.9	48.2	5000
17-21-3-1	4725.4	79.3	5000	4695.6	83.2	5000	4687.6	83.1	5000	4719.7	87.6	5000	4692.1	86.6	5000
19-22-2-1	8729.9	114.0	5000	8680.8	109.8	5000	8682.8	114.3	5000	8768.5	114.2	5000	8721.4	101.6	5000
21-27-3-1	5308.4	224.1	5000	5277.9	218.6	5000	5287.1	221.9	5000	5265.4	215.2	5000	5180.9	218.8	5000
23-27-3-1	4985.6	267.0	5000	5181.3	269.7	5000	5143.2	264.3	5000	5147.4	262.9	5000	5172.8	258.0	5000
25-31-4-1	6879.4	269.0	5000	6869.0	265.2	5000	6788.2	261.3	5000	6658.8	259.7	5000	6892.8	256.5	5000
27-34-5-1	7109.9	343.4	5000	7278.5	333.6	5000	7093.1	344.2	5000	7154.5	333.1	5000	7129.7	327.9	5000
Average	5676.3	141.0	5000	5702.3	139.1	5000	5670.7	140.1	5000	5676.1	138.6	5000	5682.8	136.0	5000

Table E.5: Results from the tuning runs of parameter p. Each instance was run five times for each value of p. The column $\overline{Obj.}$ presents the average objective values of the five runs. The column *Time[s]* specifies the average time for the runs of the instance in seconds, and the column *Iter.* presents the average number of iterations completed for the five runs.

Appendix F

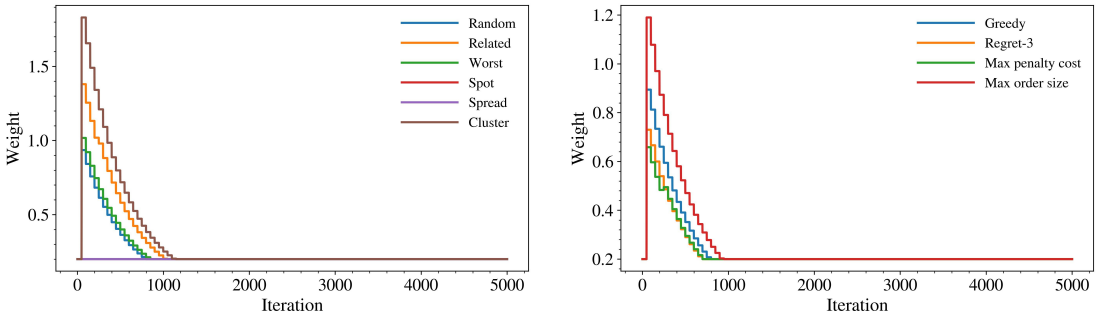
Adaptive Weights Development

This appendix shows the development of the adaptive weights used in the ALNS heuristic for test instances {7-8-1-3}, {9-10-1-1}, {11-13-2-1}, {13-15-2-2}, {15-18-2-1}, {17-19-3-2}, {19-21-3-2}, {21-24-3-1}, {23-27-4-1} and {27-32-5-1}. Hence, the development for one instance from each instance group is presented. Note that no weight development from instance group 5 is presented as the construction heuristic is able to find the optimal solution for all instances in the group.



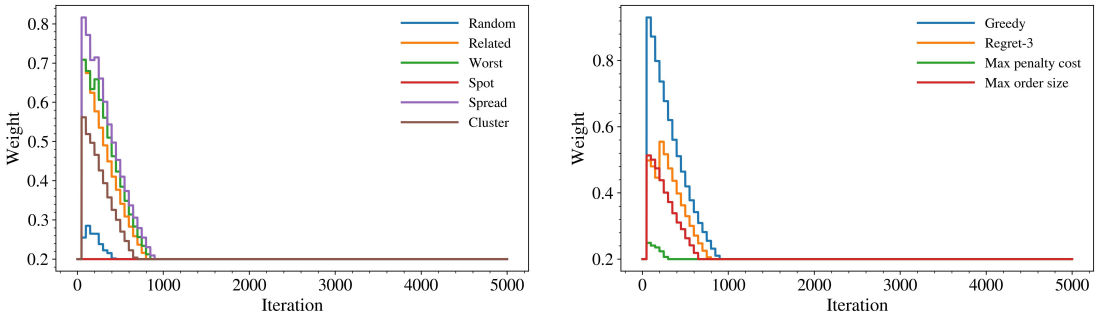
(a) Destroy heuristics **(b)** Repair heuristics

Figure F.1: Development of the weights associated to the destroy and repair heuristics. The plots represent a run of the test instance 7-8-1-3, and values are sampled at every update of the weights.



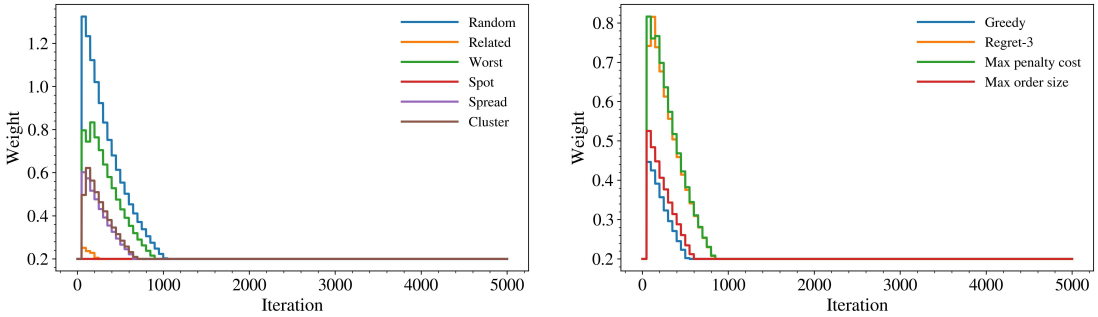
(a) Destroy heuristics **(b)** Repair heuristics

Figure F.2: Development of the weights associated to the destroy and repair heuristics. The plots represent a run of the test instance 9-10-1-1, and values are sampled at every update of the weights.



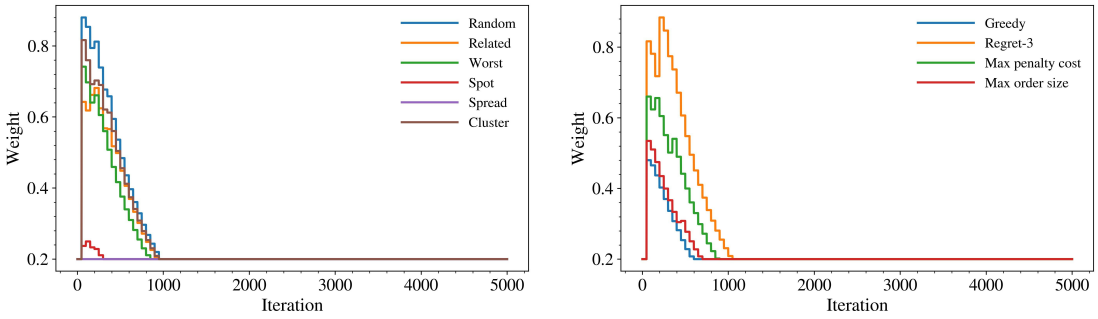
(a) Destroy heuristics **(b)** Repair heuristics

Figure F.3: Development of the weights associated to the destroy and repair heuristics. The plots represent a run of the test instance 11-13-2-1, and values are sampled at every update of the weights.



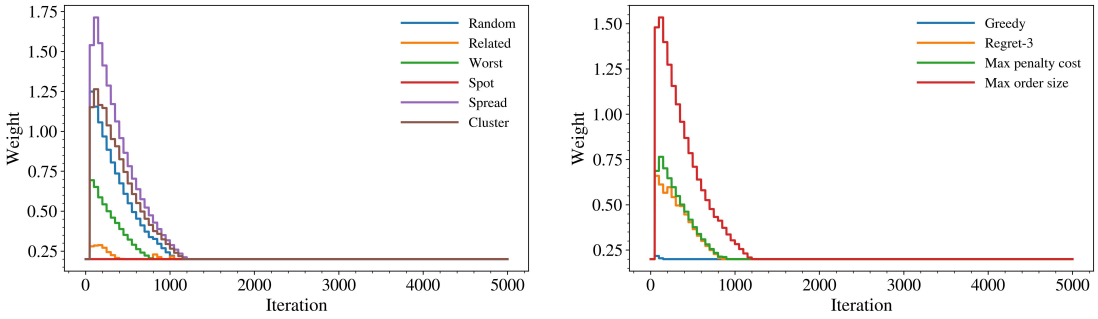
(a) Destroy heuristics **(b)** Repair heuristics

Figure F.4: Development of the weights associated to the destroy and repair heuristics. The plots represent a run of the test instance 13-15-2-2, and values are sampled at every update of the weights.



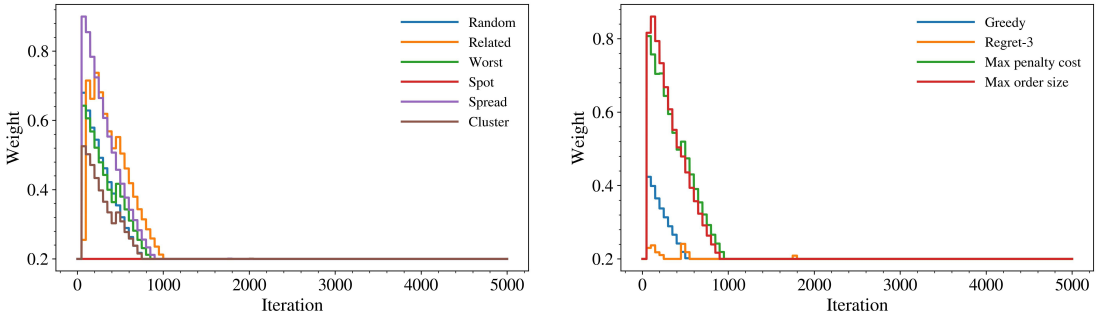
(a) Destroy heuristics **(b)** Repair heuristics

Figure F.5: Development of the weights associated to the destroy and repair heuristics. The plots represent a run of the test instance 15-18-2-1, and values are sampled at every update of the weights.



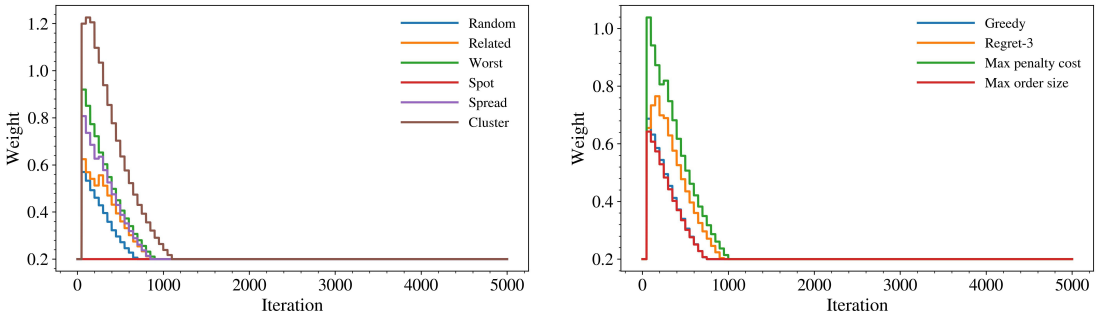
(a) Destroy heuristics **(b)** Repair heuristics

Figure F.6: Development of the weights associated to the destroy and repair heuristics. The plots represent a run of the test instance 17-19-3-2, and values are sampled at every update of the weights.



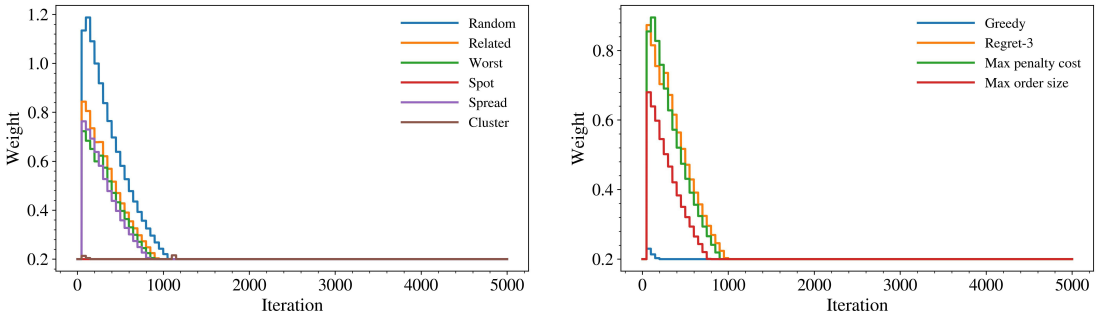
(a) Destroy heuristics **(b)** Repair heuristics

Figure F.7: Development of the weights associated to the destroy and repair heuristics. The plots represent a run of the test instance 19-21-3-2, and values are sampled at every update of the weights.



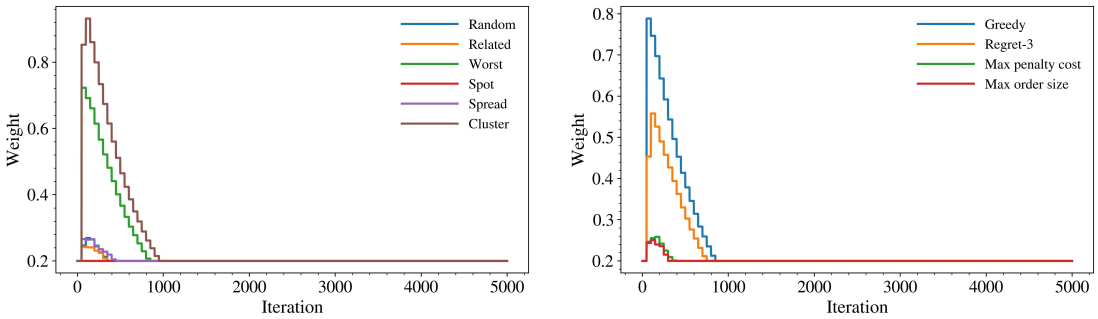
(a) Destroy heuristics **(b)** Repair heuristics

Figure F.8: Development of the weights associated to the destroy and repair heuristics. The plots represent a run of the test instance 21-24-4-1, and values are sampled at every update of the weights.



(a) Destroy heuristics **(b)** Repair heuristics

Figure F.9: Development of the weights associated to the destroy and repair heuristics. The plots represent a run of the test instance 23-27-4-1, and values are sampled at every update of the weights.



(a) Destroy heuristics **(b)** Repair heuristics

Figure F.10: Development of the weights associated to the destroy and repair heuristics. The plots represent a run of the test instance 27-32-5-1, and values are sampled at every update of the weights.

Appendix G

Code and Test Instances

The code and test instances can be found in the provided ZIP file named "code-masters-thesis-ulsrud-vandvik.zip" attached to the delivery of this thesis, or at https://github.com/avandvik/masters-thesis. The ZIP file contains two folders:

1. exact_solver: The implementation of the exact model and the preprocessing procedure described in Chapter 4.
2. alns: The implementation of the ALNS heuristic, including the VCP, local search, and subproblem described in Chapters 5 and 6. All test instances, the instance generation procedure described in Chapter 7, and output files from the solution methods is also found here.

G.1 Exact Solver

The code for the arc-flow model presented in Chapter 4 is found in the folder "exact_solver/arc_flow". The folder "exact_solver/arc_flow/preprocessing" contains the files with the implementation of the preprocessing arc-generation procedure explained in Section 4.2, while the implementation of mathematical model formulated in Section 4.3 is found in "exact_solver/arc_flow/mathematical_model". All parameters for these methods are set in "data.py". This file also reads all problem instance-specific data at the beginning of each run. Model output is saved in the folder "exact_solver/output".

G.2 ALNS Heuristic

The code for the implementation of the ALNS heuristic with all its extensions described in Chapter 6 is found in the folder "alns/src/main/java". This folder also contains the

code for the implementation of the subproblem described in Chapter 5, and a file called "Parameters.java" setting the parameters of the heuristic and its extensions. The output from running the heuristic is saved in JSON files and stored in "alns/output". The folder "alns/output/solstorm" contains all output files from the runs used in the computational study in Chapter 8. These files are used to generate the dataframes in the Jupyter Notebooks in the folder "alns/plot", providing the basis for the tables presented in Chapter 8.

G.3 Instance Generator

The instance generator outlined in Chapter 7 is found in the folder "alns/src/main/java/utils". The generated test instances are found in subfolders in the folder "alns/src/main/resources/. The tuning instances are in the folder "alns/src/main/resources/tuning", and the test instances used for performance testing are found in the folder "alns/src/main/resources/performance". All instances used in this thesis are generated by this instance generator, except the small example instances presented in Chapter 9.

Karl Petter Ulsrud and Anders Vandvik

An ALNS Approach to the TDVRPSO

NTNU
Norwegian University of
Science and Technology