Master's thesis

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

Aksel Borgen, Anders Bjelland and Sjur Wold

A Memetic Algorithm With Optimal Quantity Assignments for the Fish Feed Maritime Inventory Routing Problem

Master's thesis in Industrial Economics and Technology Management
Supervisor: Kjetil Fagerholt
Co-supervisor: Kristian Thun and Simen Tung Vadseth
June 2022

NTNU
Norwegian University of Science and Technology

Aksel Borgen, Anders Bjelland and Sjur Wold

A Memetic Algorithm With Optimal Quantity Assignments for the Fish Feed Maritime Inventory Routing Problem

Master's thesis in Industrial Economics and Technology Management
Supervisor: Kjetil Fagerholt
Co-supervisor: Kristian Thun and Simen Tung Vadseth
June 2022

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

NTNU
Kunnskap for en bedre verden

DEPARTMENT OF INDUSTRIAL ECONOMICS
AND TECHNOLOGY MANAGEMENT

TIØ4905 - Managerial Economics and Operations Research, Master's Thesis

A Memetic Algorithm With Optimal Quantity Assignment for the Fish Feed Maritime Inventory Routing Problem

Authors:
Anders Bjelland
Aksel Borgen
Sjur Wold

Supervisor:
Kjetil Fagerholt

Co-supervisors:
Simen Tung Vadseth
Kristian Thun

June, 2022

Preface

This thesis is a deliverable for the work done in TIØ4905 – Managerial Economics and Operations Research, Master's Thesis at the Norwegian University of Science and Technology. The thesis was written during the spring of 2022.

We want to thank our supervisor, Professor Kjetil Fagerholt, and our co-supervisors, researcher Kristian Thun, and PhD student Simen Tung Vadseth, for their valuable guidance and help. Furthermore we extend our gratitude to our industry partners: SINTEF Ocean, and Mowi, for giving us data and insight into the industry. Finally, credits must be given to Ivar Brekkå and Solveig Randøy, for supplementing us with data and help when needed, based on the work they did for their master's thesis during the spring of 2021.

Aksel Borgen, Anders Bjelland, and Sjur Wold

Trondheim, June 2022

Abstract

Farmed Atlantic salmon is one of Norway's most valuable export goods. Supporting this industry, we find the fish feed industry responsible for producing and distributing around 5,000 tonnes of feed to fish farms along the Norwegian coast every day. Low margins make efficient logistics a necessity for operating profitably. We present the current process of producing and distributing fish feed to Atlantic salmon fish farms based on information from SINTEF Ocean and our industry collaborator, Mowi. The focus is on the scale of operations, logistic constraints, demand variations, and challenges and limitations in the current planning process. The current order-based system is a limitation, and industry partners believe that cost savings can be realized by changing to vendor-managed inventory (VMI).

This serves as the basis for our problem, the fish feed maritime inventory routing problem (FFMIRP). It is a heterogeneous vessel fleet, multi-product maritime inventory routing problem (MIRP) formulation for the distribution process, where the feed supplier is responsible for maintaining sufficient inventory at farms. We propose a mathematical model for the FFMIRP, formulated as an arc-flow mixed-integer linear program with a discrete time representation. Due to the scale and complexity of the problem, it is difficult, if not impossible, to solve real-world instances to optimality using a commercial solver.

Consequently, we develop a matheuristic intended to be applied to larger instances. The algorithm is termed the Scalable Memetic Optimization algorithm with LP-based search Techniques (SMOLT). SMOLT employs a memetic algorithm to decide how to route each vessel and combines this with a linear program (LP) for optimal quantity assignment. We use a novel representation for routing consisting of a list of *(port, time)*-pairs for each vessel. Our solution method implements a series of mutations operating on routing solutions – including a set of ruin & recreate methods, among these an adaption of SISR (Christiaens and Vanden Berghe, 2020) to the MIRP. SMOLT has been designed to be easily parallelized, allowing it to be scaled out to nearly arbitrary levels of parallel execution. This allows us to solve larger problems than those addressed in the literature so far.

Test instances are created using real vessel and farm data provided by our industry partner, Mowi. Smaller instances are made by varying the duration of the planning horizon and by sampling a random subset of available farms, vessels, and products. When compared, we find that a commercial solver performs better than SMOLT on some small instances. However, our algorithm usually obtains solutions close to that of the solver in terms of objective value. For larger instances, SMOLT outperforms the commercial solver, and is able to identify reasonable solutions for real-world-sized instances. As part of our analysis, we present SMOLT's performance on the MIRPLib Group 1 instances (Papageorgiou et al., 2014c) to give a comparison to existing solution methods. Despite not being specifically designed to solve the MIRPLib instances, the results indicate that SMOLT can obtain high-quality solutions for most instances. Finally, we give examples of how SMOLT can be used to support more strategic decisions and discuss possible future improvements for the solution approach.

Sammendrag

Oppdrettslaks er en av Norges mest verdifulle eksportvarer. Mindre kjent er fiskefôrindustrien som daglig produserer og distribuerer omtrent 5.000 tonn fôr til anlegg langs hele norskekysten. Lave profittmarginer gjør effektiv logistikk en nødvendighet for å tjene penger i industrien. I denne masteroppgaven presenterer vi de nåværende produksjons- og distribusjonsprosessene av laksefôr basert på informasjon fra SINTEF Ocean og vår industripartner, Mowi. Fokuset ligger på skaleringen av operasjonelle aspekter, restriksjoner på logistikken, variasjon i fôretterspørselen og utfordringer og begrensninger ved dagens system. I dag er distribusjonsprosessen basert på ordre utstedt av anleggene, som håndteres av forhåndsdefinerte ruter. Våre industripartner tror at det kan gjøres besparelser ved å gå over til en form for vendor-managed inventory (VMI).

Dette danner utgangspunktet for vårt problem, maritim ruting og lagerstyring av fiskefôr (fish feed maritime inventory routing problem – FFMIRP). FFMIRP består av en heterogen flåte og flere produkter, og tar for seg en distribusjonsprosess der distributøren er ansvarlig for å opprettholde tilstrekkelig lagerbeholdning på alle anlegg. Vi foreslår en matematisk modell for problemet, formulert som en kantflyt-basert blandet heltallsmodell med en diskret representasjon av tiden. Grunnet problemets kompleksitet er det vanskelig, om ikke umulig, å løse reelle probleminstanser til optimalitet med en kommersiell løser.

Som en konsekvens av problemets kompleksitet, utvikler vi en matheuristikk som er tiltenkt å løse større probleminstanser. Vi har kalt algoritmen "Scalable Memetic Optimization algorithm with LP-based search Techniques" (SMOLT). SMOLT benytter en memetisk algoritme for å bestemme hvordan hvert skip skal rutes, og kombinerer dette med et lineært program (LP) for å tilordne optimale leveringsmengder. Vi benytter en ny representasjon av rutingen, som består av *(port, tid)*-par for hvert skip. Løsningsmetoden vår inkluderer en rekke mutasjonsoperatorer som endrer ruteløsningene – blant annet ulike destruer & gjenoppbygg-metoder. SMOLT er designet for å kunne parallelliseres lett; dette tilrettelegger for høy grad av parallell utførelse. Denne parallelliseringen bidrar til å gjøre det mulig for SMOLT å løse større problemer enn de som er adressert i litteraturen frem til nå.

Testinstansene er generert basert på ekte data om skip og anlegg som vi har fått fra industripartneren vår, Mowi. Mindre instanser er generert ved å variere lengden på planleggingshorisonten og ved å trekke ut et tilfeldig utvalg av tilgjengelige anlegg, båter og produkter. Vi ser at den kommersielle løseren finner bedre løsninger enn SMOLT på noen av de små instansene, men SMOLT finner vanligvis omtrent like gode løsninger også i disse tilfellene. På de større, mer reelle instansene, utkonkurrerer SMOLT den kommersielle løseren og finner fornuftige løsninger. Som en del av analysen, presenterer vi SMOLT sine resultater på "MIRPLib Group 1"-instansene (Papageorgiou et al., 2014c) for å muliggjøre sammenligning med andre løsningsmetoder. Til tross for at SMOLT ikke er utviklet for å løse disse problemene, viser resultatene at algoritmen finner gode løsninger for de fleste instansene. Til slutt gir vi eksempler på mulige områder SMOLT kan bidra som et beslutningsstøttende verktøy i strategiske beslutninger, og diskuterer mulige fremtidige utbedringer av den foreslåtte løsningsmetoden.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This thesis builds upon our specialization project Bjelland et al. (2021), and investigates vendor-managed inventory (VMI) within the salmon feed industry. The thesis focuses on feed distribution for our industry collaborator, the Norwegian salmon farming company, Mowi. The following chapter is an updated version of the introduction in the specialization project.

The Norwegian aquaculture industry has seen tremendous growth over the last couple of decades. In 2021 the industry accounted for 116.6 billion NOK, equalling approximately 21.5% of the total Norwegian mainland exports (Statistics Norway, 2021). Farmed salmon constitutes approximately 70% of the fish exported, and production totaled[1] almost 1.6 million tonnes in 2021 as shown in Figure 1.1 (Directorate of Fisheries, 2021). In the same year, the aquaculture industry reported consumption of just over two million tonnes of fish feed, more than twice as much as 15 years ago (Directorate of Fisheries, 2021). This implies that on average, over 5,000 tonnes of fish feed has to be produced and distributed daily. Increasing demand for fish feed puts a strain on feed producers, as there is a continuous push for higher production. In addition, there is a downwards push on margins for fish feed producers. Fish feed is the largest operational cost in the aquaculture industry, which leads to farming companies working hard to keep prices down. As a result, some aquaculture companies, such as Mowi, have started vertically integrating feed production into their operations to gain a competitive advantage.

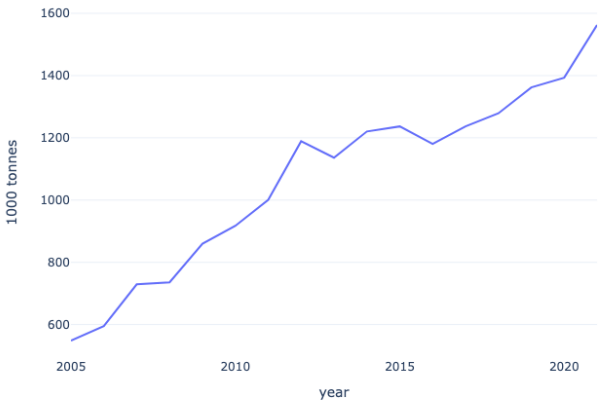


Figure 1.1: Harvested salmon in Norway from 2005 to 2021.

Source: Directorate of Fisheries (2021)

In combination with intense competition, these factors have led to low margins in the fish feed

[1]The sum of the domestically consumed and exported volumes.

production industry, particularly when compared to the salmon farming industry. For comparison, Mowi Feed and Mowi Farming had an operational EBIT margin in 2021 of 2.7% and 14.4%, respectively (Mowi, 2021a). Therefore, focusing on costs is vital for feed producers to remain competitive. The main cost components in the fish feed industry are the raw materials used for the feed and other production costs. As the available amount of fish meal has decreased, and the relative cost of marine ingredients compared to vegetable ingredients has increased, producers of fish feed have gradually replaced the marine ingredients with vegetable ingredients such as soy (Mowi, 2021b).

Another major cost component is the cost of distribution. Distribution is usually done by specialized feed vessels transporting the different feed products from the factories to the fish farms along the Norwegian coast. For BioMar, a large fish feed supplier, this is the second-largest cost category, making up almost 5% of their total costs (BioMar Group A/S, 2021). Increased production puts a bigger burden on effective logistics. Larger shipments and more unique delivery locations increase the difficulty of planning routes for the vessels. Manually planning a route that ensures availability of feed at all customer locations becomes difficult by itself, and doing so in a cost-effective manner may become intractable. Applying mathematical optimization principles and approximation methods can be a helpful solution to cope with this increased complexity. Even minor relative improvements in vessel routing can lead to considerable savings in absolute terms, due to distribution being a large cost center for feed suppliers.

Traditionally, salmon farmers have placed orders for feed that must be delivered within a given time window. Based on the received orders, the supplier creates a distribution schedule. This schedule is often constructed manually. Both placing the orders and planning the distribution are time-consuming. Also, this approach rarely results in an optimal solution. With ever more sophisticated storage monitoring techniques used at the farms, it is now possible to reduce the time required by farmers to place orders. Instead of placing orders stating the quantity and delivery window, the farmers can instead provide the feed producer with its storage data and expected feed consumption throughout the planning horizon. The feed supplier's task then changes from serving all orders to ensuring that the farmers' feed storage never runs empty within the planning horizon. This approach, commonly known as VMI, increases the feed supplier's flexibility and can result in more efficient routes and schedules for their fish feed vessels.

The overall objective of this master's thesis is to develop an algorithm that can serve as the basis for a decision support system for planning the distribution schedule of fish feed. The distribution plan should state which farms will be visited by which vessels at what time and the quantity of the different feed types delivered during a visit. The plan should minimize the supplier's operational distribution costs while meeting all system requirements, e.g., ensuring a sufficient supply of feed for farmers and respecting the capacity of the vessels. Therefore, such a tool can make the interaction between farmers and feed suppliers more seamless while also substantially improving the quality and robustness of the planning process.

Optimizing the distribution schedule for fish feed is a complex problem. When only focusing on serving the placed orders, the problem is a variant of the vehicle routing problem (VRP). However, when allowing the supplier to decide both when and how much feed to deliver, the problem changes to an inventory routing problem (IRP). This is commonly referred to as a maritime inventory routing problem (MIRP) for maritime applications. Optimal ship routing and scheduling are in general well studied, and according to Christiansen et al. (2013) the volume of research within the field doubled every decade up to 2013. This strong interest and attention suggest that robust and high-quality planning tools for ship routing are essential for remaining competitive. The MIRP is an NP-hard problem, meaning that no known polynomial-time algorithm solves it. Despite the extensive research conducted within the field, for instance by Papageorgiou et al. (2014b), Hemmati et al. (2016), and Friske et al. (2022), the solution methods for MIRPs are not yet sophisticated enough to handle many of the problem instances encountered in real-world logistics operations. One of the main challenges when optimizing a feed supplier's distribution schedule is the system's size and complexity. Since finding the optimal solution is close to impossible, we instead develop a memetic matheuristic that can be applied to identify good solutions. In addition, the matheuristic can also be of value when evaluating strategic decisions, such as installing extra storage capacity or acquiring additional vessels. For example, one can compare the solutions generated when the

new vessels are included and when they are not.

The thesis includes primarily two contributions to the MIRP research. Firstly, we have created an extended mathematical model describing a complex MIRP for the fish feed industry with a new combination of constraints. To our knowledge, this is the first formulation of this variant of the MIRP. Secondly, we have implemented a novel matheuristic for solving real-life instances of the MIRP. The matheuristic is termed the Scalable Memetic Optimization algorithm with LP-based search Techniques, which is abbreviated SMOLT. SMOLT combines a memetic algorithm, which is a genetic algorithm using local search techniques for making routing decisions, and a linear program (LP) for deciding the quantities to deliver along the routes. Each individual in the population gives an exhaustive route for all vessels, including the times of all visits. The linear program is then solved for the routes with the goal of keeping the feed inventories within the specified limits at all times.

This report is organized as follows: Chapter 2 describes the context and background of the problem. Next, the problem is formally described in Chapter 3, and related literature is discussed in Chapter 4. A mathematical model is presented in Chapter 5. In Chapter 6, our matheuristic, named SMOLT, is described in detail, and the test instances and computational study is presented in Chapter 7 and Chapter 8, respectively. Lastly, our concluding remarks and suggestions for future work are outlined in Chapter 9 and Chapter 10.

Chapter 2

The Fish Feed Industry

This chapter gives an overview of the Norwegian fish feed industry and how distribution is planned and executed. A substantial share of the content presented is based on information provided by our industrial partner. Lastly, some challenges with the current distribution planning are identified. This chapter is retrieved from our specialization project Bjelland et al. (2021), but some numerical values are updated.

2.1 Industry Overview

With the rise of the Norwegian aquaculture industry, a large salmon feed market has emerged. The industry has grown steadily, with a compound annual growth rate (CAGR) of approximately 8.5% the last decade, and totaled over NOK 27 billion in revenues in 2019 (EY, 2020). The Norwegian salmon feed industry is dominated by a few prominent actors and has become increasingly consolidated in the last decade (Mowi, 2019). Skretting, Ewos, and BioMar were the major actors and controlled much of the feed output in the years prior to Mowi's establishment of feed production in 2014 (Mowi, 2019). Mowi is aiming to become self-sufficient for all feed requirements and achieved a 95% self-sufficiency rate in Norway in 2021 (Mowi, 2021a). The company is the world's largest producer of farmed salmon and harvested more than one-fifth of the total volume in Norway in 2020 (Mowi, 2021b). Due to its strategy and size, Mowi's share of the Norwegian salmon feed market has increased significantly since 2014 and was approximately 19% in 2018, as shown in Figure 2.1.

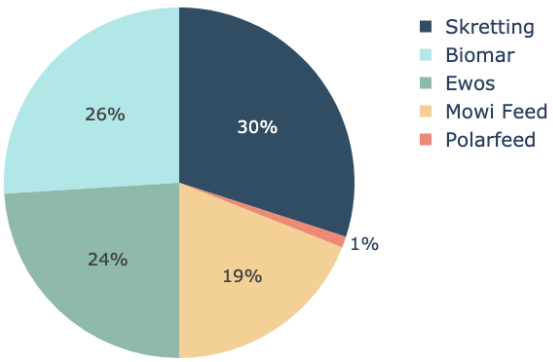


Figure 2.1: Market shares in the Norwegian fish feed industry in 2018.

Source: Mowi (2019)

The Norwegian fish feed market is nearly perfectly correlated to the production of salmon in Norway. Since the 1970s, the volume of harvested Norwegian salmon has increased to approximately 1.6 million tonnes (Directorate of Fisheries, 2021), and farmed salmon has gradually become one of Norway's most important industries. The Norwegian government has an ambition of increasing the export volume by a factor of five by 2050, as shown in Figure 2.2 (Norsk Industri, 2017). This makes it reasonable to expect an increase in the demand for salmon feed in the coming years and hence continued growth of the fish feed industry. This growth also comes with challenges for the suppliers, as high volumes will increase the complexity of the distribution system. Developing sophisticated planning tools for distribution can help suppliers gain a competitive advantage compared to their competitors.

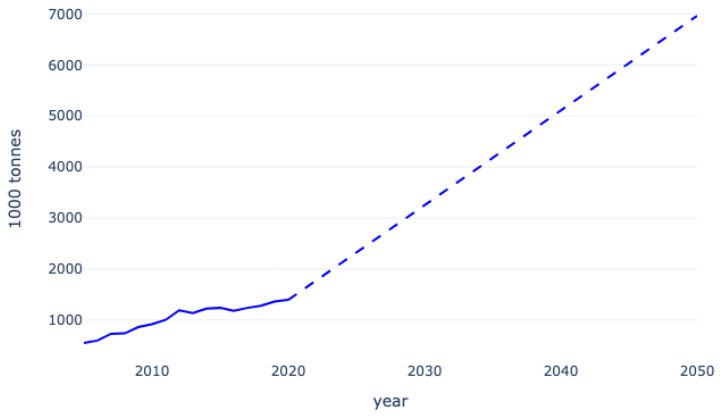


Figure 2.2: The Norwegian government is aiming at harvesting five times more salmon in 2050 compared to today.

Source: Norsk Industri (2017), Directorate of Fisheries (2021)

2.2 Production Process

This section describes the production process of fish feed and seeks to give the reader an understanding of the product's characteristics. The content is based on conversations with Mowi and the description of Mowi's current production facilities in Bjugn in Trøndelag by Haugland and Thygesen (2017). The plant in Bjugn is Mowi's only feed factory in Norway and produces nearly all the feed supplied to the company's Norwegian farms. The facility is capable of producing feed continuously throughout the year, and the production totaled 358,769 tonnes in 2021, close to the total capacity of 400,000 tonnes (Mowi, 2021a). Also, as the optimal feed for salmon varies through its life cycle in terms of size and ingredients, a range of products are produced at Bjugn. The raw materials going into production are roughly categorized as wet or dry, and the plant has a storage capacity of 10,000 and 16,000 tonnes for each, respectively. Lastly, it has a storage capacity of 10,000 tonnes of the finished products. The fish feed must satisfy different requirements to be considered high quality. First, the product must have the physical properties that allow it to be transported in large volumes. The finished fish feed has the shape of pellets and is stored and transported in large bags or silos. During transportation, some degree of pellet erosion causes degradation and pulverization of pellets. Due to friction, the pellets are especially exposed to erosion during movement. In practice, this occurs when a bag is moved, or a silo is loaded or unloaded. For bulk transport, Mowi operates with a degradation ratio of 0.5%, while they consider the degradation of feed transported in bags to be slightly higher. When pulverized feed is released into the water, it negatively impacts the water quality and can cause an overgrowth of algae, negatively influencing the salmon's health and growth. Therefore, the pellet's physical

properties should minimize the pulverization occurring during transport.

Other important requirements concern the pellets' properties in contact with water. For instance, the pellets must sink at a speed that allows the fish to eat them before reaching the cage's bottom. Furthermore, all nutrients should remain inside the pellets until they are eaten. If nutrients dissolve into the water, the feed quality is significantly reduced, which negatively impacts the growth of the salmon. Lastly, the pellets must be attractive for the fish to eat, setting requirements to taste and consistency.

The fish feed production starts with different raw materials that are often categorized as macro, micro, and oils and are normally transported by sea. Macro materials are plant and animal ingredients like fish meal and barley grain, while micro materials are vitamins and minerals. The oils used are both animal- and plant-based. When the raw materials arrive at the production plant, they are stored in large silos. The first step in the production process is to scale and measure the different materials going into the feed to secure the correct composition. Next, the components are mixed and ground in large blenders before water is added to the mixture. At this point, the mixture is ready to be shaped into pellets. Depending on the specific feed being produced, the pellets have a diameter between 0.6 mm and 13 mm. The pellets are then dried and treated with oil. The final step is to place the pellets in an environment with specific pressure and temperature to obtain the desired characteristics. The finished product is then stored in large bags or silos before being transported by sea to farms. An overview of the production process is illustrated in Figure 2.3.

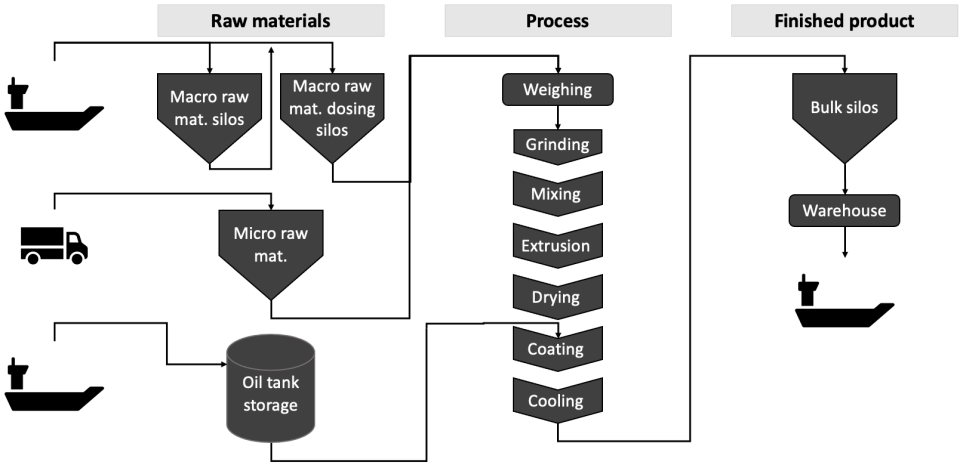


Figure 2.3: Production process of salmon feed at Mowi's production plant in Bjugn.

Source: Haugland and Thygesen (2017)

The physical properties and ingredients of the salmon feed have significantly changed since the early phases of the industry. In the 1970s, farmers produced their own feed, mainly consisting of fish waste, fish meal, and shrimp shells. The feed had high water content, high levels of marine protein, and low levels of fat and oil (Mowi, 2021b). In the 1990s, the feed consisted of 45% protein, which was mainly made up of marine protein (Mowi, 2021b). The main differences between today's and early phase feed are the reduced level of marine protein and the higher inclusion of fat. The level of marine protein has decreased due to cost optimization and the volatile supply of fish meal, and higher inclusion of fat has become possible through technological improvements and extruded feeds. The marine ingredients have to some extent been replaced with vegetarian alternatives such as corn or soy, and the main ingredients in today's feed are vegetarian meal, vegetarian oil, and fish meal, as shown in Figure 2.4.

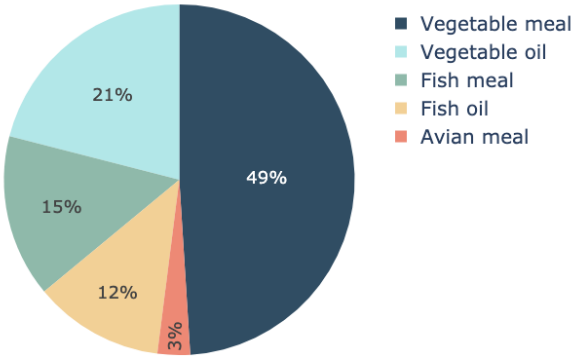


Figure 2.4: Mowi feed composition 2020.

Source: (Mowi, 2021b)

2.3 Distribution System

In this section, we discuss Mowi's current distribution system. Section 2.3.1 briefly describes restrictions in the current distribution network. Further, we look into different factors that affect the demand for fish feed, such as sea temperature and growth stage, in Section 2.3.2. Section 2.3.3 describes how the distribution is currently planned. Lastly, Section 2.3.4 highlights challenges in the current setup, and motivates a transition to a VMI-based system.

2.3.1 Current Distribution Network

There is a multitude of complicating factors in today's distribution network. We will separate these into two main groups, restrictions caused by the vessels and those caused by the fish farms. There are two main types of vessels in terms of how the feed is carried; the feed can either be bulked in silos or packed in large bags. For the silo-based vessels, the number of different feed types is limited by the number of silos, as feed types cannot be mixed in a single silo. On the other hand, vessels using large bags do not have the same constraint, as the content of the bags is not mixed. Furthermore, the vessels have a maximum load that sets an absolute upper limit for the amount carried. The final restriction caused by the vessels is their cruising speeds; different vessels have different cost-efficient cruising speeds, meaning that specific routes might be feasible for one vessel but not for another.

As for the fish farms, the most apparent implication on the network is the feed demand; as we comment further in Section 2.3.2, each fish farm has a known demand for each feed type, and this demand varies based on the season. Furthermore, the farms imply some restrictions for the vessels. Firstly, each farm has a time window it can receive deliveries. Secondly, some farms are not compatible with receiving the feed in bulk format and must thus be serviced by a vessel transporting the feed in bags. The location of the farms can also cause restrictions - some are located in fjords with too shallow water for the largest vessels, making visits impossible.

Finally, to prevent disease outbreaks, precautions have to be made by the vessels. Fish farms are categorized in three disease categories - *red*, *yellow* and *green* farms. Fish farms in which a disease outbreak is currently finding place belong to the red category. A region of a specified size surrounding the red farms is referred to as a *red zone*. Healthy farms located within this red zone belong to the yellow category. In other words, the yellow farms are healthy but located in a region where the risk of a future outbreak is considered higher than outside the red zones. Healthy farms located outside red zones are green. In general, green farms should be visited before yellow farms

and yellow farms before red farms. This is to minimize the risk of causing outbreaks in healthy farms. However, it is possible to visit a healthy farm after an infected one, but then the vessel has to wait for a specified *recovery time* to ensure that diseases are not brought to the healthy farm.

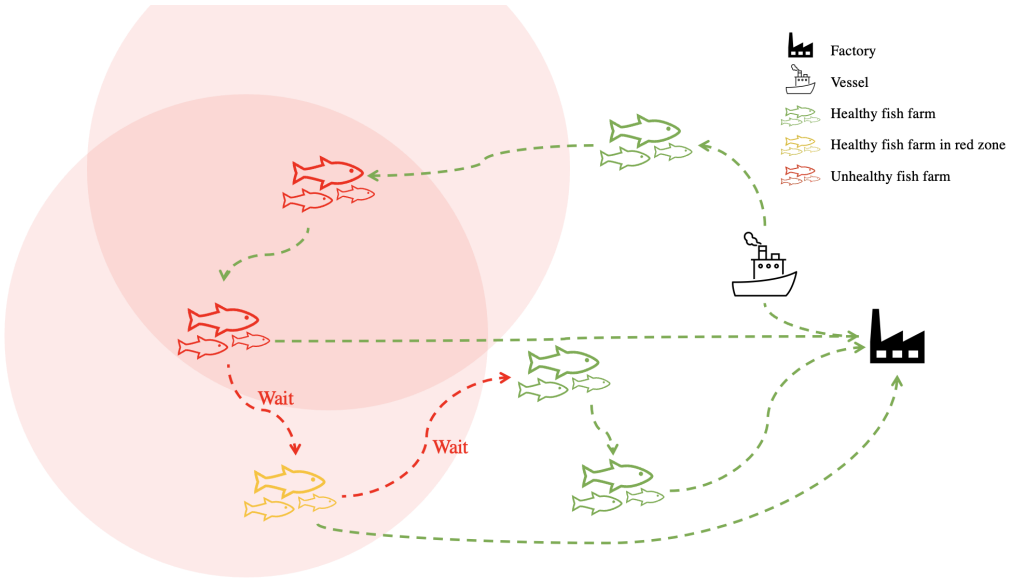


Figure 2.5: Example of route travelling to a red zone before all healthy farms are served, forcing it to either wait the recovery time or to return to the factory.

Figure 2.5 shows an example of a route that travels into the red zone before it has served all farms it is supposed to serve. Consequently, the planner must decide whether the vessel should wait the recovery time before continuing the route or return directly to the factory, which is permitted immediately after red farm visits. Figure 2.6 illustrates an example of a route avoiding this problem. The total distance traveled is longer than the previous route. However, waiting is avoided due to the different order of farm visits, and the total duration is less than in the original route.

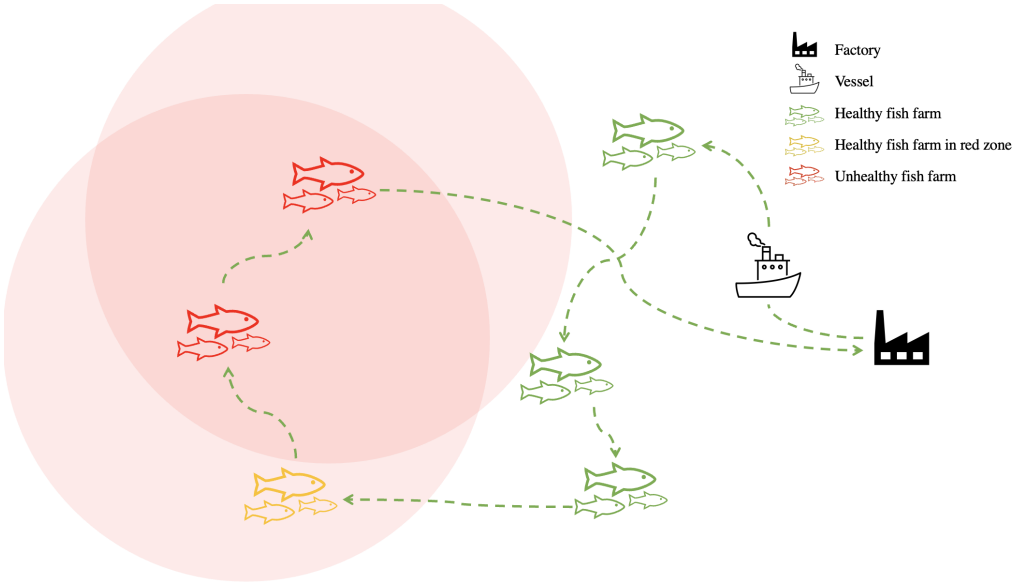


Figure 2.6: Route serving all healthy farms before it enters the red zone, thus avoiding all waiting.

2.3.2 Factors Impacting Fish Feed Demand

Demand for fish feed changes throughout the year. This can be due to biological effects affecting the amount of feed a fish will eat, planned events such as slaughter, or unexpected events such as disease outbreaks.

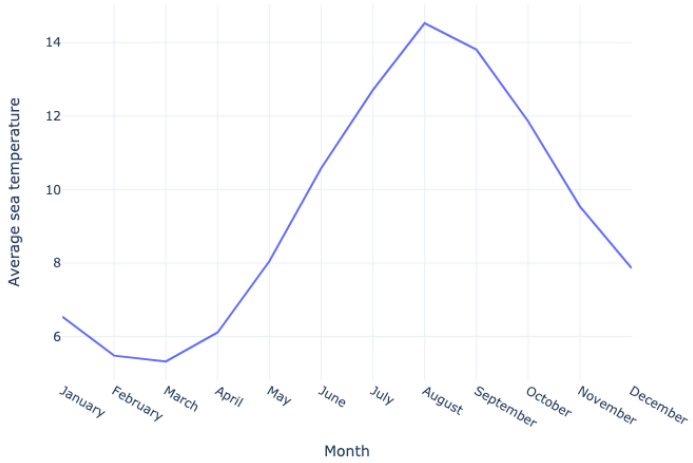


Figure 2.7: Sea temperature in Mowi's Norwegian production facilities. Average over five years.

Source: Mowi (2021b)

Most species of fish, including Atlantic salmon, are ectothermic. Their body does not actively attempt to remain at a specific temperature. Instead, the body temperature of a salmon will closely match the temperature of the surrounding water. It is common for ectotherms to grow slower when subjected to lower temperatures (Angilletta et al., 2004). This also applies to the Atlantic salmon, whose growth rate depends on abiotic[1] factors such as temperature and light (Austreng et al., 1987). The growth rate of a fish impacts how much feed it needs. Consequently, the demand for fish feed will change as the water temperature changes throughout the four seasons.

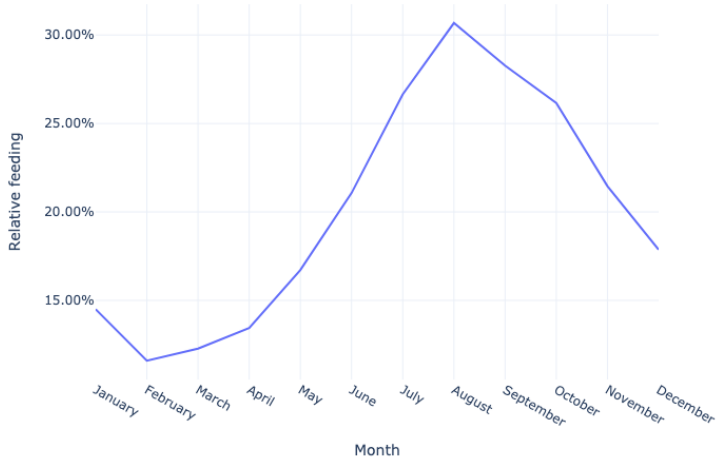


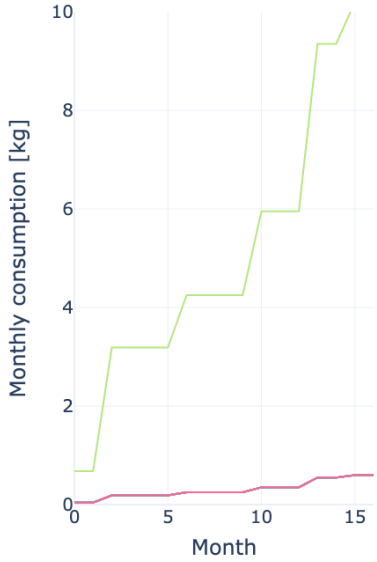
Figure 2.8: Consumption of feed relative to the total biomass, average 2016-2020.

Source: Mowi (2021b)

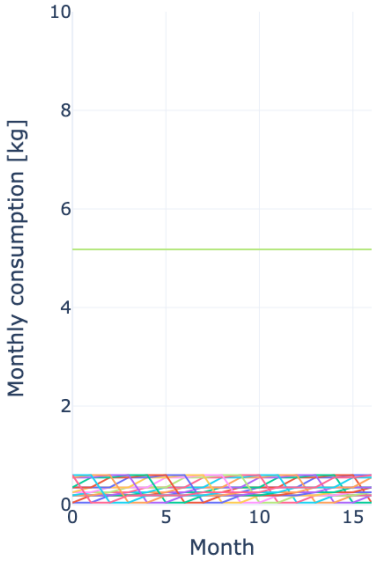
[1] Abiotic factors refer to nonliving factors in an environment, for instance, sea temperature.

According to Mowi (2021b), the optimal growth conditions for salmon are obtained in the temperature range of 8-14°C, but they will tolerate temperatures ranging from 0°C to 20°C. This is aligned with the results of Handeland et al. (2008), who found 14°C to be the temperature for which Atlantic salmon smolts have the highest growth rate. They also found feed conversion efficiency, the ratio of feed converted into meat, to be high between 10°C and 14°C, with a peak at 10°C. As shown in Figure 2.7, the sea temperature in Norwegian salmon farms varies from less than 6°C in winter to approximately 14°C in late summer. This impacts the feed consumption of the fish, as evident by the relative feeding for Mowi's farming facilities, shown in Figure 2.8. Consequently, this gives rise to a high and low season for fish feed, where feed demand during the low season can be as little as 30% of the demand during the high season (Mowi, 2021b). This creates challenges for the feed producers. Large quantities of feed with a limited shelf life must be intermediately stored to meet the demand in the high season, and scheduling the deliveries of the feed might prove difficult.

Furthermore, fish will eat feed at different rates depending on its growth stage. A fish requires more feed as it grows larger over time. This is shown in Table 2.1. To keep feed demand more stable throughout the year and to make full use of their quotas, farming companies will release smolt several times throughout the year (Hartvigsen, 2019). Figure 2.9 illustrates the difference in total feed demand depending on the timing of smolt releases for a set of farms. The figure is based on the data in Table 2.1, and sea temperature is thus not accounted for. As a result, despite giving the impression of constant feed demand, the demand in Figure 2.9b is still prone to seasonal variations. However, variations caused by having salmon in different growth stages are eliminated.



(a) Feed demand over time with synchronous smolt release. Simultaneous release of smolt causes the demand to spike significantly throughout the life span of the fish, as the monthly consumption for a fish increases throughout its life.



(b) Feed demand over time with asynchronous smolt release. Releasing smolt several times a year can be used to smooth out changes in total demand. In the later stages of growth, the high demand can be averaged out by having farms with early-stage fish as a counter-weight.

Figure 2.9: Illustration of a simplified scenario where 17 farms only grow one salmon each. In Figure 2.9a, one smolt is released simultaneously in all 17 farms. The lowest line represents the monthly consumption of every individual smolt, and as they are released simultaneously, their monthly consumption will always be identical. In Figure 2.9b, one farm releases one smolt every month, and when in a steady-state, a salmon is harvested from one of the farms every month. Since the smolts are released asynchronously, their monthly consumption will not be identical as in Figure 2.9a. In both Figure 2.9a and Figure 2.9b, the upper green line represents the total consumption across all farms, and it is clear that releasing smolts with an offset as in Figure 2.9b keeps the total monthly demand constant.

Growth intervals	**0.1 - 0.2 kg**	**0.2 - 1 kg**	**1 - 2 kg**	**2 - 3 kg**	**3 - 4 kg**	**4 - 5 kg**
Feed consum. [kg]	0.08	0.75	1.00	1.05	1.10	1.20
Time [months]	2	4	4	3	2	2

Table 2.1: The duration of, and total feed consumption in, the different stages of growth for Norwegian Atlantic Salmon.

Source: Mowi (2021b)

Sea temperature and a fish's growth stage are factors that need to be considered in combination with inventory costs and the feed's limited shelf life when production is planned. However, several discrete events can cause abrupt changes in fish feed demand. For example, demand at a farm will significantly decrease after harvesting. Since harvesting is usually known in advance, feed producers can incorporate this into their operational planning, potentially reducing distribution costs by avoiding unnecessary deliveries. Disease outbreaks also affect feed demand but cannot as easily be planned for. Unhealthy fish generally consume less feed, and large outbreaks might lead to the farmer deciding to harvest the salmon earlier than planned.

2.3.3 Current Distribution Planning

As mentioned in Section 2.3.1, not all vessels can visit all farms due to different delivery format requirements, i.e., bulk or bag. In practice, each vessel is assigned a subset of the farms, and the farms visited on a voyage are based on those farms' orders. Consequently, today's standard is mostly fixed routes for each vessel. On average, each farm needs a refill once or twice a week, depending on the consumption and storage capacity. A typical voyage, i.e., leaving and at a later point arriving at a production site, consists of 15 to 20 visits per vessel.

One of Mowi's vertical integration advantages is the data accessibility, allowing the logistics department to conduct precise analyses and create future demand forecasts for the individual farms. These allow production to be planned to produce a sufficient amount of feed. At least two weeks before the feed ought to be delivered, the orders are placed. The orders contain information regarding the farm, delivery time, and the quantity of the demanded feed types. When these orders are received, the orders can be used to generate a schedule for every vessel. Today, this is to a large extent, done manually. An initial schedule is set up well before the vessel leaves the production facility but is continuously changing in the period leading up to the final deadline for orders, four days before the vessel departs from the production facilities.

As mentioned in Section 2.3.1, all farms have a minimum inventory level that should be maintained at all times. This is typically equivalent to 1-2 days of feed and can be used when deliveries are delayed. Keeping a feed buffer is necessary as routes may change after the vessel has left the origin. If a farm detects a disease outbreak after the route has been set, scheduled visits might have to be skipped. Other factors, such as bad weather making docking of the vessels challenging, can lead to similar consequences. In general, the vessels should visit the skipped farms on the way back to the destination, as lost feed days incur a high cost for the farms.

2.3.4 Challenges in the Current Setup

As mentioned in Section 2.3.3, the current distribution setup heavily relies on farmers placing orders that are subsequently used in the planning process. Allowing farmers to place orders creates an incentive to place orders that are ideal for them at a local level. However, such orders might fit poorly into the distribution plan as a whole. For instance, a farmer might order significantly more feed in a specific time window than necessary, making it difficult for the planner to satisfy other orders along the vessel's route. Consequently, the created routes and schedules become suboptimal. In addition, Mowi reports that negotiations on specific orders, e.g., changing the quantities, are necessary relatively often, which in itself is very time-consuming. One farmer placing locally optimal orders might not significantly affect the entire distribution plan in isolation. However,

many locally optimal orders may do, and much flexibility is likely to be gained by transitioning to a VMI system. With a VMI setup, the planners can account for the entire system of farms when planning the distribution and therefore arrive at a better distribution schedule for Mowi as a whole.

Furthermore, today's setup depends on manual input; people in the planning department with specific domain knowledge use this knowledge to formulate the routes. There are multiple suboptimal aspects to this approach. Firstly, the dependency on manual work leaves room for human error. A miscalculation or hasty decisions can affect the feasibility of a route, e.g., that a vessel does not arrive at a farm within its time window. Another factor leading to human mistakes is the high dependency on communication between the farmers and the planners. They communicate through emails and phone calls, and misunderstandings can easily occur. Furthermore, due to the extensive use of personal domain knowledge, the process leading to a final route may seem like a "black box" for others than the planner. This means that controlling the work can be difficult. Previously, when there were fewer farms to serve, it was doable to maintain control and, to some degree, find close to optimal routes in terms of costs. However, as the network of farms has grown more complex and the feed processing industry now includes more feed types, the problem is far too complex to be dealt with manually. During high season, even avoiding farms running out of feed is challenging.

Finally, due to present bias[2] (Chakraborty, 2021), manual planning will generate plans that work well in the period close to the beginning of the planning horizon, but as time goes by, some decisions are likely to be suboptimal. This can lead to farms being under-supplied, and consequently, planned routes can be forced to be altered to ensure that the under-supplied farms do not run empty. As soon as this begins, the focus will move from supplying the farms in a cost-efficient way to just ensuring sufficient supply to all farms.

[2]Present bias is a cognitive bias leading humans to value payoffs closer to the present higher than those that will occur in the future.

Chapter 3

The Fish Feed Maritime Inventory Routing Problem

The Fish Feed Maritime Inventory Routing Problem (FFMIRP) is the problem of keeping a set of fish farms supplied with fish feed over a given planning horizon using a given fleet of fish feed vessels operated by the fish feed supplier. Keeping the fish farms stocked is thus the responsibility of the fish feed vendor, making this an application of VMI. The fish feed originates at a *production port* belonging to the vendor, which is a feed factory. Each *consumption port*, i.e., a fish farm, has a per time period consumption rate broken down by feed type. This consumption rate is assumed to be variable but deterministic and known in advance. In addition, each fish farm has the infrastructure required to hold a certain amount of inventory assumed to be known. This inventory is in the form of a set of silos with a given capacity that has been assigned to different feed types in advance. Furthermore, each farm has a per time period lower limit on the inventory of every feed type that serves as a safety buffer in the case of unforeseen events. The vendor has two options when supplying a farm. The feed can either be supplied by one of the fish feed supplier's vessels or by buying feed from the spot market at a predefined cost per quantity.

Each vessel has a known cost structure, in the form of a fixed cost per time period it is in use and an additional transportation cost for sailing between two ports, where a port is either a consumption port or a production port. The first can be considered the fixed cost for keeping the ship operational and in use, including the fuel cost when idling. The transportation cost is the extra marginal cost due to, e.g., increased fuel consumption during sailing. Each vessel has a predetermined cruising speed, determining how much time it needs to sail between two ports. Just like fish farms, each vessel has a specific storage capacity in the form of silos. In contrast to a farm, the silos on a vessel are not assigned to specific feed types and can thus transport any feed type at any time. However, a silo may only contain a single type of feed at a time. When arriving at a port, a vessel can load or unload a given quantity of feed in a fixed time, which may vary by port. Each port only has a limited number of berths available at given times, and a vessel is only allowed to dock for unloading or loading if there is a berth available.

Not all vessels are allowed to visit all ports, e.g., due to shallow waters. A vessel is assumed to be available from a given time at a specific port and can be situated at any farm or production port at the end of the planning horizon.

Furthermore, it is assumed that there is always sufficient feed supply at the production site to cover the farms' demands. We consider this assumption reasonable for two reasons. Firstly, at Mowi's production facility at Bjugn, the production level is close to the total capacity, implying that the gain from optimizing the production based on raw material prices and storage is limited. Secondly, the procurement, production schedule, and storage are not directly connected to the distribution, which is the main focus of this master's thesis.

With all of this given, we want to determine a distribution plan for the given planning horizon that minimizes the sum of the fixed costs, transportation costs, and spot market costs. In other words,

to determine the route and delivered quantities for all vessels, as well as when and how much to buy from the spot market throughout the planning horizon, while minimizing the total cost.

Chapter 4

Literature Review

The MIRP is a variant of the inventory routing problem (IRP), first described by Bell et al. (1983). Solving the combined routing and inventory problem allows for VMI, meaning that all inventory decisions are made by a central supplier and not by the consumer. It aims at reducing logistics costs by making decisions for products delivered to customers based on the current inventory levels and consumption rates (Coelho et al., 2014). Combining a traditional routing problem with inventory management, the IRPs are complex problems and have been of high research interest within the operations research field. MIRPs are a subgroup of the IRPs with some distinctive properties. The traditional IRPs, using land-based vehicles, often require the vehicles to return to their origins. Further, the trips[1] typically last for one day, and a homogeneous fleet of capacitated vehicles is usually employed. MIRPs often have more time-consuming voyages over longer distances, making it advantageous to allow them to end in different ports than where they started. Further, the properties of the vessels used often differ, and consequently, most problems consist of a heterogeneous fleet.

The MIRP has caught the interest of both academia and industry in the later years, as technological advances allow real-life problems to be solved, and the results have proven to enable monetary benefits (Christiansen et al., 2013). Consequently, comprehensive work within the field has been conducted, and there are numerous studies available. Even though the MIRP is well studied, there has not been done much work on the problem in the salmon farming industry. However, Brekkå et al. (2022) present a heuristic solving a problem combining a production scheduling problem and a rich vehicle routing problem (VRP) on a dataset similar to ours. The literature also includes a study on a single-product MIRP in the salmon farming industry by Agra et al. (2017). This section presents literature relevant to this thesis, focusing on different applications of the MIRP. The presented studies range from the pioneering study done within the area by Christiansen (1999) to the latest research within solution methods using matheuristics by Friske et al. (2022).

The literature review has been split into two main parts – a part where we look into the characteristics and different structures of the MIRP, and a part that explores existing solution approaches. The characteristics of the problem include the length of the planning horizon, how time is represented, how routing is handled, and more. When looking into solution methods, we first examine studies on exact solution methods before exploring some of the literature related to approximation approaches.

When we conducted the review, we used the search engine *Google Scholar*. To get an overview of the problem, we focused the search around two search phrases, namely "maritime inventory routing problem" and "ship routing and scheduling." Both phrases generated a significant number of studies relevant to the problem. To our knowledge, no studies have been conducted with a MIRP with the same characteristics as ours. Therefore, we researched a broad part of the MIRP literature to obtain ideas.

[1] A trip/voyage is defined as a single route for a vehicle, starting and ending in a depot, including visits to one or multiple consumption ports along the route.

In Section 4.1 we first give an overview and classify literature and research on the MIRP from the last two decades. Further in the section, we discuss the characteristics of different MIRPs, and what has been done previously. Further, in Section 4.2 we discuss existing literature on solution methods for the MIRP. First, we explore previous research on the problem using exact solution methods in Section 4.2.1. Then, we look into previous studies discussing approximation methods for solving the problem in Section 4.2.2. The research on approximation methods is divided into one part on heuristics and one on matheuristics. Finally, we discuss how our work contributes to the research on the MIRP in Section 4.3.

Significant parts from Section 4.1 are retrieved from our project thesis, Bjelland et al. (2021).

4.1 The Maritime Inventory Routing Problem

In this section, we investigate the existing literature on the MIRP in a structured way; we look into different decisions, and problem-specific characteristics researchers are faced with when modeling the problem. We have chosen to classify the studies along a set of dimensions to get a more structured walk-through of the them. The chosen dimensions are inspired by the study done by Christiansen and Fagerholt (2009), but also include aspects we found relevant in order to highlight how our work differs from previous studies. A summary of the previous studies is given in Table 4.1.

Table 4.1: Summary of previous studies on the MIRP. * – the study has used the MIRPLib benchmark library presented by Papageorgiou et al. (2014c) as part of its computational study.

Study	Type	Sol. method	Time	Network	Products	$\|\mathcal{I}\|$	$\|\mathcal{V}\|$	$\|\mathcal{P}\|$	$\|\mathcal{T}\|$ (u)
(Christiansen, 1999)	Short	Exact	Cont.	$N-N$	Single	16	5	1	36 (d)
(Ronen, 2002)	-	Exact	Discrete	$1-1$	Multiple	5+2	-	5	30 (t)
(Dauzère-Pérès et al., 2007)	Short	Heuristic	Discrete	$1-N$	Multiple	10+1	17	16	10 (d)
(Grønhaug and Christiansen, 2009)	Deep	Exact	Discrete	$N-N$	Single	3+3	5	1	30 (t)
(Grønhaug et al., 2010)	Deep	Exact	Discrete	$N-N$	Single	3+3	5	1	60 (t)
(Christiansen et al., 2011)	Short	Heuristic	Discrete	$N-N$	Multiple	49+12[2]	5	11	14 (d)
(Engineer et al., 2012)	Deep	Exact	Discrete	$N-N$	Single	6+4	6	1	14 (d)
(Stålhane et al., 2012)	Deep	Matheuristic	Discrete	$1-N$	Multiple	17+1	46	2	366 (d)
(Agra et al., 2013)	Short	Exact	Discrete	$N-N$	Single	6	5	1	30 (d)
(Song and Furman, 2013)	-	Matheuristic	Discrete	$N-N$	Single	5+5	8	1	60 (d)
(Uggen et al., 2013)	Deep	Matheuristic	Discrete	$N-N$	Single	10	5	1	180 (d)
(Agra et al., 2014)	Short	Matheuristic	Cont.	$N-N$	Multiple	7	2	4	180 (d)
(Papageorgiou et al., 2014a)*	Deep	Matheuristic	Discrete	$N-N$	Single	9+4	17	1	60 (t)
(Papageorgiou et al., 2014b)*	Deep	Matheuristic	Discrete	$N-N$	Single	9+4	17	1	60 (t)
(Agra et al., 2015)	Short	Exact	Cont.	$N-N$	Multiple	7	2	4	8 (d)
(Rakke et al., 2015)	Deep	Exact	Discrete	$N-N$	Multiple	15	25	2	90 (d)
(Hemmati et al., 2015)	Deep	Heuristic	Discrete	$N-N$	Single	30[3]	8	1	30 (d)
(Hemmati et al., 2016)	Short	Matheuristic	Cont.	$N-N$	Multiple	20	10	4	1440 (h)
(Agra et al., 2017)	Short	Matheuristic	Cont.	$N-N$	Single	60+1	2	1	10 (d)
(De et al., 2017)	Short	Heuristic	Discrete	$N-N$	Multiple	10	6	2	22 (t)
(Agra et al., 2018)	-	Matheuristic	Cont.	$N-N$	Single	6	5	1	60 (d)
(Friske and Buriol, 2018)*	Deep	Matheuristic	Discrete	$N-N$	Single	9+4	17	1	60 (t)
(Papageorgiou et al., 2018)*	Deep	Matheuristic	Discrete	$N-N$	Single	9+4	17	1	60 (t)
(Diz et al., 2019)	Short	Matheuristic	Discrete	$N-N$	Single	4+2	5	1	15 (d)
(Siswanto et al., 2019)	Short	Heuristic	Cont.	$N-N$	Multiple	2+1	2	2	25 (d)
(Friske and Buriol, 2020)*	Deep	Matheuristic	Discrete	$N-N$	Single	9+4	17	1	60 (t)
(Misra et al., 2020)	Short	Exact	Hybrid	$N-N$	Multiple	6+2	5	3	8 (d)
(Sanghikian et al., 2021)	Short	Matheuristic	Cont.	$N-N$	Multiple	18	15	9	60 (d)
(Friske et al., 2022)*	Deep	Matheuristic	Discrete	$N-N$	Single	9+4	17	1	60 (t)
This work*	**Short**	**Matheuristic**	**Discrete**	$N-N$	**Multiple**	**80+1**	**10**	**6**	**290 (h)**

The column "Type" indicates the type of the underlying problem. As in Hemmati et al. (2014), we distinguish between two types of MIRPs – *deep-sea* and *short-sea* problems. In deep-sea problems, the vessels travel long distances across at least one of the big oceans. On the other hand, short-sea problems are associated with shorter travel distances. The column "Sol. method" indicates whether the study solves the MIRP using an exact solution method, a heuristic, or a matheuristic. The column "Time" indicates if the problem is modeled using discrete or continuous time, or using a hybrid of the two, as done by Misra et al. (2020). All studies using discrete or hybrid time schemes

[2]Each port represents a single silo, and the ports are thus less complex than ports with demand for multiple products, as the silos are dedicated to a particular product.

[3]The ports are here cargoes, i.e., there are not 30 ports in the problem, but 30 cargoes to be delivered.

also have a variable consumption rate throughout the planning horizon, while the opposite is true for the studies using continuous time. The column "Network" indicates what the distribution network looks like; the first number indicates if there are one or multiple production ports, and the second indicates the same for consumption ports. The values in the column "$|\mathcal{I}|$" indicate the number of ports in the largest test instance. The number of production and consumption ports are separated with a '+' sign, where such a distinction is given. Columns "$|\mathcal{V}|$" and "$|\mathcal{P}|$" represent the number of vessels used and products demanded in the largest problem instance, respectively. Finally, column "$|\mathcal{T}|$" indicates the length of the planning horizon, either as a measure of time (d = day, h = hour) or as a number of time periods (t).

In general, all MIRPs are complex problems, but certain problem characteristics can either reduce or increase the complexity of the problem. In the remainder of this section, we walk through some crucial things to consider when formulating a MIRP. Section 4.1.1 discusses the length of the planning horizon and how the time is represented. Section 4.1.2 discusses single and multi-product problems. Finally, Section 4.1.3 looks into problems with single or multiple production facilities.

4.1.1 Planning Horizon, Time and Consumption Rates

When solving a MIRP, the length of the planning horizon is an essential part of the problem. As mentioned, we separate between short-sea and deep-sea problems. As a consequence of the traveling distances, decisions are usually made more frequently, and the planning horizon is usually shorter, in short-sea problems than in deep-sea problems. The difference in frequency is a direct consequence of the fraction of the time spent traveling between ports in the two problem types; in deep-sea problems, the time spent loading and unloading is often diminishing compared to the time spent traveling. Stålhane et al. (2012) propose a matheuristic for a deep-sea problem generating an annual delivery plan (ADP) for a liquefied natural gas (LNG) company. The problem is solved with a planning horizon up to 366 days. Similarly, Rakke et al. (2015) consider an ADP for LNG, and solve it for a set of time horizons between three months and a year, and Uggen et al. (2013) solve a set of LNG MIRPs with time horizons of 180 days. Further, the studies by Agra et al. (2018) and Friske et al. (2022) are also solving deep-sea problems, both with a planning horizon of 60 days. Agra et al. (2017), Dauzère-Pérès et al. (2007), De et al. (2017), Diz et al. (2019) and others consider short-sea problems, and the planning horizons in these problems are usually significantly shorter than the studies previously mentioned. Our problem is a typical short-sea problem.

The time can be handled in three ways – continuous, discrete, or a combination of the two. A continuous time scheme is typically applied to problems where parameters and other conditions remain constant throughout the planning horizon. This scheme is used by Christiansen (1999), Agra et al. (2014), Hemmati et al. (2016), Sanghikian et al. (2021) and others. Models using continuous time often use a visit count for each vessel-port combination to keep track of the sequence of port visits. This is how e.g., Hemmati et al. (2016) and Agra et al. (2017) modeled it. Introducing discrete time models increases the number of variables and constraints but allows the model parameters to vary with time. Thus, in problems where demand, costs, or any other parameters are time-dependent, discrete time is often preferred. As can be seen in Table 4.1, discrete time schemes are often used. Also, note that most discrete time studies have solved the problem using some approximation technique rather than an exact solver. In the studies considering the MIRPLib benchmark instances, introduced by Papageorgiou et al. (2014c), a discrete time scheme is employed. This includes the studies by Papageorgiou et al. (2014b) and Friske et al. (2022) among others.

Misra et al. (2020) use a hybrid time scheme, combining continuous and discrete time. The idea is to divide the planning horizon into n events, and within each event, there are multiple timelines. The timelines dealing with variables changing continuously are handled in that way, while discrete events, such as when a jetty in a port becomes available, are discretized.

As mentioned, continuous time schemes are often preferred when the problem's parameters are assumed to remain constant. In short-term planning problems, this assumption simplifies the problem without significantly impacting the model's ability to reflect reality. The integrality bounds are

often tighter when using discrete time. However, Agra et al. (2014) argue that when the consumption rates are close to constant throughout the planning horizon, the decreased tightness of the model is compensated by decreased complexity and running time. On the other hand, in cases where the consumption rates are expected to fluctuate throughout the planning horizon, a discrete time scheme is preferred, which is the case in our problem.

4.1.2 Single-Product or Multi-Product

Christiansen (1999) covers a fleet of vessels transporting ammonia, which is the only product produced at the production facilities, and thus, the problem only consists of a single product. The same goes for Diz et al. (2019) who handle the distribution of crude oil for a Brazilian oil producer. In other studies, simplifications are made in order to get a single-product problem, this is the case in Agra et al. (2017). The study discusses the salmon feed distribution for Mowi (then Marine Harvest) but due to lack of information on feed types, it operates with a single product. Further, the MIRPLib benchmark instances consist of a single product.

Christiansen et al. (2011) consider a problem where multiple grades of cement are considered; each port in the problem represents a specific silo with a known time-dependent consumption of a given cement grade. This is similar to our problem, where different feed types must be kept apart. When there are multiple products, there are different ways of handling the allocation of compartments on vessels and consumption ports. Hemmati et al. (2016) consider a multi-product problem, but does not consider the allocation of the products, and thus, different products are allowed to be mixed within a single compartment according to the model. Another way of handling the allocation of different products is to use designated compartments for each product. This is done by Agra et al. (2014) where different oil products have pre-determined compartments on the vessels and consumption ports throughout the planning horizon. Finally, the allocation of compartments onboard vessels can be done as part of the optimization problem, as done by Christiansen et al. (2011) and in our project.

4.1.3 Distribution Network and Routing Constraints

The distribution network can either be *one-to-one*, *one-to-many* or *many-to-many*. In a one-to-one network, all voyages consist of one production port and one consumption port. In a one-to-many network, all voyages start in the same production port but can visit multiple consumption ports. Finally, in a many-to-many network, there are multiple production and consumption ports, and one or more of them can be visited during a voyage. Ronen (2002) addresses a one-to-one MIRP for liquid products transported in bulk, where each voyage consists of one origin and one destination. Dauzère-Pérès et al. (2007) consider the distribution of Norwegian calcium carbonate slurry to multiple European customers. The company has a single factory where all voyages begin; the problem is thus a one-to-many problem. Similarly, Stålhane et al. (2012) consider a case from the distribution of LNG with a single production port and multiple consumption ports. As mentioned previously, due to the distance traveled in most sea-based voyages, allowing vessels to begin their voyage in one port, and end it in another, can often prevent vessels being forced to travel long distances without any load. Consequently, most previous studies employ a flexible model allowing this.

There are many configurations of the many-to-many networks in terms of routing constraints; we have distinguished these networks along two dimensions – constraints regarding initial load and initial position for the ships. Hemmati et al. (2016) assume that all ships are empty at the beginning of the planning horizon, and consequently, must travel directly from their initial position to a loading port. Other studies allow an initial load on the vessel, and as such, the voyages can begin from unloading ports, given a positive load. While Agra et al. (2017) assume that all ships begin the planning horizon at a port and thus are available from the beginning of the planning horizon, Friske et al. (2022) allow the ships to be located anywhere at initialization. The ships must then travel from their initial positions to a port before becoming available. We employ a

model similar to that described by Friske et al. (2022); the network has a many-to-many structure[4] and the vessels are allowed to carry an initial load, and they are not required to be located at a port at the start of the planning horizon but do not become available before arriving at one.

4.2 Solution Approaches for the Maritime Inventory Routing Problem

Since the MIRP was first discussed, researchers have investigated numerous approaches for solving it. An observation pointed out in several papers is that problem instances of real-life size are too complex to be solved using an exact solution method. However, different mathematical models for the MIRP have been developed, and some of these are discussed in Section 4.2.1. Given the complexity of the problem, approximation techniques for solving it have been widely exploited. We present some of the novel work in these areas in Section 4.2.2.

4.2.1 Exact Methods

Particularly in the early studies on the MIRP, much effort was put into formulating mathematical models to describe and solve the problem. In the pioneering work by Christiansen (1999), the Combined Inventory and Time Constrained Ship Routing Problem, which is similar to a MIRP, is modeled using an arc-flow (AF) model. In the model, each node corresponds to a particular visit to a harbor, i.e., a harbor can be visited multiple times. Arcs connect the nodes, and each arc corresponds to a vessel traveling from one harbor visit to another.

As mentioned in Section 4.1.1, time-dependent parameters often imply a discrete time scheme. Song and Furman (2013) proposed, and Papageorgiou et al. (2014b) and Friske et al. (2022) later applied, an AF model over a time-space (TS) network, which can be viewed as an integer multi-commodity network flow formulation. The vessels constitute the commodities, while the nodes represent a visit to a port at a specific time, i.e., there is one node for each combination of port and time. The arcs in the network represent feasible moves for a particular vessel, and constraints balance the flow over the arcs to ensure feasible routing. The set of nodes is equivalent for all vessels, while each vessel has one set of arcs representing the feasible traversals for the particular vessel.

Another formulation that was introduced early was the path-flow (PF) model. Christiansen (1999) reformulates and decomposes the AF model of the problem using a Dantzig-Wolfe decomposition to obtain a PF model. The PF formulation consists of two phases – in the first, vessel schedules are generated, and in the second, a feasible set of these schedules is selected. The problem is decomposed into subproblems for each vessel and each harbor inventory. Columns are then generated in the subproblems until no further improving columns can be generated. The remaining constraints are kept in the master problem.

Grønhaug and Christiansen (2009) formulate both a PF model and an AF model. The PF model enumerates all possible routes and then solves a mathematical model for maximizing the profit for an LNG-based problem, ensuring that the selected routes constitute a feasible solution. In the AF model, all vessels have an associated network where the nodes consist of a port. A binary routing variable is used to ensure that the routing is feasible. The study also includes a computational study of the two models, comparing their performance. Results indicate that commercial mixed-integer programming (MIP) solvers struggle to solve the two exact models when the problem size becomes larger; on a test instance with three vessels, four ports, and 60 time steps, neither formulations could solve the problem in ten hours.

Agra et al. (2013) argue that the bounds provided by the AF model are weak and thus propose a strengthened formulation – a fixed charge network flow (FCNF) formulation. The MIRP is

[4]Even though the data provided by Mowi only contains one production port, yielding a one-to-many network in the particular case, our model is flexible and can handle multiple production ports.

formulated as a single-commodity FCNF, and several constraints are added to strengthen the formulation further. The FCNF formulation provides better bounds than the AF model, according to the computational results. Friske et al. (2022) also compared the two formulations by employing them in a Relax & Fix and Fix & Optimize matheuristic; the results from this study substantiates the findings by Agra et al. (2013) – the FCNF formulation provides better bounds than the TS model.

4.2.2 Approximation Methods

As evident from the computational study completed on a MIRP considering routing of LNG done by Grønhaug and Christiansen (2009), the MIRP is a too complex problem for being solved with exact models – particularly when the problem size becomes realistic. This makes approximation techniques appropriate for solving the problem. While most of the pioneering studies on the MIRP focus on describing it by formulating mathematical models, most recent studies focus on developing heuristic algorithms for solving real-life problems. First, we look at some of the heuristics that different authors have formulated before looking closer into studies combining heuristics with mathematical models solved by commercial MIP solvers, i.e., matheuristics.

Heuristics

Christiansen et al. (2011) formulate a construction heuristic for solving a problem in the Norwegian cement industry with multiple products. The heuristic works by iteratively selecting a silo to be served and a silo to serve it based on some greedy criteria. Then a vessel is chosen to transport the cement, ensuring that no different cement grades are mixed in the compartments. Iteratively shipments, constituted by an origin silo, a destination silo, a vessel to transport, cement quantities, and a time for departure from the origin and arrival at the destination, are added to the plan. Finally, a genetic algorithm is used to adjust the weights used for selecting shipments.

Hemmati et al. (2015) consider a routing and scheduling problem for a tramp shipping company and formulate a two-phased heuristic that converts the cargo and inventory problem into a traditional cargo routing problem. This is done by first transforming the inventories into cargoes. These cargoes have predefined origins and destinations, time windows, and quantities. With the cargoes fixed, the cargo routing problem is solved using an Adaptive Large Neighborhood Search (ALNS). The heuristic then alternates between updating the cargoes and solving the ALNS.

Further, De et al. (2017) formulate an evolutionary algorithm to solve a MIRP with multiple products and discrete time. A Particle Swarm Optimization for Composite Particle (PSO-CP) model is employed. The PSO-CP consists of a swarm of composite particles, an extension of the traditional PSO designed to avoid immature convergence and entrapments in local optima. The particles consist of all variable types from the mathematical model of the MIRP. Three particles are grouped to form a composite particle based on proximity in Euclidean distance in variable space. The composite particles in the swarm then interact with each other to locate the global optimum. During an iteration, the position of a particle is updated based on its best-known position and the current global best solution.

Siswanto et al. (2019) also develop a heuristic for the MIRP with multiple time windows and products for a national oil company in Southeast Asia. A multi-heuristic genetic algorithm is proposed and consists of five main steps: (1) a vessel to be routed is selected, (2) the routing of the vessel is decided, including a supply port and a number of demand ports, (3) the compartment allocation and loading quantities are decided, (4) the unloading quantities are decided, and (5) the fitness of the solution is calculated. As the name suggests, different heuristics are used for making decisions (1)-(4), and these strategies constitute the chromosomes of the population.

Matheuristics

Pure heuristics have proven efficient for solving problems with binary variables. For most variations of the VRP, heuristic approaches perform best; Christiaens and Vanden Berghe (2020), and Vidal (2022) present the current state-of-the-art algorithms for the VRP. However, when continuous variables, like the inventory variables in the MIRP, are introduced, heuristics not incorporating some mathematical model often fall short. If we investigate the benchmark instances introduced in Papageorgiou et al. (2014c), all new best-known solutions since the publication have been discovered by matheuristics. In this section, we look into the existing solution methods using matheuristics to solve the MIRP based on the classification of matheuristics used for routing problems by Speranza and Archetti (2014). They classify the matheuristics into three classes:

1. *Decomposition approaches* – the problem is divided into smaller and less complex subproblems. The subproblems are solved using a MIP solver to optimality or suboptimality. In algorithms with a decomposition matheuristic, the matheuristic is normally the main component of the algorithm. The decomposition approaches are divided into Two-Phased, Partial Optimization, and Rolling Horizon (RH) approaches.
2. *Improvement heuristics* – improves a solution found by another heuristic approach using MIP solvers. Improvement heuristics are widely used as they can be applied to any problem independently of the initial solutions generated. The improvement heuristics typically play a "minor" role in the overall algorithm and are intended to improve already generated solutions. The class is further divided into one-shot approaches and approaches using MIP models for local optimization.
3. *Branch-and-price/column generation-based approaches* – consist of a master problem in which the routes (columns) to be used to minimize costs are selected and a subproblem that generates the routes (columns). In the matheuristic, the exact method is modified to reduce running time and does not guarantee optimality.

The focus in this section is put exclusively on the two first classes, as these are the two most used for inventory routing problems.

Decomposition Approaches

RH approaches are popular time-based decompositions techniques used by several authors. Agra et al. (2014) and Papageorgiou et al. (2018) both employ an RH decomposition to solve the problems at hand. Agra et al. (2014) divide the current planning horizon into three parts – a frozen part where all binary variables are fixed, a central part where neither restrictions nor relaxations are made, and a forecasting period, where binary variables are relaxed. A MIP is then solved with the above restrictions, and the central period in iteration n becomes frozen in period $n+1$. The study also proposes two additional procedures used in combination with the RH. First, a local branching restricts the number of variables allowed to change in search of a local optimum, starting with a feasible solution. Secondly, a feasibility pump heuristic seeks to obtain a feasible solution by rounding the values of relaxed binary variables.

To improve the solutions generated by the RH, Papageorgiou et al. (2018) discuss a set of frequently seen improvement heuristics for MIRPs. From the computational study, it is evident that a hybrid between an RH and a K-opt local search provides promising results for the hard problems in the study. In this approach, the RH is run for $T/2$ time periods before a K-opt local search that fixes the routes for all but K vessel classes[5] is run. Then the RH is run for the rest of the time periods before the K-opt is called again.

Another time-based decomposition is Relax & Fix (R&F), a technique that shares most of its characteristics with the RH. The main difference between RH and F&R is the length of the planning horizon considered in each iteration. In RH, only a part of the problem is considered. In contrast, the entire planning horizon is considered in the R&F. Uggen et al. (2013), Friske and Buriol (2018)

[5] In the benchmark instances proposed by Papageorgiou et al. (2014c) all vessels belong to a particular vessel class with similar properties.

and Friske et al. (2022) use F&R to generate initial solutions; the traditional F&R divides the planning horizon into n subhorizons, constituting n subproblems, which are solved iteratively by relaxing integer constraints. Similar to the RH approaches mentioned, the planning horizon is split into different parts. In addition to the fixed block, the central part, and the relaxed forecasting period, a part of the end, called the End Block, is either cut off or dealt with using a simplified mathematical model, reducing the number of variables.

To improve the initial solutions, Uggen et al. (2013) propose another time-based decomposition; the planning horizon is again split into m subhorizons. m is typically smaller than the n subhorizons used for the R&F. The subhorizons are iterated through, fixing all integer variables in future subhorizons. All variables in the current subhorizon and the continuous variables in the future subhorizons are optimized. Friske and Buriol (2018) and Friske et al. (2022) develop similar Fix & Optimize (F&O) algorithms with some extensions. In addition to decomposing the variables based on time, Friske and Buriol (2018) also generate neighborhoods where all integer variables except those associated with two vessels were kept fixed. The vessel pairs are selected iteratively until no further improvements are found. The study also includes a decomposition combining the time and vessel decompositions. The planning horizon is split into m intervals, and while iteratively optimizing the intervals, the variables associated with one vessel are optimized. Then, a new vessel is optimized with the same interval, giving a total of $m|\mathcal{V}|$ iterations. In Friske et al. (2022) the F&O is further extended with one more decomposition strategy – the variables are decomposed based on the associated port type. First, all discharging port variables are unfixed; then, all loading port variables are unfixed. The variables connecting the two port types are unfixed in both subproblems.

Papageorgiou et al. (2014b) present a two-phase algorithm for solving a single-product deep-sea MIRP. The main idea behind the decomposition is to first "zoom out" from the problem aggregating information before "zooming in" and solving a series of subproblems. In the "zoom out" phase, ports are aggregated to regions and vessels to vessel classes. The aggregated model is solved, producing a solution indicating the number of visits from each vessel class to each region and each visit's entering and leaving times. The vessels are then assigned to the regional visits using a first-in, first-out principle. In the "zoom in" phase, a subproblem is solved for each region on an augmented TS network, deciding the routes and loading and unloading decisions. After the subproblems have been solved, a local search is used to remove infeasibilities and improve the solution. The study also proposes some valid inequalities and branching decisions to reduce the number of infeasible solutions.

In Papageorgiou et al. (2014a) a novel solution approach using approximate dynamic programming (DP) to solve a deep-sea MIRP with a long time horizon is proposed. They define a mathematical formulation, as well as a dynamic programming formulation. The DP formulation uses a state representation consisting of two vectors: one for current and future vessel positions, and one for current and future inventory levels. The DP value function is approximated using a piecewise linear function whose value only depends on the future inventory levels at discharging ports, and whose value is determined by solving a MIP. At each step of the approximate DP, the value function approximation is updated based on the occurrences of stockout in the previous solution. In addition, the approximate DP is supplemented with a local search to find even better solutions.

Hemmati et al. (2016) convert the MIRP into a less complex problem not involving inventory decisions, a pure cargo routing problem, by transforming the inventories into cargoes. The cargoes are generated using two mathematical models. First, a transportation model generates a set of cargoes (quantity, origin, and destination), guaranteeing that all ports have a sufficient amount of all products loaded or unloaded during the planning horizon. Then, a time window model maximizes the time window in which cargoes can be delivered while ensuring no inventory violation occurrences. When all orders are generated, an ALNS is used to solve the ship routing problem. Further, the quantities in the orders are updated based on information from the ALNS.

Agra et al. (2017) present a decomposition-based matheuristic for a problem similar to Mowi's within the Norwegian salmon farming industry. First, a mathematical model using branch-and-cut is run on an AF formulation. Suppose a feasible solution can be found by the exact method. In that case, the solution is decomposed into a traveling salesman problem for each vessel with time

windows, fixing unloading quantities and adding the unloading times to the sailing time of the route. When a feasible solution is not retrieved from the exact model, an R&F model considering one vessel at the time is deployed to build each vessel's route.

Diz et al. (2019) propose a robust variation of the R&F and F&O algorithm for a MIRP where the time spent by vessels at ports is considered uncertain. A decomposition based on ports is done for the R&F phase. Through iterations, a subset of the ports is selected. All variables not associated with the selected ports are relaxed. In the F&O phase, the problem is decomposed on vessels rather than ports. The matheuristics were applied on an extended FCNF formulation and executed for different levels of robustness, depending on how conservative the estimations of time spent by the vessels in ports were. Finally, a Monte Carlo simulation was run to estimate the probability of the obtained solutions being infeasible for each robustness level.

Improvement Approaches

Stålhane et al. (2012) formulate a construction and improvement heuristic for creating an ADP for an LNG company, complemented by a branch-and-bound algorithm on a MIP model. The proposed solution uses a multi-start approach where a greedy construction heuristic generates several initial solutions by first selecting a contract to be served before a vessel is selected. Finally, the first day the vessel can serve the contract is found. An artificial node is used for adding some randomness and thus, allowing different solutions to be generated. After the set of initial solutions is generated, a local search using different neighborhood operators and a restricted version of the exact model of the problem is applied to improve the solutions. The restricted MIP works by generating a reduced set of variables where multiple factors are kept fixed, creating a problem that is a fraction of the size of the original problem.

In the study by Song and Furman (2013), a solution technique for solving MIRPs with practical features is introduced. First, a preprocessing reducing the size of the search space is completed. Then, a feasible integer solution to the problem is found by running a mathematical model. If the model cannot find a feasible solution, it is replaced with a feasibility pump. A Large Neighborhood Search (LNS) is applied from the initial feasible solution, generating small subproblems that are solved using a MIP model to improve the initial, feasible solution. In the LNS, all binary variables except those associated with a vessel pair are fixed. An AF model then solves the subproblem.

Agra et al. (2018) propose a robust solution to a MIRP where the sailing times between ports are considered uncertain. To ensure robustness against delays in sailing time, the problem is decomposed into a master problem and a subproblem. In the master problem, the constraints associated with sailing time are considered for a set of sailing time scenarios. The subproblem then verifies if the solution is feasible for the remaining sailing time scenarios; if not, more scenarios are added to the master problem until a feasible solution is found. The study proposes a matheuristic based on a local search to solve larger instances, which uses a conservative solution generated by the decomposed model as its starting point.

Friske and Buriol (2020) introduce a one-shot matheuristic approach to solve the instances introduced by Papageorgiou et al. (2014c). The first parts of the algorithm are purely heuristic. A greedy construction heuristic generates a set of solutions with randomness. Deliveries are generated by selecting (1) urgent ports based on inventory violation times, (2) counterpart ports of opposite type based on proximity to the urgent port and violation time, or at random, (3) a vessel to perform the voyage, and (4) delivery time and quantities. This is done until all inventory violations are removed. To improve the solutions, an LNS destroys the routes of a set of vessels before rebuilding them using the greedy construction heuristic. After the best solution from the LNS has been retrieved, a reduced MIP, keeping all variables except inventory variables, spot market variables, and some other continuous variables fixed, is run, potentially improving the solution.

Sanghikian et al. (2021) present an algorithm to solve a real-life multi-product MIRP faced by a vertical integrated offshore oil and gas company in Brazil. First, an initial solution considering only the vessel's initial positions is constructed. Then, the routing of the vessels is done iteratively by an LNS, while an LP is used for optimizing inventory levels. The solution is generated in a loop that uses four different operators to generate neighborhoods. For each neighborhood, the inventories are solved to optimality by the LP. If the new solution is better than the old best solution, it is

accepted, or by some probability, if it is not better.

Table 4.2: Summary of previous studies on matheuristics developed to solve the MIRP.

Study	Model	B	V	R	Construction	Improvement	Classification
(Stålhane et al., 2012)	Voyage-based				-	F&O	Improvement
(Song and Furman, 2013)	TS	•	•		-	F&O	Improvement
(Uggen et al., 2013)	AF				R&F	F&O	Decomposition
(Agra et al., 2014)	AF	•	•		RH	-	Decomposition
(Papageorgiou et al., 2014a)	TS				TD	F&O	Decomposition
(Papageorgiou et al., 2014b)	TS	•	•		TD	F&O	Decomposition
(Hemmati et al., 2016)	AF				TD	ALNS[6]	Decomposition
(Agra et al., 2017)	AF	•	•		R&F	TD	Decomposition
(Diz et al., 2019)	FCNF			•	R&F	F&O	Decomposition
(Agra et al., 2018)	AF		•	•	-	MIP LS	Improvement
(Friske and Buriol, 2018)	FCNF		•		R&F	F&O	Decomposition
(Papageorgiou et al., 2018)	TS	•	•		RH	F&O	Decomposition
(Friske and Buriol, 2020)	FCNF				-	F&O	Improvement
(Sanghikian et al., 2021)	AF				-	F&O	Improvement
(Friske et al., 2022)	TS, FCNF		•		R&F	F&O	Decomposition
This work	**TS**[7]				**-**	**F&O**	**Improvement**

A summary of the studies discussed in this section is given in Table 4.2. The column "Exact model" indicates the formulation used for the underlying model used for the matheuristic. As can be seen, all studies use an AF (or TS) or an FCNF formulation. The difference between the AF and TS formulations is that the TS formulation consists of nodes with both a time and a port, while the ordinary AF formulation has a network of ports, or port and visit pairs, only. Thus, most studies using a discrete time scheme use the TS formulation, while those with continuous time use the AF. The columns "B," "V," and "R" indicate if the studies introduce some branching, valid inequalities, or robustness to handle uncertainty, respectively. Further, the columns "Construction" and "Improvement" indicate the matheuristic techniques used for constructing initial solutions and improving them, respectively.[8] A summary of the matheuristic techniques mentioned in Table 4.2 with explanations of what is included in each technique is given in Table 4.3. Finally, the column "Classification" indicates if the main matheuristic proposed in the study is classified as a decomposition or an improvement heuristic, according to the classes presented by Speranza and Archetti (2014). As can be seen, our algorithm is classified as an improvement matheuristic – the commercial solver is solely used for assigning quantities given a complete routing solution to the problem. Thus, we argue that the LP plays a minor role through local optimization, with the heuristic doing most of the work. This is discussed further in Chapter 6.

Table 4.3: Matheuristic techniques presented in the chapter.

Matheuristic	Description
R&F	*Relax & Fix* – includes all strategies that consider the entire planning horizon and splits the planning horizon into parts were some variables are relaxed and other are fixed.
RH	*Rolling Horizon* – includes all strategies that split up the planning horizon without considering the entire planning horizon at the same time.
TD	*Tailored Decomposition* – includes all strategies that use some tailored decomposition of the problem into subproblems based on the structure of the problem.
F&O	*Fix & Optimize* – includes all strategies that use some operator to generate neighborhoods where the neighborhoods are used to fix certain variables. I.e., heuristics like ALNS and other local searches followed by a MIP are included in this technique.
MIP LS	*MIP-based Local Search* – proposed by Friske et al. (2022) and includes all strategies using a local search with a MIP embedded on a problem with uncertainty.

[6] In the work by Hemmati et al. (2016) the ALNS used for improving the solutions generated by the construction matheuristic is just focusing on the routing and not using a mathematical model, and is thus a pure routing heuristic.

[7] In our work the exact model solved for the time-space network is not a part of the matheuristic, rather other linear programs are used.

[8] Note that the column "Improvement" is not the same term as from the classification by Speranza and Archetti (2014), but rather an indication of the technique used for improving the initial solutions.

4.3 Our Contribution

In this section we present the contribution to the MIRP research from this thesis. Our contribution is two-fold – we have formulated an extended mathematical model describing the MIRP faced by our industrial partner, Mowi, and we have developed a novel algorithm for solving real-life MIRPs of substantial complexity.

Even though the MIRP is a well-studied problem and there has been proposed numerous mathematical models describing the problem, the thesis presents a new model in Chapter 5. The model is an extended version of the TS network model first introduced by Song and Furman (2013). Our problem is to our knowledge the first MIRP formulated with the combination of discrete time, multiple products, and a fleet with heterogeneous, non-dedicated compartments. As the products cannot be mixed, the model also incorporates allocation of compartments, which has not been modeled for the TS model previously.

The proposed matheuristic, named SMOLT, uses a memetic algorithm to decide how to route each vessel, and combines this with a continuous LP for optimal quantity assignment. For routing, we use a novel representation consisting of a list of $(port, time)$-pairs for each vessel. Our solution method implements a series of mutations operating on routing solutions. This includes a set of ruin & recreate methods (R&R), among these an adaption of Slack Induction by String Removal (SISR) proposed by Christiaens and Vanden Berghe (2020) to the MIRP. SMOLT has been designed to be embarassingly parallel[9], allowing it to be scaled out to nearly arbitrary levels of parallel execution. This allows us to solve larger problems than those that have been addressed in literature so far.

[9] A problem where minimal effort is required for separating the problem into a independent tasks that can be run in parallel.

Chapter 5

Mathematical Formulation

This section presents the MIP model for the FFMIRP. The model is inspired by, and extends, the MIP model first formulated by Song and Furman (2013), and later used by Papageorgiou et al. (2014c), and describes the same problem as the matheuristic presented in Chapter 6 solves. In Section 5.1 we present the underlying modeling assumptions and relevant data structures. This includes how fish farms, production facilities, and vessels are represented and several decisions and assumptions related to modeling the specific problem at hand. In Section 5.2 we present the notation used in the model. Finally, we present the model and explain it in Section 5.3.

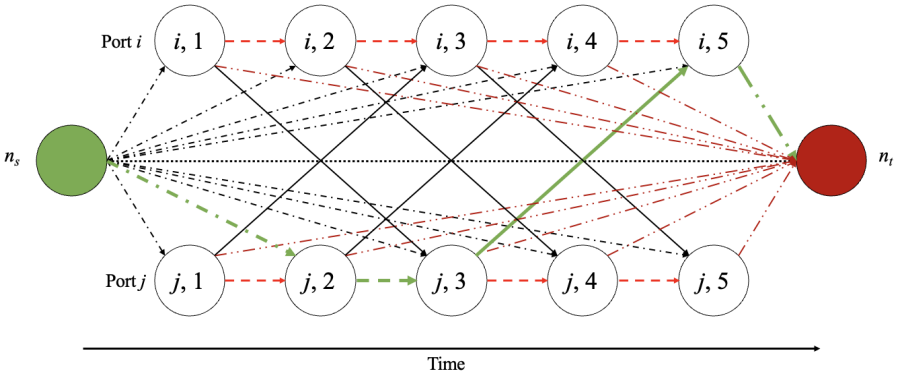
5.1 Modeling Approach and Assumptions

This section presents the foundation and assumptions on which our mathematical formulation is based. In Section 5.1.1 we elaborate on our representation of time. Then, in Section 5.1.2 we present the chosen port representation of fish farms and factories and introduce the TS network structure used in the model. In Section 5.1.3 and Section 5.1.4, we discuss the modelling of inventory management in vessels and in ports, respectively. Lastly, our routing assumptions are presented in Section 5.1.5, and measures to avoid adverse end-effects are discussed in Section 5.1.6.

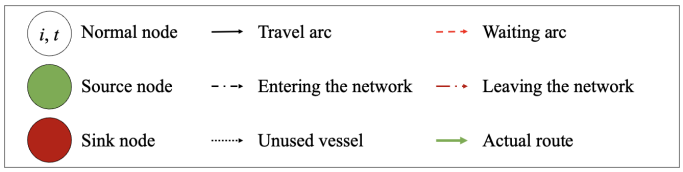
5.1.1 Discrete Time and Planning Horizon

The model is formulated using discrete time period, and the planning horizon is represented as a sequence of consecutive time periods of equal length. The discretized time allows the feed consumption rate at a farm to vary throughout the planning horizon. Varying consumption rate is necessary to incorporate events such as harvesting which can dramatically affect consumption within a planning horizon.

When using discrete time periods, an activity like sailing or unloading feed is started in a time period and will last for a given number of time periods. Therefore, the length of the time periods is crucial for the model's accuracy. The length is not explicitly defined in the mathematical formulation but must be stated when solving problem instances. The model's accuracy decreases as the length of time periods increases since longer time periods increase the error between actual and modeled duration for sailing and loading. However, while shorter time periods lead to more accurate results, it does so at the expense of making the model harder to solve. This is due to an increase in the number of constraints and variables. Therefore, choosing the duration of the time periods is a trade-off between the accuracy of the results and the difficulty of solving the model. For the problem at hand, i.e., Mowi's problem, having a time period of one hour results in activities lasting several time periods. Therefore, the inaccuracies caused by the discretization of time are believed insignificant enough to support our decision to use a discrete-time model.



(a) A complete TS network without any restrictions



(b) Node and arc information

Figure 5.1: Illustration of how a full TS network is structured

5.1.2 Port and Network Representation

The fish farms and feed factories are represented as consumption and production ports, respectively. As in Song and Furman (2013), our model uses a TS network that consists of *(port, time)*-pairs, each represented as a node in the network. The set of nodes is shared between all vessels, but each vessel has its own set of associated arcs in the TS network. These directed arcs represent the potential flow through the network for each vessel, which we elaborate on later.

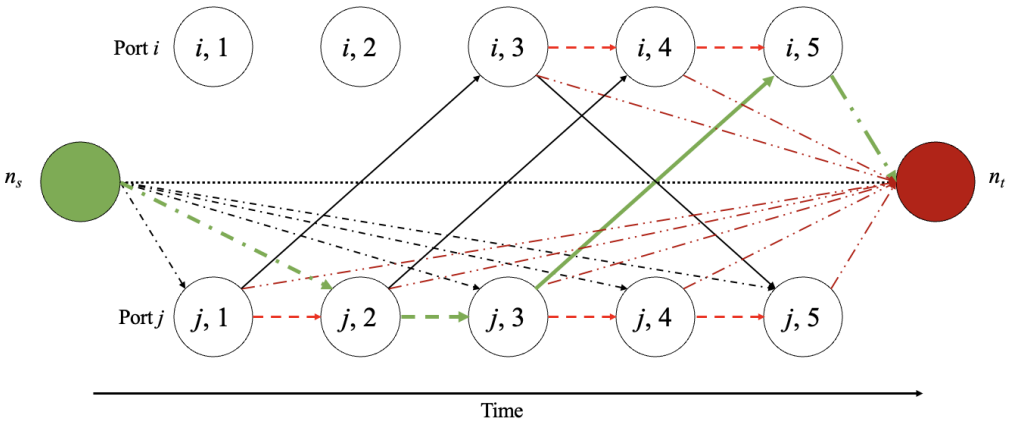
As explained in Chapter 3, every vessel is associated with a port as its origin and a time period in which it becomes available. This is incorporated by ensuring that every vessel has to travel to its origin from an artificial source node in the time period before it becomes available. There are no restrictions on the number of vessels with a particular port as their origin, nor are there any restrictions on how many times a vessel can visit the same port during the planning horizon.

In Figure 5.1 we see a similar TS network as the one used by Song and Furman (2013). The network consists of a source node, n_s, a sink node, n_t, and several normal nodes consisting of a port and a time period. A set of different arc types connects the nodes in the network: (1) travel arcs connect different ports, (2) waiting arcs allow the vessels to remain at a port between time periods, (3) the source and sink nodes have only outgoing and incoming arcs, respectively, and (4) an arc connecting the source and sink directly allows for not using vessels.

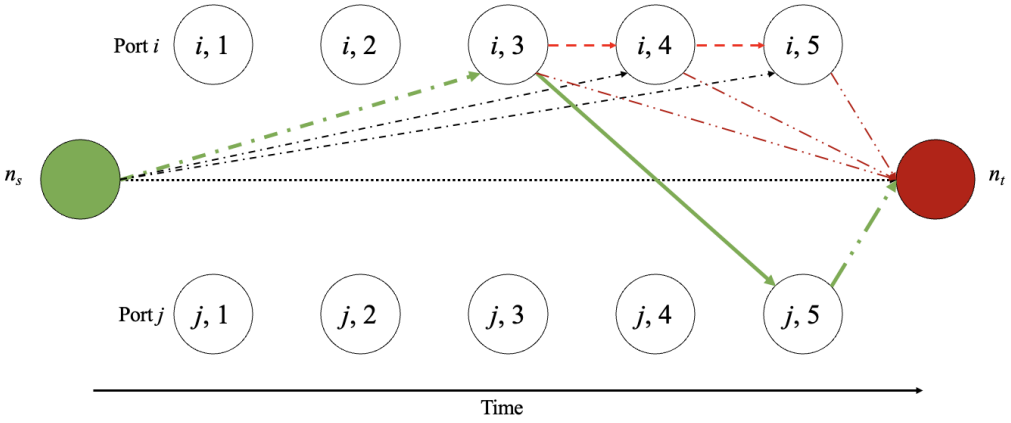
Figure 5.2 illustrates what the underlying network looks like in our case and how adjusting the set of arcs available for a vessel deals with the routing restrictions. The first thing to note is that all vessels are assigned to an origin – the port in which it becomes available. In Figure 5.2a we see the TS network of a vessel that has no travel restrictions; it is available from the first time period, it can visit both ports, but it has to start in port j. Figure 5.2b shows the TS network of a vessel that becomes available in port i in the third time period. Finally, Figure 5.2c displays a case in which a vessel is not compatible with port j, and thus cannot visit it throughout the planning horizon. Note that the travel time between node i and node j is two time periods.

5.1.3 Inventory Management in Vessels

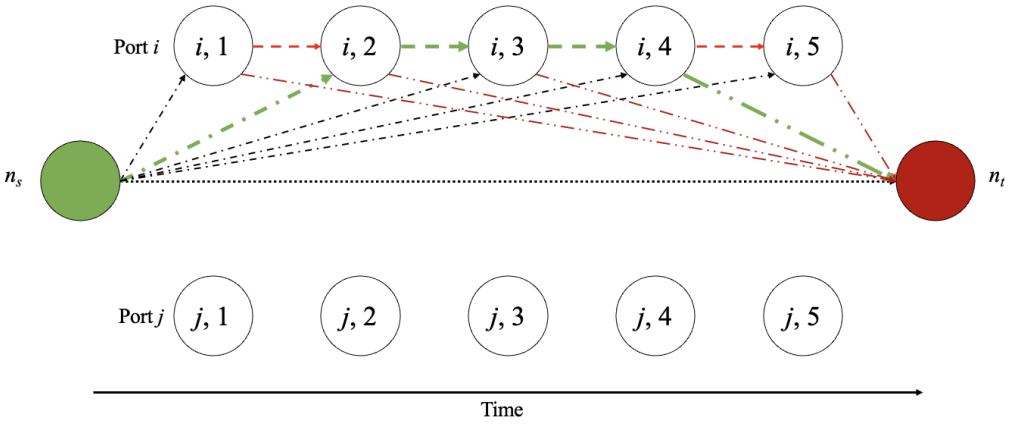
In practice, each vessel will usually be empty when it becomes available in its origin, but it is no formal requirement. Hence, each vessel can have a nonzero amount of feed on-board when it first



(a) Network for a vessel having port j as its origin, without any further restrictions



(b) Network for a vessel becoming available in the third time period with port i as its origin



(c) Network for a vessel that is not compatible with port j

Figure 5.2: Illustration of how the TS network is structured in our thesis. Note that the arcs represent the same as in Figure 5.1b.

becomes available. This flexibility enables the modeling of situations where vessels are executing a previously planned route, but some unforeseen event makes it preferable to reschedule. One such event could be a disease outbreak that makes a vessel's route disadvantageous. The model could then be re-optimized with the farms in which the vessels currently reside as their respective origins and their current load as their initial load.

Since Mowi has two types of vessels, namely bulk and bag vessels, that impose different requirements on inventory management, this has to be handled. Instead of separating the two vessel types in our mathematical model, we adjust our representation such that both vessel types can be handled similarly. For bulk vessels, we have to ensure that the capacity of all silos is respected and that no mixing of multiple products within the same silo occurs. To be able to treat bag vessels similarly, we represent each potential bag as a silo with a capacity equivalent to the bag size. Compared to Mowi's real-life problem, this is a minor simplification as they have multiple bag sizes, which are not accounted for in the model.

5.1.4 Inventory Management at Ports

Even though Mowi still has farms designed for storing bags and not in silos, we have made a simplifying assumption that all farms store their feed in silos. Currently, the number of bag farms is low, and the company plans to move entirely away from them in the future, which justifies the simplification.

In contrast to silos in vessels, we have simplified the modeling of silos in consumption ports. We assume an assigned storage capacity for each feed type for these ports, and the assigned capacity is not allowed to change throughout the planning horizon. In other words, the allocation of feed types to silos is given in advance as input to the model. This differs from vessels with silos, where the model itself decides how to allocate feed types to silos. We argue that this is a fair assumption, as the allocation of silos on farms is closely related to the growth stage of the fish and is thus unlikely to change significantly during the relatively short planning horizon. The allocation on farms is thus more stable than the allocation of the vessels' load, which has to account for multiple farms with potentially vastly different demands.

In the model, we ensure that the production ports do not deliver anything they have not produced. I.e., we incorporate production rates at all ports and ensure that the inventory stays between a minimum and a maximum. However, in Mowi's case, we assume that the production ports can always supply the vessels with the demanded quantity of all products. This simplification is made because the FFMIRP does not incorporate inventory management or production planning on the production side.

Lastly, as Mowi is an vertically integrated company, it is assumed that the costs of storing the feed are independent of the specific location in which it is stored. Thus, we assume that the storage costs at the factory and all fish farms are the same, and consequently, they are left out of the model.

5.1.5 Routing Assumptions

Upon arrival at a node in the network, the vessel must either wait or start the unloading or loading process in the subsequent time period. However, as a good solution most likely tries to avoid waiting, we impose an extra cost if the vessel waits before starting to unload. When a vessel becomes available in its origin, it is given one additional option to initiate a sailing activity immediately, without loading or unloading any quantity at the origin port. This is necessary when the vessel has a nonzero initial feed load or has its origin in a consumption port. Additionally, as mentioned in Section 5.1.2, there is no restriction on the number of times a vessel can visit a particular port. This allows a vessel to visit a production or a consumption port multiple times to either load or deliver more feed during the planning horizon.

5.1.6 Dealing with End-Effects

A challenge when solving MIRPs is to avoid inventory levels at the end of the planning horizon that are unfortunate for planning beyond the planning horizon. When only minimizing the distribution cost, the model has no incentive to travel to a port unless the port will experience an overflow or stockout by the end of the planning horizon. As a result, the ports risk having a meager inventory (or a lot in the case of production ports) in the final time period. Such solutions seem attractive to the model as it only considers the costs within the planning horizon. However, it can make planning beyond the planning horizon difficult. To reduce such negative end-effects a revenue is given per quantity delivered or picked up at the ports. Setting the revenue sufficiently high will incentivize the model to make deliveries even when it is not strictly needed. Since a new planning horizon always follows another, this might help ease the planning and reduce costs over several planning horizons.

5.2 Notation

The following section presents the notation used in our mathematical model. First, we introduce all sets in Section 5.2.1, then we explain the parameters and variables in Section 5.2.2 and Section 5.2.3, respectively.

5.2.1 Sets

$v \in \mathcal{V}$	–	Set of vessels
$i \in \mathcal{I}^C$	–	Set of consumption ports
$i \in \mathcal{I}^P$	–	Set of production ports
$i \in \mathcal{I}$	–	Set of all ports, $\mathcal{I} = \mathcal{I}^C \cup \mathcal{I}^P$
$t \in \mathcal{T}$	–	Set of time periods in the planning horizon
$p \in \mathcal{P}$	–	Set of different fish feed products
$c \in \mathcal{S}_v$	–	Set of silos on vessel $v \in \mathcal{V}$
$n \in \mathcal{N}$	–	Set of port-time pairs that constitute the nodes. $\mathcal{N} = \{(i,t) : i \in \mathcal{I}, t \in \mathcal{T}\}$
$n \in \mathcal{N}_{st}$	–	Set of nodes including the source and the sink. $\mathcal{N}_{st} = \mathcal{N} \cup n_s \cup n_t$
$a \in \mathcal{A}_v$	–	Set of all arcs associated with vessel $v \in \mathcal{V}$
$a \in \mathcal{FS}_{nv}$	–	Set of all outgoing arcs associated with node $n \in \mathcal{N}_{st}$ and vessel $v \in \mathcal{V}$
$a \in \mathcal{RS}_{nv}$	–	Set of all incoming arcs associated with node $n \in \mathcal{N}_{st}$ and vessel $v \in \mathcal{V}$

5.2.2 Parameters

Vessel-specific parameters

Q_{vc}	–	The capacity of silo $c \in \mathcal{S}_v$ on vessel $v \in \mathcal{V}$
S^0_{vcp}	–	The initial inventory of product $p \in \mathcal{P}$ in silo $c \in \mathcal{S}_v$ on vessel $v \in \mathcal{V}$
C^T_{va}	–	Cost to traverse arc $a = ((i_1,t_1),(i_2,t_2)) \in \mathcal{A}_v$ with vessel $v \in \mathcal{V}$

Port-specific parameters

B_{it} – Berth capacity (maximum number of vessels loading or unloading) at port $i \in \mathcal{I}$ at time $t \in \mathcal{T}$

$\overline{S}_{ip}$ – Total capacity of product $p \in \mathcal{P}$ at port $i \in \mathcal{I}$

$\underline{S}_{ip}$ – The lower inventory limit of product $p \in \mathcal{P}$ at port $i \in \mathcal{I}$

S^0_{ip} – The initial inventory of product $p \in \mathcal{P}$ at port $i \in \mathcal{I}$

D_{itp} – The consumption/production of product $p \in \mathcal{P}$ at port $i \in \mathcal{I}$ in time $t \in \mathcal{T}$

I_i – Indicator of the port type. 1 if the node is a production port, and -1 if the node is a consumption port

ϵ – Nonnegative and small cost parameter for attempting to load or unload at a port

β_{it} – The maximum amount of all products that can be bought from the spot market by port $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$

β^{TOT}_i – The maximum amount of all products that can be bought from the spot market by port $i \in \mathcal{I}$ throughout the planning horizon

F^{MIN}_{it} – The minimum amount that can be loaded/unloaded on/from a single vessel in port $i \in \mathcal{I}$ in time $t \in \mathcal{T}$ if such action takes place

F^{MAX}_{it} – The maximum amount that can be loaded/unloaded on/from a single vessel in port $i \in \mathcal{I}$ in time $t \in \mathcal{T}$

C^S_{it} – The unit cost of buying from the spot market for port $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$

R_{it} – The revenue generated per unit delivery to port $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$

5.2.3 Variables

x_{va} – 1 if vessel $v \in \mathcal{V}$ traverses arc $a \in \mathcal{A}_v$, 0 otherwise

z_{nv} – 1 if vessel $v \in \mathcal{V}$ can load/unload at node $n \in \mathcal{N}$, 0 otherwisse

q_{nvcp} – The quantity loaded/unloaded from/to silo $c \in \mathcal{S}_v$ from vessel $v \in \mathcal{V}$ of product $p \in \mathcal{P}$ at node $n \in \mathcal{N}$

y_{vctp} – 1 if silo $c \in \mathcal{S}_v$ in vessel $v \in \mathcal{V}$ contains product $p \in \mathcal{P}$ at time period $t \in \mathcal{T}$

α_{itp} – Amount of product $p \in \mathcal{P}$ bought/sold from/to the spot market by port $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$

s^P_{itp} – The current stock of product $p \in \mathcal{P}$ at port $i \in \mathcal{I}$ at the *end* of time period $t \in \mathcal{T}$

s^V_{vctp} – The current stock of product $p \in \mathcal{P}$ in silo $c \in \mathcal{S}_v$ in vessel $v \in \mathcal{V}$ at the end of time period $t \in \mathcal{T}$

5.3 Model Formulation

In this section, we present the mathematical model.

5.3.1 Objective Function

The objective consists of four terms: the revenue generated from delivering feed to the nodes, the cost associated with traversing the arcs between ports, the cost that occurs when the spot market is used, and finally, the cost associated with docking at a port. The objective is given in Equation (5.3.1) and maximizes the profit generated. Note that the objective is minimized. This is because

the purpose of the revenue is to reduce adverse end-effects, while the main focus is on minimizing distribution costs.

$$\min z = \sum_{v\in\mathcal{V}}\sum_{a\in\mathcal{A}_v} C^T_{va}x_{va} + \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}} C^S_{it}\alpha_{itp} + \sum_{i\in\mathcal{I}}\sum_{v\in\mathcal{V}}\sum_{t\in\mathcal{T}}(t\epsilon)z_{(i,t)v} - \sum_{n\in\mathcal{N}}\sum_{v\in\mathcal{V}}\sum_{c\in\mathcal{S}_v}\sum_{p\in\mathcal{P}} R_n q_{nvcp} \tag{5.3.1}$$

The first term constitutes the total travel cost for all vessels. The second term states that there are costs related to buying products from the spot market for every combination of a port and a time period. The third term deals with the cost of docking at a port. As in Papageorgiou et al. (2014c), we have multiplied by t in the objective. This is done to urge performing the deliveries as early as possible, which has proven beneficial. The absolute value of ϵ is very low, as it should not affect the primary goal of minimizing the costs, contributing to telling the model that a solution in which a docking occurs earlier than in another is preferred. The final term makes up the received revenue generated from the products being delivered and picked up.

5.3.2 Routing Constraints

As we represent the problem using a TS model, we must ensure that the flow through the network is balanced. This means that the number of incoming and outgoing arcs in the network should be equal for all regular nodes in the network. The source node should have one arc leaving for every vessel and none entering. Similarly, the sink node should have one arc entering and none leaving. This is represented by Constrains (5.3.2).

$$\sum_{a\in\mathcal{FS}_{vn}} x_{va} - \sum_{a\in\mathcal{RS}_{vn}} x_{va} = \begin{cases} 1 & \text{, if } n = n_s \\ -1 & \text{, if } n = n_t \\ 0 & \text{, otherwise} \end{cases}, \quad n \in \mathcal{N}_{st}, \quad v \in \mathcal{V} \tag{5.3.2}$$

Note that as long as Constraints (5.3.2) are respected, most other aspects of the routing are handled by pre-processing the set of arcs. A vessel will not be able to start before it becomes available, as there are no arcs to travel along in these time periods. It is also forced to start in its origin as this is the only arc available when it becomes available. Particular ports that cannot be visited by certain vessels are handled by excluding all arcs to these ports. Furthermore, waiting in a port is possible as the network includes arcs going from a port back to the same port with a travel distance of one time period. Finally, the travel time aspect is handled by the time it takes to traverse an arc; for every vessel, the difference between the start and end point of a travel arc represents the travel time. This also allows the travel time between two ports to depend on the time period in which the travel starts.

As we do not have any hard requirements on the vessels being empty upon returning to a production facility and have included the discount for traveling empty, another level of complexity is added. Papageorgiou et al. (2014c) assume that all vessels are empty when they return to a production facility and when empty, travel costs are reduced by a discount factor. However, we argue that a similar assumption can be made in our case; in Mowi's problem, we ignore the production scheduling part as discussed earlier. Thus, the inventory levels at the production sites are ignored. This means we can assume high enough inventory at the factories when a vessel arrives to fill it. Consequently, the only reason for traveling back to a factory with products in stock would be to save loading time by loading more than needed for the next voyage at a production facility with high loading rate than the next planned production facility visit. However, in Mowi's problem, there is currently just one factory, and thus, we assume that vessels always return empty to a production facility. Hence, the costs of traversing all arcs entering a production node are discounted as fuel consumption is lower when traveling empty. In other words, it is cheaper to travel to a production node, than to leave a production node.

5.3.3 Inventory Constraints

This section presents the constraints related to inventory control for ports and vessels.

Port Storage

At the end of a time period, the inventory at a port is the sum of the inventory at the beginning of the period, the consumed/produced amount, the loaded/unloaded amount, and the amount bought from/sold to the spot market. This is represented by Constraints (5.3.3).

$$s^P_{itp} = s^P_{i,t-1,p} + I_i(D_{itp} - \sum_{v\in\mathcal{V}}\sum_{c\in\mathcal{S}_v} q_{vctp} - \alpha_{itp}), \quad n = \{(i,t) : i \in \mathcal{I}, t \in \mathcal{T}\backslash\{0\}\}, p \in \mathcal{P} \tag{5.3.3}$$

The initial inventory of all products in all ports is handled by Constraints (5.3.4).

$$s^P_{i0p} = S^0_{ip} + I_i(D_{i0p} - \sum_{v\in\mathcal{V}}\sum_{c\in\mathcal{S}_v} q_{vc0p} - \alpha_{i0p}), \quad i \in \mathcal{I}, p \in \mathcal{P} \tag{5.3.4}$$

At all times, the inventories of all fish feed products have to stay between the provided minimum and maximum bounds. This is ensured by Constraints (5.3.5).

$$\underline{S}_{ip} \leq s^P_{itp} \leq \overline{S}_{ip}, \quad i \in \mathcal{I}, t \in \mathcal{T}, p \in \mathcal{P} \tag{5.3.5}$$

Vessel Storage

The outgoing inventory level of a product in a particular silo in a vessel in a time period is the sum of the incoming inventory and the amount loaded or unloaded throughout the time period from the compartment. This is enforced by Constraints (5.3.6).

$$s^V_{vctp} = s^V_{vc,t-1,p} + \sum_{i\in\mathcal{I}} I_i q_{vc(i,t)p}, \quad v \in \mathcal{V}, c \in \mathcal{S}_v, t \in \mathcal{T}\backslash\{0\}, p \in \mathcal{P} \tag{5.3.6}$$

We also have to handle the initial inventory of all products in all compartments in all vessels; this is done by Constraints (5.3.7).

$$s^V_{vc0p} = S^0_{vcp} + \sum_{i\in\mathcal{I}} I_i q_{vc(i,0)p}, \quad v \in \mathcal{V}, c \in \mathcal{S}_v, p \in \mathcal{P} \tag{5.3.7}$$

Further, as we do not require the vessels to be empty when arriving at the production ports, it must be made sure that the inventory levels in all silos are below the capacities of the silos. This is enforced by Constraints (5.3.8).

$$\sum_{p\in\mathcal{P}} s^V_{vctp} \leq Q_{vc}, \quad v \in \mathcal{V}, c \in \mathcal{S}_v, t \in \mathcal{T} \tag{5.3.8}$$

Constraining the storage is more complicated for the vessels due to the non-allocated compartments. The product stored in a silo can change throughout the planning horizon multiple times. At the same time, it is crucial to ensure that multiple products are never mixed in the same compartment. Constraints (5.3.9) make sure that there is at most one product located in each compartment in all vessels throughout all time periods. Further, Constraints (5.3.10) ensure that the product stored in a compartment cannot change unless it is emptied.

$$\sum_{p \in \mathcal{P}} y_{vctp} \leq 1, \quad v \in \mathcal{V}, c \in \mathcal{S}_v, t \in \mathcal{T} \tag{5.3.9}$$

$$s^V_{vctp} \leq Q_{vc} y_{vctp}, \quad v \in \mathcal{V}, c \in \mathcal{S}_v, t \in \mathcal{T}, p \in \mathcal{P} \tag{5.3.10}$$

Note that the Q_{vc} values work as "big-M"-bounds in Constraints (5.3.10).

Loading and Unloading

There is a limited number of berths available for docking at the ports; it must therefore be enforced that the number of vessels loading or unloading does not surpass this number. This is done by Constraints (5.3.11).

$$\sum_{v \in \mathcal{V}} z_{nv} \leq B_{it}, \quad i \in \mathcal{I}, t \in \mathcal{T} \tag{5.3.11}$$

Further, a vessel can only deliver products to a port if the vessel is located at that port. This is ensured by Constraints (5.3.12).

$$z_{nv} \leq \sum_{a \in \mathcal{RS}_{nv}} x_{va}, \quad n \in \mathcal{N}, v \in \mathcal{V} \tag{5.3.12}$$

All *(vessel, port)*-pairs have lower and upper bounds on the loading and unloading rates within a time period, meaning that the total flow of products must be within these bounds. Constraints (5.3.13) enforce this.

$$F^{\text{MIN}}_{it} z_{nv} \leq \sum_{c \in \mathcal{S}_v} \sum_{p \in \mathcal{P}} q_{vcnp} \leq F^{\text{MAX}}_{it} z_{nv}, \quad n \in \mathcal{N}, v \in \mathcal{V} \tag{5.3.13}$$

Further, we have a restriction on the amount that can be bought from the spot marked in each time period. Constraints (5.3.14) make sure that this is enforced.

$$\sum_{p \in \mathcal{P}} \alpha_{itp} \leq \beta_{it}, \quad i \in \mathcal{I}, t \in \mathcal{T} \tag{5.3.14}$$

Finally, the total amount of a product bought or sold at all ports must be within the total limit for the planning horizon. This is ensured by Constraints (5.3.15).

$$\sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}} \alpha_{itp} \leq \beta^{\text{TOT}}_i, \quad i \in \mathcal{I} \tag{5.3.15}$$

5.3.4 Red, Yellow and Green Zones

As mentioned earlier, disease outbreaks also have to be handled. We use three categories of fish farms – *red, yellow,* and *green* farms. For descriptions of what the labels mean, consult Section 2.3.1. Vessels leaving red and yellow farms have to wait for a specified *recovery time* before they can dock at a green farm, and similarly, vessels leaving a red farm have to wait before they can dock at a yellow farm. These constraints are handled by pre-processing the data. All arcs leading to a recovery time have an associated travel time equal to the maximum of its original travel time and the recovery time. As this potentially increased time is accounted for in the cost, we avoid all prohibited travels without further constraints.

5.3.5 Binary and Non-Negativity Constraints

$$x_{av} \in \{0,1\}, \quad a \in \mathcal{A}_v, v \in \mathcal{V}, \tag{5.3.16}$$

$$z_{nv} \in \{0,1\}, \quad n = (i,t) \in \mathcal{N}, v \in \mathcal{V} \tag{5.3.17}$$

$$y_{vcpt} \in \{0,1\}, \quad v \in \mathcal{V}, c \in \mathcal{S}_v, p \in \mathcal{P}, t \in \mathcal{T} \tag{5.3.18}$$

$$q_{vcpn} \geq 0, \quad v \in \mathcal{V}, c \in \mathcal{S}_v, p \in \mathcal{P}, t \in \mathcal{T} \tag{5.3.19}$$

$$\alpha_{pit} \geq 0, \quad p \in \mathcal{P}, i \in \mathcal{I}, t \in \mathcal{T} \tag{5.3.20}$$

$$s^P_{pit} \geq 0, \quad p \in \mathcal{P}, i \in \mathcal{I}, t \in \mathcal{T} \tag{5.3.21}$$

$$s^V_{pvct} \geq 0, \quad p \in \mathcal{P}, v \in \mathcal{V}, c \in \mathcal{S}_v, t \in \mathcal{T} \tag{5.3.22}$$

Chapter 6

A Scalable Memetic Optimization Algorithm with LP-based Search Techniques

This chapter introduces our algorithm, termed the Scalable Memetic Optimization algorithm with LP-based search Techniques (hereafter SMOLT), an algorithm for solving the FFMIRP. It has a novel solution representation and several novel mutation operators.

An overview of SMOLT is given in Figure 6.1. As can be seen, SMOLT consists of two main components – a heuristic for deciding the routes of all vessels and an exact solution method for assigning the loading and unloading quantities given a routing solution. The core idea of SMOLT is to keep the reliance on mathematical programs to a minimum – the LP solved is simple and *all* routing decisions are made by the heuristic. The rationale behind this is to ensure the scalability of SMOLT; a well-developed heuristic can scale well for complex problems, but an exact method becomes intractable when the number of variables and constraints gets too large.

Section 6.1 gives an overview of the metaheuristic SMOLT is based on, the memetic algorithm (MA), and introduces the concept of islanding used to increase performance and population diversity. Then, Section 6.2 presents the chosen solution representation in SMOLT, and Section 6.3 describes its fitness function. Further, Section 6.4 gives a thorough presentation of the different operators used, emphasizing the mutation operators used for conducting local searches. Finally, Section 6.5 presents how islanding is implemented in SMOLT specifically, how compartment allocation is handled, and how we gradually expand the planning horizon to improve the scalability of SMOLT.

6.1 Memetic Algorithms with Islanding

MAs are a subgroup of a more broad class of metaheuristics, namely the genetic algorithm (GA). GAs are inspired by the principle of evolution applicable to solving a wide variety of optimization problems. Though different variations exist, the core principle is straightforward. GAs work by maintaining a population of candidate solutions that are evolved throughout *generations* by perturbing single individuals and by constructing new individuals by combining multiple candidate solutions. This is analogous to the principles of genetic mutation and sexual reproduction found in nature. Candidate solutions within a population are commonly known as *individuals*, which is a convention we will apply here.

To guide evolution, each individual has an associated *fitness*. This is a scalar- or vector value defining the quality of the solution represented by the individual. An ordering between fitnesses allows us to say whether one solution is better than another. Evolutionary pressure is usually

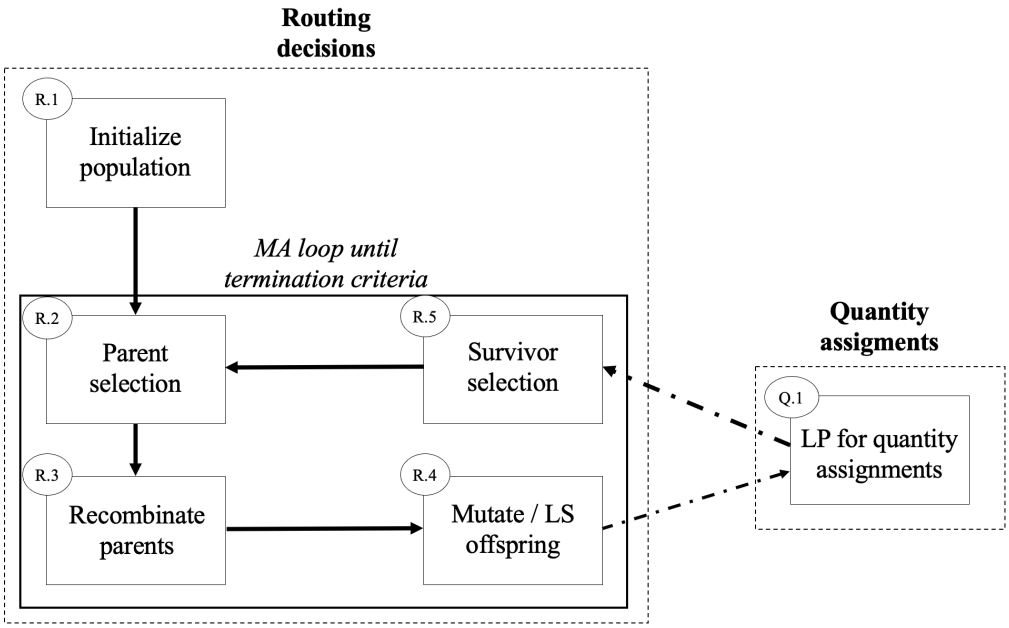


Figure 6.1: An overview of the components of SMOLT. First the population is initialized as empty solutions. After the population has been initialized, the memetic algorithm loop starts. The loop consists of four steps: (1) parents are selected for generating offspring based on their fitness, (2) the parents are recombined to create offsprings, (3) the offsprings are mutated through simple mutations and *local search* techniques, and (4) the new population is selected from the combination of parents and offspring. Every time a routing solution changes, and the delivered quantities are requested, the LP for quantity assignments is invoked.

applied in at least two ways: (1) by increasing the likelihood that fit individuals are selected for reproduction and (2) by making it more likely for fit individuals to survive from one generation to the next. This is akin to how different species evolve. Individuals who are slightly better adapted to their surroundings, i.e., have better fitness, are more likely to survive and produce offsprings that carry their genes into the next generation. These improvements accumulate over time, such that a population of individuals slowly evolve into a form that is well-suited for the environment in which they live. Similarly, as generations pass, the fitness of the population within a GA tends to improve, which is equivalent to saying that the quality of the solution candidates increases. A graphical illustration is provided in Figure 6.2.

SMOLT uses a concept known as *islanding* to improve the performance of the GA (Whitley et al., 1999). Rather than maintaining just one population of individuals that are evolved together, we instead maintain multiple "islands." Each island is a separate population that is evolved independently from all the others. At regular intervals, individuals are exchanged between different islands in a process known as *migration*.

The purpose of islanding is twofold. The first is to preserve diversity. The individuals within a GA's population will tend to lose diversity over time. That is to say that many of the individuals within the population will be very similar. This can lead to the GA being stuck in a local optimum. Islanding works to preserve diversity, as the independent evolution of each island might lead each of them towards different local optima. The migrations work as a mechanism to share knowledge (i.e., good solutions), and provide islands with a possibly different set of individuals (i.e., increase diversity). The second purpose of islanding is to enable parallel computation for faster convergence. Each island is only loosely coupled through the migration mechanism. This makes it possible to run each island in parallel with only minimal overhead. Consequently, we can achieve a near-linear increase in throughput as a function of available processor cores.

MAs are GAs augmented with local search techniques. SMOLT uses an islanding GA combined with an ensemble of local search operators, primarily based on the ruin & recreate (R&R) paradigm. R&R is also commonly known under the name "destroy and repair". The usage of such local search

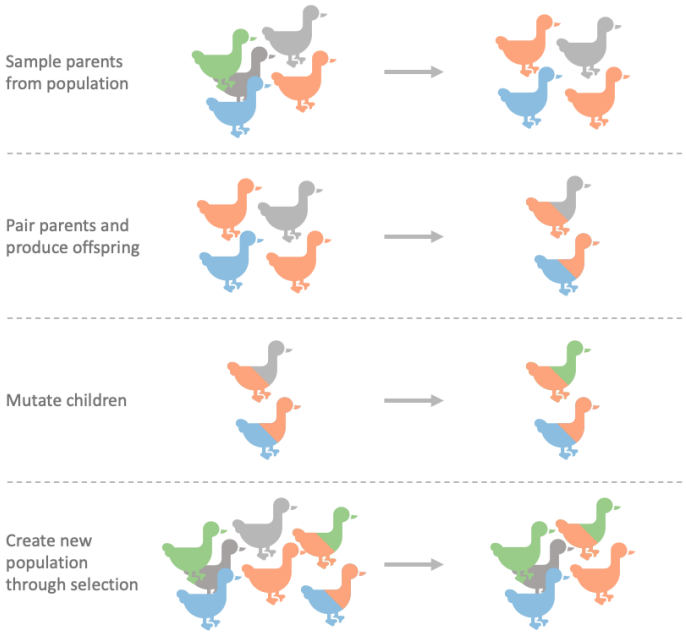


Figure 6.2: An illustration of the high-level steps in GAs and MAs.

operators causes SMOLT to be categorized as an MA. Hereafter, we will only refer to MAs when discussing concepts related to the two classes of algorithms.

6.2 Solution Representation

Central to every MA is the solution representation. It defines the boundaries which the MA's operators will have to operate within. In a MIRP, there are two distinct kinds of decisions – routing and quantities. Routing decisions concern what locations each vessel should visit, in what order, and at what time. The quantity decisions concern how much a vessel should pick up or deliver at each location it will visit. In SMOLT, the MA handles routing decisions while the optimal quantities for a given routing solution are determined by solving an LP. This means that the quantity decisions are always optimal and entirely determined by the routing decisions. Hence, the MA can be limited to only working on the routing decisions without impacting SMOLT's possibility of finding optimal solutions. A full candidate solution is the routing decisions combined with the corresponding optimal quantity assignment. This section first presents the routing representation before providing a brief overview of the LP determining the quantities.

6.2.1 Routing

Each candidate solution in the MA consists of a routing plan for each vessel. Each vessel's routing plan is a list of $(port, time)$-pairs. We refer to each such $(port, time)$-pair as a *visit* and it is equivalent to a node in the TS network presented in Chapter 5. Each vessel's solution tells which nodes it should visit and when it should do so. An illustration of the routing representation is provided in Figure 6.3.

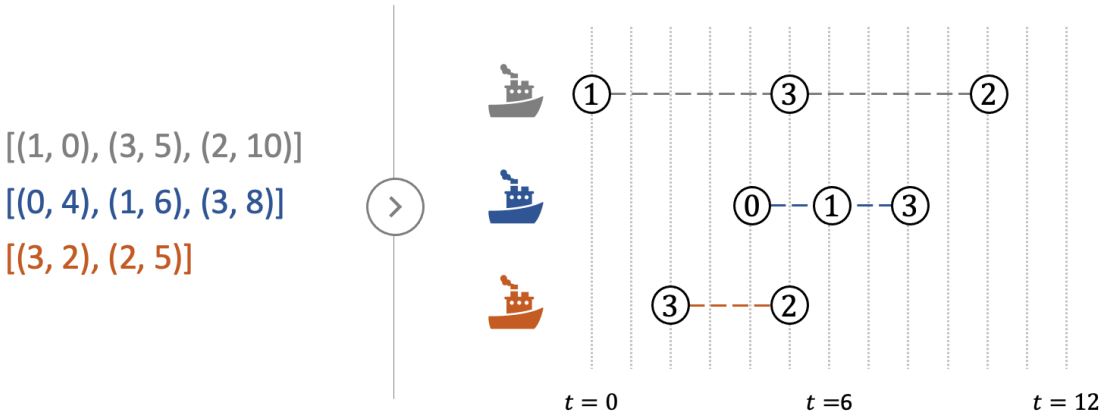


Figure 6.3: Solution representation. The left half shows a single routing solution in its list-based representation, while the right side shows the corresponding routing plan of the vessels. Time is shown by the vertical bars, while each numbered circle corresponds to a location. Do note that the solution representation does not consider travel time in any way.

The chosen representation is flexible. In particular, it allows the distance in time between two visits to be less than the time required to travel between the corresponding ports. We refer to this concept as *time warp*. This is allowed in the representation but is discouraged by (1) penalizing solutions that have time warp and (2) targeted mutations. However, despite the flexible representation, we do enforce the following three constraints:

1. The first visit in each vessel's plan must correspond to a visit at its origin at the time it becomes available
2. No visit can happen after the end of the planning horizon
3. There can be no more than one visit at a given time in any vessel's plan. That is, a vessel v can not have two different visits that happen simultaneously

Mutations and the recombination are designed such that these three constraints are satisfied at all times.

6.2.2 LP Model for Quantity Assignments

As mentioned, the MA is responsible for all routing decisions. However, the algorithm makes no quantity decisions; these are done by an LP model embedded in the MA. This LP takes a routing solution from the MA as input and assigns all loading and unloading quantities. In this section, we shortly present the main idea of the LP, which can be read in its entirety in Appendix A. Table 6.1 summarizes the notation used for presenting the overview of the LP.

Table 6.1: Notation used for the LP.

Sets	
$i \in \mathcal{I}$	Set of all ports
$v \in \mathcal{V}$	Set of vessels
$t \in \mathcal{T}$	Set of time periods in the planning horizon
$p \in \mathcal{P}$	Set of different fish feed products
Parameters	
C^S_{it}	The unit cost of buying from the spot market for port $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$
R_{it}	The revenue generated per unit delivered to port $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$
τ	Weight used for adjusting the punishment for inventory violations in ports
ε	Weight used for adjusting how much the algorithm prefers early deliveries
Variables	
x_{ivtp}	The quantity loaded/unloaded from/to port $i \in \mathcal{I}$ from vessel $v \in \mathcal{V}$ of product $p \in \mathcal{P}$ at time $t \in \mathcal{T}$
w_{itp}	Inventory violation at production port $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$ of product $p \in \mathcal{P}$
α_{itp}	Amount bought from/sold to the spot market by port $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$ of product $p \in \mathcal{P}$

The main idea behind the LP is based on the following; if all routes are considered fixed, the available load on the vessels should be allocated such that inventory violations and spot market transactions are kept at a minimum, while striving to deliver as much as possible. This is represented by equation (6.2.1).

$$\max z = \sum_{i\in\mathcal{I}}\sum_{v\in\mathcal{V}}\sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}}(R_{it} - \varepsilon t)x_{ivtp} - \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}}(C^S_{it}\alpha_{itp} + \tau w_{itp}) \tag{6.2.1}$$

The term $R_{it}x_{ivtp}$ expresses the revenue associated with all loading and unloading actions. Further, the term $\varepsilon t x_{ivtp}$ makes early deliveries preferable to later ones. The term $C^S_{it}\alpha_{itp}$ constitutes the costs for buying products from the spot market or the lost revenue that incurs if the product has to be sold from the production ports due to overflow. Finally, the term τw_{itp} expresses the inconvenience caused by inventory violations in either consumption or production ports. Note that the second and fourth terms include weights; these are intended to let the user adjust the importance of the different parts. Similar weights for revenue and cost can be attained by directly modifying the revenue parameters R_{it} and cost parameters C^S_{it}, and are thus not included. A specific routing solution from the MA gives which vessels that can load or unload at which ports in which time periods. When a vessel is not present at a port, the associated x variable is given a upper bound of zero.

The objective represented by Equation (6.2.1) is only subject to two groups of hard constraints. First, the upper inventory limit for consumption ports and the lower inventory limit for production ports must be respected. Secondly, the vessels' inventory limits are respected at all times. The other inventory constraints at ports are soft, and are accounted for by decreasing the objective through the violation variable, w.

We mainly prioritize reducing violations and dependency on the spot market in our implementation. As Mowi is a vertically integrated company, the revenue associated with delivering products to different farms is assumed to be more or less equivalent. Thus, the revenue term is mainly used for avoiding adverse end effects, as mentioned in Section 5.1.6. To recall, the model will not deliver beyond the required quantity to avoid violations if it has no incentive to do so. This is an inefficient long-term strategy, and the revenue term is intended to prevent this from happening. Further, the term making early actions preferable is intended to enable differentiation between similar decisions that only differs in time. In these scenarios, it is desired that the product is delivered as early as possible.

6.3 Fitness

SMOLT uses a fitness function to evaluate the quality of a solution. It tries to minimize the fitness value, and it is used to guide both parent selection and survival selection. As such, it needs to capture the qualities that we want in a solution. Our goal is to find a solution that does not violate inventory constraints while minimizing the net cost. For this reason, our fitness function ought to include these as terms. In addition, we need to penalize solutions that are allowed in our representation but that violate the problem constraints.

As explained in Section 6.2, our solution representation allows time warps. This is not allowed per the problem specification and will need to be discouraged. We add this time warp as a penalty term in the fitness function to combat this. Similarly, we also need to penalize breaches of berth capacity. The quantity assignment LP does not consider breaches of port capacity since doing so would require binary variables, which would turn it into a MIP. This is not a viable alternative since the LP for quantity assignment needs to be solved frequently. Turning it into a MIP would significantly impact the time required to solve each program. Instead, we count the number of port capacity violations in the quantity assignment given by the LP and penalize these in the fitness function.

These factors make up our fitness function. The selection pressure in the population refers to the likeliness of fitter individuals to survive or be chosen as parents. Too high selection pressure can result in premature convergence. One way to avoid overly high selection pressure is to reduce the individual's relative difference in fitness. We, therefore, apply a logarithm to a weighted sum of the mentioned factors. This also requires us to add an offset to prevent the weighted sum from becoming less than 1. Equation (6.3.1) shows the general form of the fitness function used. SMOLT seeks to minimize this value.

$$f(s) = \ln\left(\alpha \cdot \text{warp}(s) + \beta \cdot \text{berth}(s) + \gamma \cdot \text{violation}(s) + \delta \cdot (\text{cost}(s) - \text{revenue}(s)) + \text{offset}\right) \tag{6.3.1}$$

The fitness function as implemented in SMOLT uses a time-dependent set of weights, α, β, γ, δ, for the different terms of the fitness function. The idea is to prioritize cost-efficient routes in the early iterations and then ensure there are no inventory violations, berth breaches, or time warps. Thus, the weights for time warps, berth capacity, and inventory violations are set to 0 at initialization. The weights are then linearly increased, and after a while, the weights are significantly higher than that of the revenue and costs, prioritizing finding feasible solutions. An individual's fitness must be positive to work correctly with the selection procedure, and an offset large enough to avoid the sum from becoming less than 1 is therefore added in the last term.

6.4 Memetic Algorithm Components

This section presents the MA in SMOLT in a structured and detailed manner. Algorithm 1 gives an overview of the MA. First, we initialize a population of empty solutions, i.e., with no visits except the origin visit of each vessel. Then, we present how parents are selected to generate offspring in and how the next generation is selected from the pool of offspring and parents in Section 6.4.1. Further, the crossover technique used, PIX, is presented in Section 6.4.2. Finally, the mutation and local search operators are described in detail in Section 6.4.3.

6.4.1 Parent and Survivor Selection

The MA underpinning SMOLT uses a k-tournament for selecting parents. A parent is chosen by choosing the best among k randomly chosen individuals from the population. This is repeated multiple times to construct a set of parents, which will be used as the basis for recombination (i.e., reproduction) and mutation. Survivor selection uses roulette wheel selection with k-elite,

Algorithm 1 SMOLT's genetic algorithm

```
PARAMETER(k_t: Number of individuals selected in each tournament)
PARAMETER(k_e: Number of elites)
PARAMETER(p: Number of parents to generate)
PARAMETER(n: Population size)
                                                    ▷ High-level flow of SMOLT
function smolt-ga():
    /* Start with a population of empty individuals                          */
    population ← initialize-empty(n)
    while TERMINATION CRITERIA NOT MET do
        /* Select p parents by holding p k-tournaments using k_t parents in each
           tournament                                                        */
        parents ← select-parents(population, p, k_t)
        /* Generate offspring by using PIX crossover on parents              */
        offspring ← recombine(parents)
        /* Mutate offspring using a set of mutation and local search operators */
        offspring ← mutate(offspring)
        /* Select survivors from parents and offspring keeping k_e best      */
        population ← select-survivors(population, offspring, k_e)
    return population
```

first introduced by De Jong (1975). Given both the old population and the generated children, it starts by first selecting the k best. These constitute the elite and are guaranteed to make it to the next generation. The rest of the survivors are chosen by sampling the desired number of individuals from the two populations. The probability that an individual is chosen is proportional to its weight, which is a function of its fitness. Fit individuals will be given more significant weight and will thus have a higher probability of being brought into the new generation.

6.4.2 Recombination

To combine gene information between the individuals in the population, we use a simple crossover technique to generate offsprings from parents. The recombination works by combining the routes from two parents – either by directly copying the route of one or several vessels or by crossing the routes for the same vessel from the parents. In all cases, we will produce exactly two offsprings for each pair of parents.

First, two parents are selected. Then, the set of vessel indices is divided into three sets – (1) a set used for copying vessel routes directly from parent p to offspring i, (2) a set used for copying vessel routes directly from parent p to offspring j, and (3) a set used to mix the routes of the two parents. The third set is mixed using a 1-point crossover. In a 1-point crossover, a random index, n, is generated, and then all visits with an index lower than n from parent p, and all visits with an index equal to or larger than n from parent q, are selected to constitute a new route. A visual representation of the crossover is given in Figure 6.4.

The figure shows that two parents have been selected, and the vessel indices are divided into three sets. Vessels 1 and 3 constitute set 1, vessel 5 constitutes set 2, and vessels 2 and 4 constitute set 3. In "Step 1," the routes belonging to set 1 are directly copying from parent 1 to offspring 1 and from parent 2 to offspring 2. In "Step 2," the routes from set 2 are copied from parent 1 to offspring 2 and from parent 2 to offspring 1. Finally, a 1-point crossover is done on the routes in set 3 with $n = 1$ for both vessels.

6.4.3 Mutation and Local Search Operators

The most influential components of SMOLT are the solution representation combined with the recombination operator and the various mutation operators. The operators are designed to work

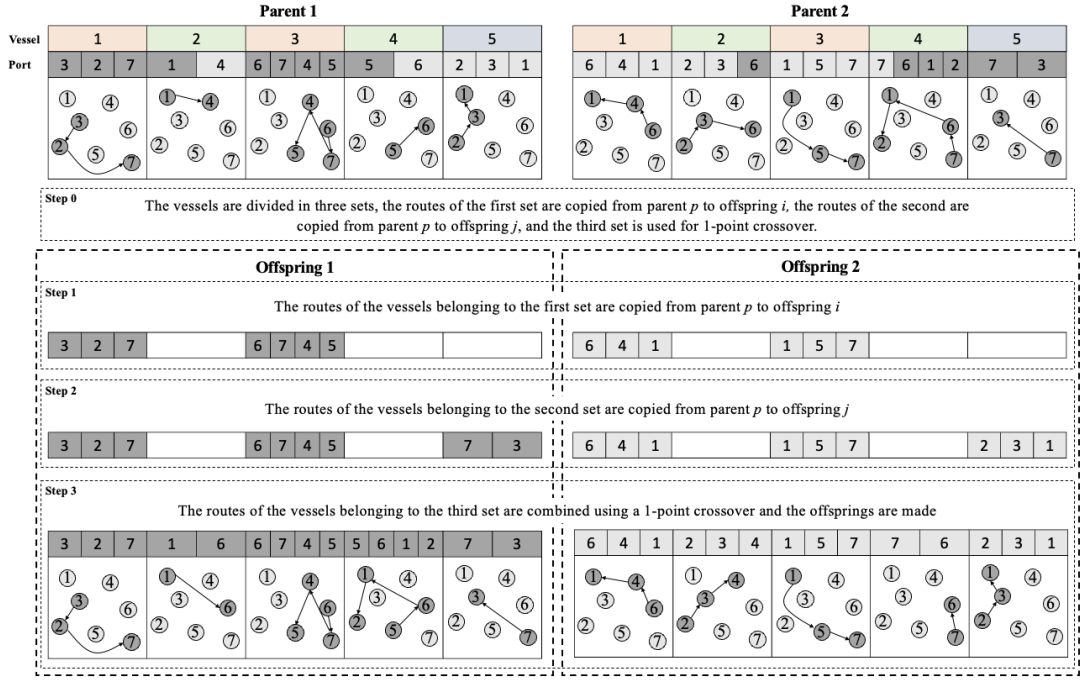


Figure 6.4: PIX crossover.

in tandem with the routing representation to allow the MA to converge to good solutions quickly. This section presents the various operators and describes their purpose. The descriptions presented here should be detailed enough to enable readers to reimplement the algorithm. However, we sometimes omit minute details for ease of reading. Handling corner cases that would drastically affect the readability and clarity of the algorithms are left out. For the actual implementations, please consult our GitHub[1].

Table 6.2: Summary of mutations used in SMOLT.

Mutations	**Description**
Remove random	Removes a random visit from a random vessel's route
Add random	Adds a random visit to a random vessel's route
Inter-swap	Swaps a random visit between two vessels. Only the ports of the visits are swapped, the time is kept constant
Intra-swap	Swaps the ports between two visits within a vessel's route
2-opt	Normal 2-opt that only considers distance
Time bounce	Push visits that cause time warp away from each other in order to reduce the warp
Ruin & recreate	A set of techniques that destroys solutions before rebuilding them

Mutations, i.e., the operators of the MA targeting single individuals, are the only way of introducing new gene material to the population and are crucial for escaping local minima and constitute exploration. SMOLT employs several mutations to explore the solution space. Some of the mutation techniques we have are general techniques that can be applied to a broad set of problems, while some are tailored for the solution representation and problem at hand. A summary of the mutations in SMOLT is given in Table 6.2. During development, a few other mutations were implemented but found to provide little or no benefit and were subsequently discarded. These can

[1] `https://github.com/fiskeforgutane/master-thesis`

be found in Appendix C.

Ruin & Recreate

Ruin & recreate (R&R) operators are based on destroying the current solution before it is rebuilt using some logic. Within the ALNS literature, this is commonly known as destroy & repair. This section presents three ruin operators employed by SMOLT and the recreate operator used for repairing the solutions afterward.

The first ruin operator destroys the current routing solutions exclusively based on the time period visits occur. A random interval within the planning horizon is selected, and if a visit takes place within the interval, it is removed with a given probability. The two first elements in Figure 6.5 illustrate this ruin operator. Here, the interval is from $t = 4$ to $t = 6$, and all visits within the interval except the grey vessel's visit to port 3 are removed. The grey vessel's visit to port 3 is left in the plan due to the stochastic nature of the removal: it was "lucky" enough to be left as-is.

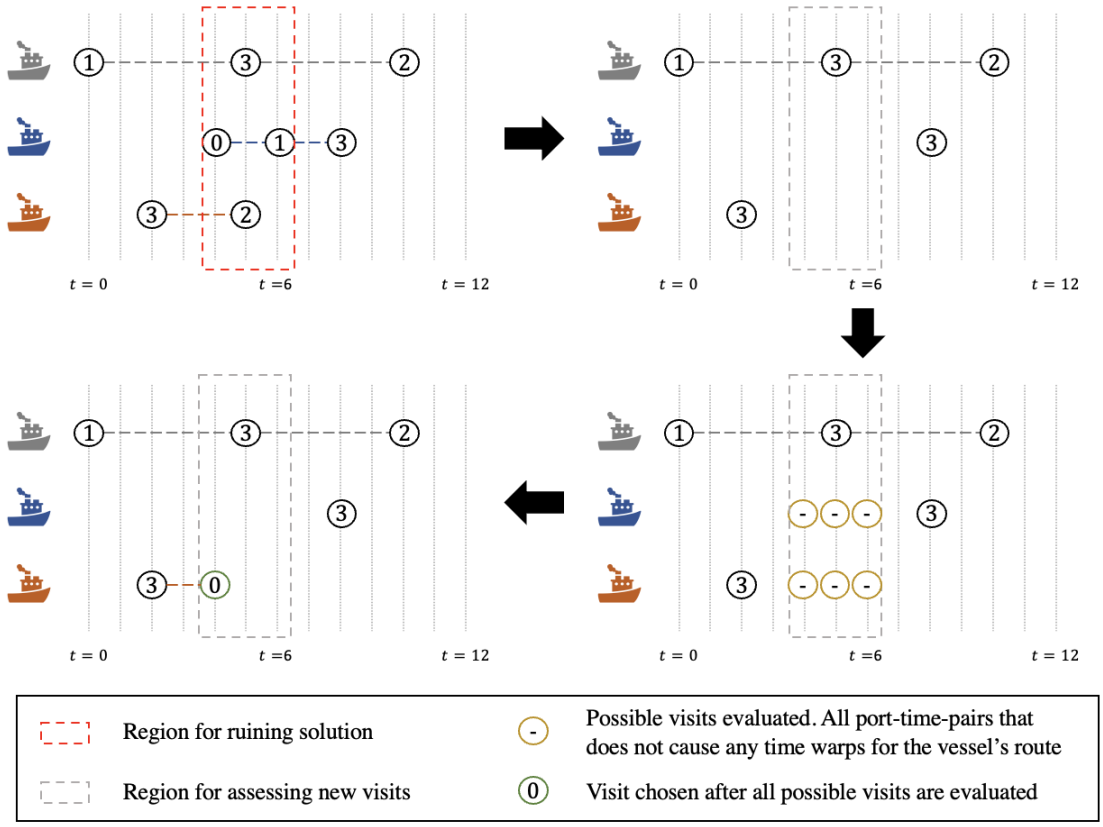


Figure 6.5: R&R based on period. In the first element, we see the routes of three vessels, and the time interval to be ruined marked with a red rectangle. In the second element, the two first visits of the blue vessel and the last visit of the orange vessel have been removed. The grey vessel's route is intact. The third element displays the candidate visits to be evaluated, and the final element indicates that a visit to port 0 was the best according to the fitness of the solution.

Further, we have implemented a ruin operator that only removes visits from one vessel. This operator first selects a random vessel, and then removes each visit in that vessel's route with a given probability.

Lastly, we have included a ruin operator based on Slack Induction by String Removal (SISR) inspired by the work of Christiaens and Vanden Berghe (2020), who applied it to solve different variations of the VRP. The concept is rather simple. When a visit is removed from a candidate

solution, it introduces *slack*. That is, it frees up resources that might allow us to insert one or multiple new visits in the vacancy left behind. In our case, removing a visit frees up the time that was spent traveling to the visit and the time spent loading or unloading, as well as possibly the load that was delivered during the visit. The authors of the original paper conjectured that this slack is most useful when it is focused around a specific area. The slack must be above a certain threshold in order to be useful.

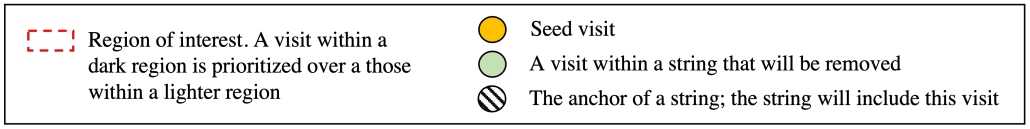
In the original SISR, this targeted slack is realized by removing multiple consecutive visits from the routes of multiple vehicles. They call each such collection of consecutive visits a *string*. By choosing geographically close strings, they are able to introduce targeted slack. This approach was shown to provide state-of-the-art results on the VRP variants they benchmarked.

We have implemented an adaption of the core principles of SISR for the FFMIRP. MIRPs often have a more complex time representation than VRPs and regular IRPs. For this reason, we chose to consider both spatial and temporal distance when determining what visits to remove. A pseudocode is presented in Algorithm 2, which is accompanied by Figure 6.6. Apart from using both space and time to determine proximity, we stay true to the original SISR. We use the same procedure as Christiaens and Vanden Berghe (2020) in order to decide on the number of strings to remove as well as the length of each string. This is outlined in Appendix B.

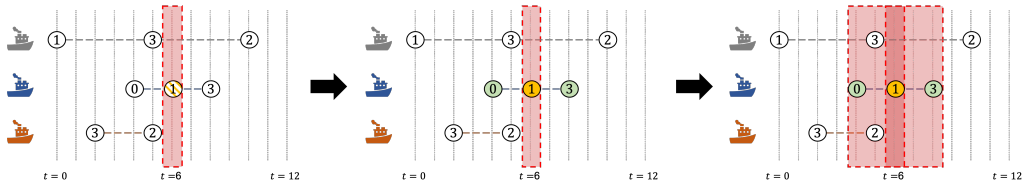
Algorithm 2 SMOLT's SISR-based ruin method

```
                                                          ▷ SISR-based ruin method
function sisr-ruin(solution):
    k ← Decide on the number of strings to remove
    seed ← Select a random visit within solution
    /* An ordered list of the time windows we will consider.  Note that each entry
       in periods is a time period, i.e., a continuous subset of 𝒯                 */
    periods ← {time-of(seed) ... time-of(seed)}
    /* The strings we have selected for removal                                    */
    strings ← ∅
    /* The vessels we have selected a string from                                  */
    used ← ∅
    /* Select k strings for removal                                                */
    repeat k times
        /* If no suitable anchor is found for any period p in periods, we try to
           find one within the full planning period 𝒯                            */
        for p ∈ periods ∪ {𝒯} do
            anchor ← Choose the visit within the time period p that is closest to node-of(seed),
             and whose vessel is not in used
            if anchor ≠ NIL then
                l ← Decide on the length of the string we will select now
                string ← Determine a string of length l that contains anchor
                strings ← strings ∪ string
                periods ← periods ∪ {time period of the newly added string}
                used ← used ∪ vessel-of(string)
                break
    /* Remove the selected strings                                                 */
    solution ← solution with strings removed
    return solution
```

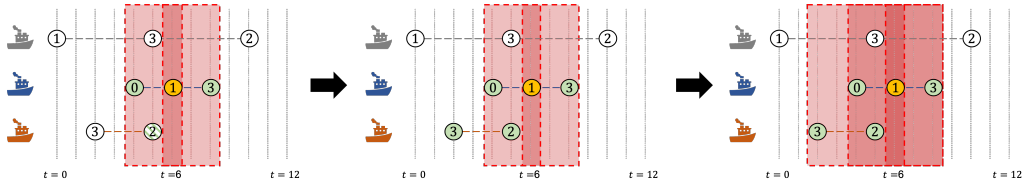
Independent of the ruin operator applied to the solution, the same recreate operator is applied. They all use an adaption of the "greedy with blinks" method as presented in Christiaens and Vanden Berghe (2020). After a routing solution has been ruined, a set of candidate visits are generated. The candidate visits are possible new insertions in the solution and consist of a visit (port and time) and a vessel to perform the visit. Each candidate is evaluated with probability $(1 - \varepsilon)$, and the one with the best fitness is inserted into the routing solution. In the case of



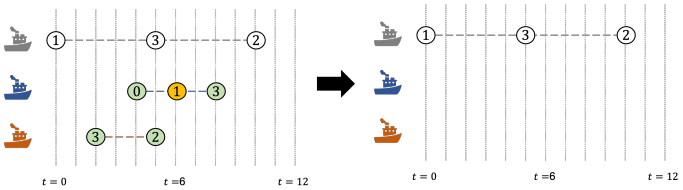
(a) Legend for the figures that follow. Note that an anchor visit is identified by pattern only, not by color.



(b) First string selection. In the first illustration, a seed visit has been chosen randomly. Since the seed visit is the closest visit to itself, it is also chosen as the first anchor. In the transition from the first to the second illustration, we have drawn a random string of length $l = 3$ that includes the anchor chosen in the leftmost picture. The rightmost illustration shows how the time-based region of interest expands after a new string has been selected.



(c) Second string selection. This works similar to the first string removal. In the leftmost illustration, the last visit of the orange boat is chosen as the anchor. The process to decide on this as the anchor proceeded as follows: first, we would try to find the closest visit within the darker red region of interest. Since there are no visits within this period, we proceed to the light red region. Here, the visit at port 2 was chosen, since the port 1 of the seed node is closer to port 2 than to port 3. We then proceed in the same manner as in the first iteration. A string is drawn that includes the chosen anchor, which in this case has length $l = 2$, and include the period occupied by the newly drawn string into the regions of interest.



(d) String removal. After $k = 2$ strings have been selected, they are removed from the routing solution. At this point, we will apply the recreate-procedure in order to rebuild a new solution.

Figure 6.6: Illustration of how the SISR-based ruin operator works.

skipped evaluation for a solution, which happens with probability ε, we assign the solution the worst possible fitness instead of evaluating it. The process outlined above is repeated until no candidates improve the solution.

In Figure 6.5, the recreate operator is applied in step 3 and 4. For the grey vessel, no visits are feasible within the assessed region, as the travel time from port 3, which is already in the route, is higher than one time period to all other ports. Further, we see that for the blue and orange vessels, all three time periods in the interval are feasible insertion points for at least one port. Then, all these insertion points are evaluated with the LP, and the visit to port 0 in time period 4 for the orange vessel was evaluated to be the best insertion. In practice, this process would be repeated until no further improvements were possible.

Time Mutation Operator

To decrease time warp occurrences, we have included a mutation for specifically targeting this problem. The idea is to push the origin of the violated arc forward in time, and the destination

backward, without causing additional time warps. An illustration of the idea is given in the context of a single vessel's route in Figure 6.7. In the left-most situation, the vessel waits for one time period in port 0 and port 3 before it starts serving. Later in the route, the vessel does not arrive at port 1 in time to perform the loading/unloading. By pushing the action at port 0 forward by one time period and arriving directly at port 3 in $t = 2$, we can get to port 1 in time.

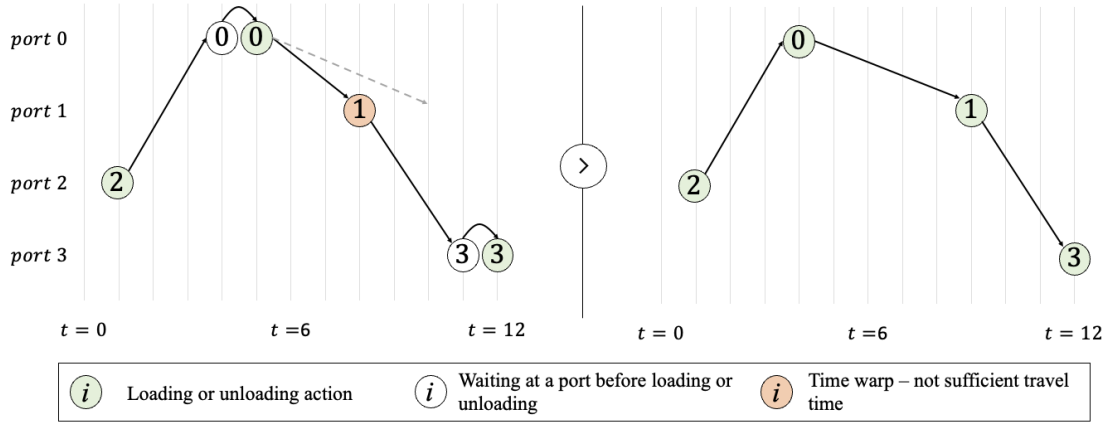


Figure 6.7: Time bounce mutation.

6.5 Other aspects

This section presents three aspects of SMOLT that are implemented in order to ensure feasibility of solutions and scalability of the algorithm. Section 6.5.1 describes how the SMOLT's islanding approach is implemented. Section 6.5.2 presents how we ensure that no products are mixed in the same compartments, even though this is not handled by the quantity assignment LP. Section 6.5.3 presents a warm-start approach used for decreasing the sensitivity for long planning horizons.

6.5.1 Islanding

SMOLT's islanding is implemented by starting separate instances of the memetic algorithm, and using messages to pass data between them. We keep track of the total number of epochs done by each island, and perform a migration once the total number of epochs surpass given thresholds.

For every p epochs, we will retrieve the populations of each island, shuffle them, and then send a subset of a given size q from each population to a different island. Each island is independent of the others at all times except when a migration occurs. This results in minimal amounts of synchronization overhead. In order to avoid resource congestion, we restrict the commercial solver to use a single thread, and typically set the number of islands to be at most equal to the number of processor cores. Consequently, each island is effectively pinned to a single core.

6.5.2 Avoiding Mixing Products

As mentioned in Chapter 3, products cannot be mixed when transported in a vessel. However, Section 6.2.2 states that the LP handling all quantity decisions does not respect any other constraints than basic inventory limit constraints. If the LP were to assign the quantities to specific compartments as well, it would become a MIP which is not computationally viable as explained in Section 6.3. Therefore, SMOLT is applied as if there were no restrictions on mixing products. When SMOLT terminates, an extended version of the LP that handles compartment assignment is applied to the best-found solution. The extended version is available in Appendix A.5.3. This does

not affect the transportation cost of the best-found solution as the LP does not alter the routes. However, if the quantities are reduced due to the compartment assignment, the revenue is reduced, and it might become necessary to buy more from the spot market. This approach is suboptimal as SMOLT does not consider the mixing constraints. However, it is believed that the optimal routes with and without these constraints are relatively similar in practice. This, combined with the computational efficiency of having an LP rather than a MIP, substantiate why this approach is reasonable.

6.5.3 Gradually Extended Horizon

As discussed multiple times, the problem at hand is highly complex. The complexity of the problem increases rapidly with the problem size, and thus, decomposition principles can be handy when dealing with large problem instances. This section proposes a novel decomposition technique based on the time periods.

The idea is based on iteratively building the solutions by first solving a problem with a reduced number of time periods. The simplified problem's solution is then used as the initial solution for a problem with a longer planning horizon. This process is repeated until the desired time periods are incorporated into the solution. In other words, our decomposition based on a gradually extended horizon can be considered a way of using the solutions of a less complex problem to warm-start a more complex problem.

In our implementation, we start solving the problems using 30 time periods. The problem is extended to 32 time periods in the next iteration. 2 and 2 time periods are then added iteratively. SMOLT is run for a user-defined amount of time in each step before moving on to the next step.

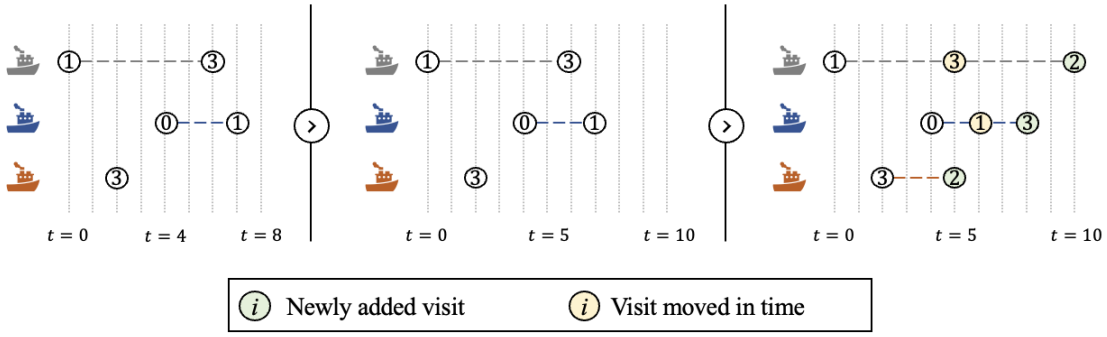


Figure 6.8: A schematic illustration of the decomposition approach employed in SMOLT. The leftmost element displays the situation after solving the problem for eight time periods. The second element displays the situation when two additional time periods are added, but before a new solution is found. Finally, the rightmost element displays the new solution found for ten time periods when the algorithm is warm-started with the eight time periods solution.

Chapter 7

Data and Test Instances

This chapter presents the data and test instances used in the computational study presented in Chapter 8, where SMOLT is applied to two different sets of problem instances. The first set contains instances based on data provided by our industry collaborator, Mowi, which produces and distributes fish feed to fish farms along the Norwegian coastline. The other set contains some of the instances found in the benchmark library MIRPLib, presented by Papageorgiou et al. (2014c).

Section 7.1 presents how the data provided by Mowi is used to replicate farms and vessels that can be used in the problem instances. Next, Section 7.2 presents the MIRPLib benchmark library, and lastly, Section 7.3 presents all the test instances. The objective of the computational study is to observe the performance of SMOLT, presented in Chapter 6. Therefore, the Mowi test instances are carefully designed to give an impression of both the achieved solution quality and the scalability of the algorithm.

Some parts of Section 7.1 are retrieved from our specialization project, Bjelland et al. (2021), however, the method used to generate test instances has been modified.

7.1 Mowi

The Mowi instances are based on data provided by Mowi and Brekkå and Randøy (2021). This includes data on production facilities, farms, vessels, as well as feed consumption forecasts from Mowi's internal forecasting software. In cases where data is missing or incomplete, we have filled the gaps using online research and our own assumptions.

7.1.1 Fish Farms

Figure 7.1 shows a map of the locations of the farms and factories that are included in the test instances. The data consists of a selection of 89 Mowi farms and their production facility at Bjugn and is based on historical orders. In addition, Mowi provided 11 feed consumption scenarios for every farm. The scenarios are relatively similar, and to avoid more test instances than possible to process in the computational study, all instances are generated based on one consumption scenario. In this scenario, the farms consume 13 different feed products every week in total. As the problem complexity is highly dependent on the number of products, we can reduce it by aggregating some products. However, this must be done with care to obtain realistic problem instances. First of all, the six most consumed products in the chosen scenario constitute 95% of the total demand. In addition, most of the products with low demand have the same weight and size as one with high demand. Therefore, a simplifying measure is to consider all product types of the same weight and size as the same product. This aggregation results in six products in demand, which reduces the problem complexity significantly.

Figure 7.1: The farms (blue) and factory (red) belonging to Mowi.

The data does not include any information on the total feed capacity at the farms. Every farm must therefore be assigned a capacity. A farm's capacity is set to 40% more than the single largest quantity delivered of a product among the historical deliveries in the data. Next, every farm is assigned an initial inventory of the six aggregated product types. The initial inventory significantly impacts the difficulty of finding solutions where the vessels are capable of satisfying all demand. We therefore introduce three levels of difficulty, A, B, and C. At level A, all farms are given an initial inventory of six feed days, i.e., the farms will run empty after six days, given no deliveries. At level B, the farms are assigned an initial inventory between four and eight feed days, and at level C, it is set between two and eight feed days. The number of feed days is drawn from a uniform distribution for levels B and C. Next, every farm has an assigned lower limit for every feed type, which is set to zero for all farms and product types. Lastly, the berth capacity at all farms is set to one, and it is set to four at the factory, where the latter means that up to four vessels can load at the factory simultaneously.

The chosen consumption forecast in the data is used to calculate the consumption rates at the farms. The forecast shows the expected consumption of each product for a week. For simplicity, the demand in every time period is set to the forecasted demand adjusted for the length of a time period. This means that the demand for every feed type is constant; however, both the mathematical model presented in Chapter 5 and the solution algorithm SMOLT presented in Chapter 6 are capable of handling varying demands.

Next, it is not given that it is possible to keep the inventory at all farms above zero by only delivering feed from the vessels. Some inventory breaches can be impossible to handle given the farms' initial inventories and the vessels' initial loads and positions. However, as mentioned in Chapter 3, it is allowed to handle such cases by ordering deliveries from a spot market. To ensure that the problem instances are guaranteed to be feasible, we impose no restrictions on the total quantity ordered from the spot market. The cost per quantity bought at the spot market is set to twice the round trip cost of visiting the farm furthest away from the factory for the most expensive vessel, divided by the vessel's capacity.

As mentioned in Section 6.2.2, a revenue is given as a reward per quantity delivered at a farm to reduce negative end-effects. The revenue in the Mowi instances is set to half the spot market cost per quantity.

Lastly, the data contains information about areas with disease outbreaks, and hence red, green, and yellow zones, as described in Section 2.3.1. This could have been included in our test instances. However, neither red, green, nor yellow zones are directly reflected in our mathematical model or SMOLT. Instead, the zones are handled by modifying the travel times between ports, as described in Section 5.3.4. Therefore, the zone information in the data is ignored, as it would only affect the travel times. All farms are thus treated as green, and the travel times between them are set to the

actual travel times for the vessels.

7.1.2 Vessels

The vessels used are presented in Table 7.1. These vessels correspond to actual vessels operated by Mowi or BioMar. BioMar is another fish feed producer distributing feed along the Norwergian coastline. The capacity and speed data originate from Mowi or BioMar, or is collected through relevant web pages. The travel costs are calculated based on LPG prices from 2022. Next, the fixed cost per hour is assumed to be driven by the crew's salary. The crew size is set to five and each crew member is assigned a cost of 350 NOK per hour, yielding an hourly fixed cost of 1,750 NOK.

Table 7.1: The name, capacity, cruising speed, travel cost, and fixed costs for the vessels considered in the test instances.

Vessel Name	**Capacity**		**Cruising Speed**	**Travel Cost**	**Fixed Cost**
	[tons]	[#silos]	[knots]	[NOK/hour]	[NOK/hour]
With Harvest	3,000	11	13	4,813	1,750
Høydal	2,006	28	11	4,147	1,750
Mikal with	1,110	13	10	3,702	1,750
Vågsund	1,306	26	10	2,649	1,750

In the FFMIRP, the vessels can become available at any factory or farm in any time period within the planning horizon. The origin of a vessel is with a probability of 50% assigned to the factory and with a probability of 50% to any of the farms. If a vessel is assigned a farm as its origin, the probability of selecting a specific farm is set proportional to how soon the farm would run empty of a product. This is considered a fair assumption as vessels are more likely to visit a farm with low stock than one with high stock. Next, the time period when the vessels become available is randomly drawn from a uniform probability distribution from zero to ten, where the duration of one time period is one hour. This procedure for assigning origins and when vessels become available is intended to make the test instances more realistic, as the vessels in reality become available at different locations and times.

Furthermore, the FFMIRP allows vessels to have an initial load. In reality, vessels visiting a factory are often empty, while they have some load when visiting a farm. In the test instances, we assign the initial loads to reflect this. If the vessel becomes available at a factory, we assign an initial inventory of zero. Next, if a vessel becomes available at a farm it starts fully loaded, and its initial inventory of every product is set proportional to the total consumption of the product across all farms.

As mentioned in Chapter 3, all vessels cannot usually visit all farms. To recall, there are at least two potential reasons why a vessel cannot visit a farm. Firstly, a bulk vessel cannot deliver feed to a bag farm, and secondly, physical constraints such as shallow waters can make it impossible for the vessel to dock at a farm. In our case, all farms and vessels are considered bulk, and according to Mowi, there are currently no vessels that cannot visit a farm due to physical constraints. All vessels are therefore allowed to visit all farms in the test instances.

7.2 MIRPLib

In Papageorgiou et al. (2014c), a library of benchmark problems for the class of deterministic, single-product MIRPs is presented. The primary purpose of including some MIRPLib instances in the computational study is to verify that SMOLT can produce high-quality solutions. In general, only toy size instances of MIRPs and FFMIRPs can be solved to optimality with a commercial solver. Hence, for larger instances, it is difficult to evaluate the quality of solutions generated by

approximation methods by comparison to solutions of a commercial solver. Since several different algorithms have been applied to the MIRPLib instances, it allows us to compare SMOLT's performance to the best-known solution methods.

The instances found in MIRPLib differ from the FFMIRP instances along a couple of dimensions. However, they are similar enough to evaluate SMOLT's performance on variants of the MIRP compared to other solution methods. If SMOLT can perform well on the MIRPLib instances, it probably also performs well when applied to FFMIRP instances. The first difference between MIRPLib and FFMIRP is that the MIRPLib instances are deep-sea problems, and the ports are located in discharging and loading regions. The distances between ports within one region are short, while the distances between regions are long. Furthermore, unlike the FFMIRP, the production ports have a production rate and an upper capacity limit. Consequently, the production ports must be visited regularly to keep the inventories below the capacity limits, and a vessel cannot fully load unless there is enough product available at the time of the visit. This difference distinguishes the MIRPLib instances the most from the FFMIRP instances. Next, the MIRPLib instances require vessels to arrive at production ports empty and leave them fully loaded. The purpose of these constraints is to fully utilize the vessel capacity, even though it has been shown that such an assumption may not be optimal (Fodstad et al., 2010). Lastly, the MIRPLib instances require all loading and unloading variables to be semi-continuous. Thus, if a loading or unloading activity is undertaken, the quantity loaded or unloaded must lie in the interval $[min, max]$, where $min > 0$. Requiring fully exploitation of vessel capacity and semi-continuous loading and unloading quantities are convenient when using MIP techniques to solve the MIPRLib instances as it reduces the search space. However, disregarding such requirements in solution methods similar to SMOLT is not a problem. Nevertheless, SMOLT is modified to accommodate all differences between the MIRPLib and FFMIRP instances, which is described in Appendix A.5. Despite not being specifically designed to solve MIRPLib instances, SMOLT's performance presented in Section 8.3 suggests that it achieves high quality solutions for these instances.

The MIRPLib library contains three main groups of problems, referred to as Group 1, 2, and 3. The instances of Group 1 have a planning horizon of 360 periods. These instances have ports organized in regions, and each region can contain multiple ports. Also, split pickups and split deliveries are often necessary to satisfy demand, meaning that the vessels must coordinate some pickups and deliveries at some ports within a small time window. Finding a feasible solution to these problems is considered a challenge, and no solutions are reported for instances with a planning horizon longer than 60 time periods. The instances of Group 2 never require split pickups or split deliveries; however, the travel times are longer, ranging from 5 to 37 time periods. For these instances, solutions are reported for planning horizons longer than 60 time periods. The Group 3 instances are provided by Jiang and Grossmann (2015).

As Group 1 is considered the most challenging group, the instances of this group are chosen to take part in the computational study. Group 2 and 3 are left out as there are some minor differences to Group 1 that would require some modifications of SMOLT. The included instances are presented in Section 7.3.

7.3 Test Instances

This section presents the test instances that take part in our computational study presented in Chapter 8. The problem complexity of the FFMIRP, described in Chapter 3, and the MIRPLib instances is assumed to be sensitive to the following factors: the total number of ports $|\mathcal{I}|$, the total number of vessels $|\mathcal{V}|$, the number of products $|\mathcal{P}|$, and the number of time periods $|\mathcal{T}|$. Further, the set of ports is segmented into two parts – $|\mathcal{I}^C|$ indicating the number of consumption ports, i.e., farms, and $|\mathcal{I}^P|$ indicating the number of factories. The problem complexity increases with the size of these sets because it increases the size of the search space.

The presentation of instances is twofold, starting with the Mowi instances in Section 7.3.1 and followed by the Group 1 instances of the MIRPLib library in Section 7.3.2.

7.3.1 Mowi Instances

Table 7.2 summarizes the different configurations of farms, factories, vessels, products, and time periods that are used to generate test instances. When sampling farms, the 89 Mowi farms are grouped into three groups of sizes 30, 30, and 29, depending on their latitude coordinate. This resulted in three groups named *south*, *middle*, and *north*. These groups are visualized in Figure 7.2. When sampling a subset of the farms, an equal number of farms are sampled from the three groups to ensure that the included farms are geographically dispersed. This causes the problem instances to be more realistic and not too easy to solve. The sampling of vessels is done by including every vessel an equal number of times and duplicating the vessels if necessary. The sampling of products is done by taking the p most consumed products where p is the number of products to include in the test instance.

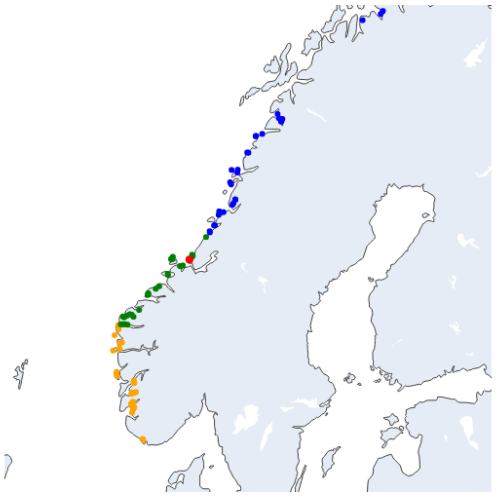


Figure 7.2: The three groups of farms, south (orange), middle (green), north (blue). Note the factory marked in red.

For every configuration in Table 7.2, six or nine test instances are generated using the planning horizon lengths in column $|\mathcal{T}|$, and the three difficulty levels, A, B, and C, presented in Section 7.1.1. E.g., in the first configuration, **p-05-02-02**, five of the 89 farms, the factory, two of the four vessels, and two products are sampled. Due to the problem complexity, the mathematical model described in Chapter 5 cannot be solved to optimality by commercial solvers for large problem instances. As the problem size grows, even finding a feasible solution in a reasonable time proves difficult. The set of configurations is therefore split into three subsets. The first subset contains configurations resulting in instances small enough for a commercial solver to at least obtain reasonable solutions. These instances are therefore used to compare SMOLT's performance with the mathematical model on small instances. The two other configuration subsets results in larger test instances used for SMOLT performance testing on more realistic instances. These two subsets are referred to as the medium and large instances, respectively. In total, 132 Mowi test instances are created.

Table 7.2: Test instance configurations used for the Mowi instances. The instances in the upper part separated with a horizontal line are attempted solved with a commercial solver, while those in two lower parts are not. For configurations in the upper part, six instances are generated, while nine instances are generated for the ones in the lower parts. The three parts are referred to as small, medium, and large instances, respectively. The ID format is $|\mathcal{I}^C|$-$|\mathcal{V}|$-$|\mathcal{P}|$. The prefix "p" stands for problem.

ID	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{I}^P\rvert$	$\lvert\mathcal{I}^C\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{P}\rvert$	Level
p-05-02-02	145, 290	6	1	5	2	2	A, B, C
p-05-02-04	145, 290	6	1	5	2	4	A, B, C
p-10-02-02	145, 290	11	1	10	2	2	A, B, C
p-10-02-04	145, 290	11	1	10	2	4	A, B, C
p-20-06-01	145, 235, 290	21	1	20	6	1	A, B, C
p-20-06-03	145, 235, 290	21	1	20	6	3	A, B, C
p-20-06-06	145, 235, 290	21	1	20	6	6	A, B, C
p-40-07-01	145, 235, 290	41	1	40	7	1	A, B, C
p-40-07-03	145, 235, 290	41	1	40	7	3	A, B, C
p-40-07-06	145, 235, 290	41	1	40	7	6	A, B, C
p-60-10-01	145, 235, 290	61	1	60	10	1	A, B, C
p-60-10-03	145, 235, 290	61	1	60	10	3	A, B, C
p-60-10-06	145, 235, 290	61	1	60	10	6	A, B, C
p-80-10-01	145, 235, 290	81	1	80	10	1	A, B, C
p-80-10-03	145, 235, 290	81	1	80	10	3	A, B, C
p-80-10-06	145, 235, 290	81	1	80	10	6	A, B, C

In Table 7.2, the leftmost column represents the configuration identification, where each of the numbers indicates the corresponding size of the set on the corresponding position in the instance ID. Please note that when referring to specific instances one also needs to include the number of time periods and the difficulty level. Therefore, when referring to the instance **p-05-02-02** with 145 time periods and level A the ID is extended to **p-05-02-02-145-A**.

In all test instances, the length of one time period is set to one hour, as sailing times between farms typically range from a few hours to a couple of days. The service times are often approximately one hour, and using a time period length of one hour therefore results in activities lasting at least one or several time periods.

7.3.2 MIRPLib Instances

All problem instances from Group 1 in the MIRPLib library are included in the computational study. The number of ports and vessels in each instance range from 4 to 13 and 6 to 17, respectively. No solution for a planning horizon longer than 60 time periods has been reported for any of the Group 1 instances. However, we include planning horizons of 90 and 120 time periods in the computational study. The instances are summarized in Table 7.3. The naming convention is the number of loading regions, the number of loading ports in each loading region, the number of discharge regions, the number of discharge ports in each discharge region, the number of vessel classes, and the number of vessels. Finally, if a letter is included at the end, this is to distinguish this instance from other instances.

Table 7.3: The Group 1 MIRPLib instances included in the compuational study.

ID	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{I}^P\rvert$	$\lvert\mathcal{I}^C\rvert$	$\lvert\mathcal{V}\rvert$
LR1-1-DR1-3-VC1-V7a	45, 60, 90, 120	4	1	3	7
LR1-1-DR1-4-VC3-V11a	45, 60, 90, 120	5	1	4	11
LR1-1-DR1-4-VC3-V12a	45, 60, 90, 120	5	1	4	12
LR1-1-DR1-4-VC3-V12b	45, 60, 90, 120	5	1	4	12
LR1-1-DR1-4-VC3-V8a	45, 60, 90, 120	5	1	4	8
LR1-1-DR1-4-VC3-V9a	45, 60, 90, 120	5	1	4	9
LR1-2-DR1-3-VC2-V6a	45, 60, 90, 120	5	2	3	6
LR1-2-DR1-3-VC3-V8a	45, 60, 90, 120	5	2	3	8
LR2-11-DR2-22-VC3-V6a	45, 60, 90, 120	6	2	4	6
LR2-11-DR2-33-VC4-V11a	45, 60, 90, 120	8	2	6	11
LR2-11-DR2-33-VC5-V12a	45, 60, 90, 120	8	2	6	12
LR2-22-DR2-22-VC3-V10a	45, 60, 90, 120	8	4	4	10
LR2-22-DR3-333-VC4-V14a	45, 60, 90, 120	13	4	9	14
LR2-22-DR3-333-VC4-V17a	45, 60, 90, 120	13	4	9	17

Chapter 8

Computational Study

This chapter presents the results of the computational study performed on the two sets of test instances introduced in Section 7.3. In Section 8.1, we present the test environment in which the computational study was performed and the parameters used. Further, Section 8.2 presents the results obtained when applying SMOLT to the Mowi instances. The section is divided into three parts – first, SMOLT is compared to the commercial solver on a set of small instances before its performance on the medium and large instances is presented. Section 8.3 presents the results from applying SMOLT to the MIRPLib instances introduced in Papageorgiou et al. (2014c) and compares the results with the currently best-known solutions. Lastly, Section 8.4 discusses and presents some examples of how SMOLT can be used as a decision-support tool when high-level strategic decisions are made.

8.1 Test Environment

Table 8.1 presents a summary of the hardware and software used for the computational study, while Table 8.2 presents the parameters used by SMOLT. For the Mowi and MIRPLib instances, SMOLT was configured differently. The Mowi instances were run with one island per core and would terminate after three hours or if SMOLT found a feasible solution and could not improve it within 30 minutes. For MIRPLib, we ran SMOLT using gradually extending horizon starting at $t = 30$ with increments of $\delta = 2$. The horizon was extended whenever SMOLT failed to find a better solution within five minutes. At selected checkpoints, e.g., $t = 45$ and $t = 60$, we let it run until it was unable to find a better solution within 30 minutes. In addition, a hard timeout of two hours was employed for $t \leq 60$. No hard timeout was used for $t > 60$.

Table 8.1: Hardware and software used for computational study.

Processor	2 x Intel E5-2670v3 @ 2.3GHz
Memory	64GB
Commercial solver	Gurobi v9.5.1
Programming language	Rust v1.60.0

Table 8.2: SMOLT parameters used for the computational study.

Element	**Parameters**
PIX	$p = 10\%$
Add random	$p = 1\%$
Remove random	$p = 1\%$
Interswap	$p = 1\%$
Intraswap	$p = 1\%$
2 opt	$p = 1\%$
Time bounce	$p = 1\%$
R&R period	$p = 10\%$, interval size $= 6$
R&R vessel	$p = 5\%$
R&R SISR	$p = 2\%$, $\bar{c} = 2$, $L^{max} = 10$
Blink rate (recreation)	5%
k-Tournament	$k = 2$
Population size per island	3
Number of islands	1 or 24

SMOLT parameters were mostly chosen through trial and error. The large number of potential parameters combined with long solution times made it infeasible to perform an exhaustive parameter tuning. However, some systematic testing was performed. We evaluated the performance of each mutation with different parameters on the largest MIRPLib Group 1 instance. A minimal set of mutations consisting of "Add random", "Remove random", and swaps was selected without further evaluation. Each remaining mutation was evaluated in isolation by adding it to the minimal set of mutations and systematically trying a few different parameters. The parameters of each mutation that seemed to perform the best in terms of solution quality and runtime were chosen.

We chose to implement SMOLT in the programming language Rust as it has high performance, a strong emphasis on memory safety, and a focus on making it easy to write correct concurrent and parallel code. In addition, Rust has good interoperability with Python. This allowed us to use Python for visualization and debugging during development. Further, Gurobi (hereafter referred to as the commercial solver) was used for solving the LP included in SMOLT. Gurobi provides state-of-the-art efficiency and has a Rust API, allowing the mathematical models to be seamlessly integrated with the rest of the code.

8.2 Mowi Instances

This section presents the results when applying SMOLT to the Mowi instances presented in Chapter 7. As mentioned, the instances are split into three subsets – small, medium, and large. It is possible to obtain relatively high-quality solutions when applying the commercial solver to the small instances, even though none can be solved to optimality. Section 8.2.1 presents the results when applying both the commercial solver and SMOLT to the small instances. The commercial solver cannot produce any valuable results for the medium and large instances. Sections 8.2.2 and 8.2.3 therefore only present SMOLT's performance on the medium and large instances, respectively.

8.2.1 Small Instances – Comparison with Commercial Solver

In Table 8.3 a summary of the number of variables and constraints for the small test instances is given. Recall three instances with difficulty levels, A, B, and C, were created for every configuration of nodes, vessels, products, and time periods. The only difference between these instances is

the farms' initial inventory. Hence, the problem sizes of the three levels are identical, and Table 8.3 therefore only reports values for each configuration of nodes, vessels, products, and time periods. The columns "#Cont. variables" and "#Int. variables" indicate the number of continuous and integer variables, respectively, while the column "#Constraints" indicates the number of constraints. The leftmost column shows the instance configurations. Table 8.3 clearly shows that the number of variables and constraints increase with the size of the input sets.

Table 8.3: Description of the problem sizes for the mathematical model for the different test instances. The reported values are before presolve is performed.

	#Cont. variables	*#Int. variables*	*#Constraints*
p-05-02-02-145	94,540	896,382	50,107
p-05-02-04-145	189,080	907,692	78,743
p-10-02-02-145	130,210	1,902,096	57,070
p-10-02-04-145	260,420	1,913,406	88,316
p-05-02-02-290	189,080	3,558,872	100,277
p-05-02-04-290	378,160	3,581,492	157,623
p-10-02-02-290	260,420	7,588,706	114,200
p-10-02-04-290	520,840	7,611,326	176,766

Table 8.4 presents the computational results from solving the small instances using the commercial solver. The results of the commercial solver were reported at two separate times – after 3,600 and 7,200 seconds. By reporting results twice, it is possible to observe the progress of the commercial solver throughout the run. The upper time limit is set to 7,200 seconds as the progression seems to stagnate after around two hours.

Table 8.4: Computational results for the commercial solver on the small test instances.

	Time Limit 3,600s				**Time Limit 7,200s**				**Improvement**	
	Primal bound	*Dual bound*	*Gap* [%]	*Time* [s]	*Primal bound*	*Dual bound*	*Gap* [%]	*Time* [s]	*Primal* bound [%]	*Gap* [p.p.]
p-05-02-02-145-A	-1,151,963	-1,233,521	7.1	3594	-1,151,963	-1,220,163	5.9	6201	0.0	1.2
p-05-02-02-145-B	-952,655	-1,050,532	10.3	3589	-952,655	-1,045,727	9.8	7110	0.0	0.5
p-05-02-02-145-C	-1,084,341	-1,245,305	14.8	3572	-1,084,341	-1,245,305	14.8	7176	0.0	0.0
p-05-02-04-145-A	-1,485,345	-1,861,032	25.3	3512	-1,486,091	-1,850,913	24.5	7179	0.1	0.8
p-05-02-04-145-B	-1,450,665	-1,775,408	22.4	3561	-1,450,665	-1,769,404	22.0	7126	0.0	0.4
p-05-02-04-145-C	-1,678,706	-1,959,103	16.7	3577	-1,683,418	-1,899,830	12.9	7121	0.3	3.8
p-10-02-02-145-A	-1,311,177	-1,795,547	36.9	3515	-1,311,177	-1,631,920	24.5	7191	0.0	12.4
p-10-02-02-145-B	-1,375,116	-1,646,025	19.7	3562	-1,444,476	-1,644,635	13.9	7067	5.0	5.8
p-10-02-02-145-C	-1,554,698	-1,880,353	20.9	3570	-1,558,110	-1,880,353	20.7	7152	0.2	0.2
p-10-02-04-145-A	-1,277,314	-2,767,810	117.0	3588	-1,826,526	-2,767,810	51.5	7,200	43.0	65.5
p-10-02-04-145-B	-2,126,800	-2,609,166	22.7	3590	-2,173,842	-2,573,315	18.4	7100	2.2	4.3
p-10-02-04-145-C	-2,361,876	-2,939,039	24.4	3544	-2,436,823	-2,939,039	20.6	7093	3.2	3.8
p-05-02-02-290-A	-1,194,671	-1,876,084	57.0	3500	-1,398,033	-1,875,969	34.2	7087	17.0	22.8
p-05-02-02-290-B	-1,326,793	-1,824,505	37.5	3577	-1,334,645	-1,824,505	36.7	7141	0.6	0.8
p-05-02-02-290-C	-1,418,548	-1,976,973	39.4	3395	-1,427,038	-1,976,973	38.5	7,200	0.6	0.9
p-05-02-04-290-A	-1,311,285	-2,741,223	109.0	3330	-1,313,523	-2,741,223	109.0	7,200	0.2	0.0
p-05-02-04-290-B	-	-2,655,191	-	2502	-	-2,655,165	-	5958	-	-
p-05-02-04-290-C	-503,224	-2,803,955	457.0	3113	-503,224	-2,803,930	457.0	5162	0.0	0.0
p-10-02-02-290-A	-1,234,663	-2,498,996	102.0	3422	-1,622,675	-2,498,484	54.0	7114	31.4	48.0
p-10-02-02-290-B	-1,027,368	-2,410,944	135.0	2404	-1,581,153	-2,410,944	52.5	7169	53.9	82.5
p-10-02-02-290-C	-1,861,315	-2,580,220	38.6	3506	-2,039,024	-2,580,220	26.5	6979	9.5	12.1
p-10-02-04-290-A	-	-3,827,517	-	3294	-	-3,827,491	-	5801	-	-
p-10-02-04-290-B	-	-3,671,591	-	3527	-	-3,671,269	-	7072	-	-
p-10-02-04-290-C	-	-4,007,156	-	3463	-	-4,007,131	-	6335	-	-

The leftmost column indicates the problem instance. The table is split into three main sections – the results after 3,600 seconds and 7,200 seconds, respectively, and the improvement between the two times. The columns "Primal bound" indicate the objective value of the best-found feasible solution. Next, the columns "Dual bound" indicate the best lower bound found, and the columns "Gap [%]" show the gap between the primal and dual bound in percentage. This gap is calculated as $|1 - \frac{dual\ bound}{primal\ bound}|$. A non-zero gap means that the commercial could not find a feasible solution that was proven optimal. Lastly, the columns "Time [s]" indicate the time at which the current best found solution was last reported. The time is less than the time limit in some cases, which means that the commercial solver did not report any new values from this time until the time limit was reached. The last section presenting the improvement consists of the columns "Primal bound [%]" and "Gap [p.p.]," which indicate the improvement of the primal bound and the percentage point improvement of the gap, between 3,600 and 7,200 seconds of running time, respectively.

The results show that even for relatively small instances, the commercial solver cannot solve any instances to optimality. For the smallest instance, **p-05-02-02-145-A** it obtains a gap of 5.9%. However, for the larger instances, such as **p-10-02-04-290-A**, the solver does not even find a feasible solution. This is expected as the problem complexity quickly increases with the size of the input sets. Furthermore, the results indicate that the solver's progress is modest between 3,600 and 7,200 seconds for several instances. However, there are a couple of exceptions, such as instance **p-10-02-02-290-B** where the primal bound is improved by 53.9% and the gap is reduced by 82.5 percentage points.

One interesting observation is that in some cases, it seems to be easier for the commercial solver to obtain a relatively tight cap for C-instances compared to their corresponding A-instances. This is surprising as the A-instances are intended to be easier to solve than the C-instances. To recall, all farms are given an initial inventory of six feed days in the A-instances, while the initial inventory is randomly set between two and eight feed days in the C-instances. The reduced flexibility in the C-instances might cause the observed behavior. This reduced flexibility results in a smaller search space for the commercial solver to investigate, which might be the reason for the smaller gaps.

Table 8.5 presents a comparison between SMOLT's and the commercial solver's performance on the small Mowi instances. The table is structured in three sections, where the first section shows the commercial solver's performance after 7,200 seconds, which is also shown in Table 8.4. The column "Lowest objective" corresponds to the "Primal bound" column in Table 8.4, the column "Gap [%]" shows the optimality gap, and the column "Time [s]" presents the time in seconds. Further, the SMOLT section consists of four columns. The columns "Lowest objective" and "Time [s]" are the same as in the section for the commercial solver. However, the columns "Gap [%]" and "Beat time [s]" are also included in this section. "Gap [%]" indicates the optimality gap obtained when using SMOLT's lowest objective and the commercial solver's obtained dual bound. The solution identified by SMOLT is guaranteed to be within the gap percentage of the optimal solution. The column "Beat time [s]" shows the time it took for SMOLT to beat the commercial solver if it managed to do so. Lastly, the "Difference" section presents the relative and absolute difference in both objective value and the time to obtain the best solution found throughout the run.

Table 8.5: Comparison of the performance of SMOLT and the commercial solver.

	Commercial Solver			**SMOLT**				**Difference**	
	Lowest objective	*Gap* [%]	*Time* [s]	*Lowest objective*	*Gap* [%]	*Beat time* [s]	*Time* [s]	*Objective* [%]/[abs]	*Time* [%]/[s]
p-05-02-02-145-A	-1,151,963	5.9	6201	-1,151,979	5.9	2075	2075	0/-15	-67/-4,126
p-05-02-02-145-B	-952,655	9.8	7110	-903,666	15.7	-	536	-5/+48,989	-92/-6,574
p-05-02-02-145-C	-1,084,341	14.8	7176	-1,029,273	21.0	-	848	-5/+55,068	-88/-6,328
p-05-02-04-145-A	-1,486,091	24.5	7179	-1,487,259	24.5	738	781	0/-1,167	-89/-6,398
p-05-02-04-145-B	-1,450,665	22.0	7126	-1,170,041	51.2	-	426	-19/+280,624	-94/-6,700
p-05-02-04-145-C	-1,683,418	12.9	7121	-1,349,377	40.8	-	3148	-20/+334,041	-56/-3,973
p-10-02-02-145-A	-1,311,177	24.5	7191	-1,171,482	39.3	-	880	-11/+139,694	-88/-6,311
p-10-02-02-145-B	-1,444,476	13.9	7067	-1,223,293	34.4	-	805	-15/+221,183	-89/-6,262
p-10-02-02-145-C	-1,558,110	20.7	7152	-1,326,327	41.8	-	549	-15/+231,782	-92/-6,603
p-10-02-04-145-A	-1,826,526	51.5	7,200	-2,239,153	23.6	510	3128	+23/-412,627	-57/-4,072
p-10-02-04-145-B	-2,173,842	18.4	7100	-2,026,134	27.0	-	5932	-7/+147,707	-16/-1,168
p-10-02-04-145-C	-2,436,823	20.6	7093	-2,294,560	28.1	-	1537	-6/+142,263	-78/-5,556
p-05-02-02-290-A	-1,398,033	34.2	7087	-1,539,748	21.8	529	561	+10/-141,714	-92/-6,526
p-05-02-02-290-B	-1,334,645	36.7	7141	-1,389,950	31.3	2641	2777	+4/-55,304	-61/-4,364
p-05-02-02-290-C	-1,427,038	38.5	7,200	-1,557,887	26.9	1285	2047	+9/-130,848	-72/-5,153
p-05-02-04-290-A	-1,313,523	109.0	7,200	-1,708,972	60.4	148	2485	+30/-395,449	-65/-4,715
p-05-02-04-290-B	-	-	5958	-2,103,859	26.2	8	7110	-/-	+19/+1,152
p-05-02-04-290-C	-503,224	457.0	5162	-1,790,312	56.6	8	3168	+256/-1,287,088	-39/-1,994
p-10-02-02-290-A	-1,622,675	54.0	7114	-1,721,215	45.2	1240	3666	+6/-98,540	-48/-3,448
p-10-02-02-290-B	-1,581,153	52.5	7169	-1,743,349	38.3	583	6962	+10/-162,196	-3/-207
p-10-02-02-290-C	-2,039,024	26.5	6979	-1,640,710	57.3	-	2080	-20/+398,314	-70/-4,899
p-10-02-04-290-A	-	-	5801	-2,565,496	49.2	8	2160	-/-	-63/-3,641
p-10-02-04-290-B	-	-	7072	-2,268,751	61.8	9	3102	-/-	-56/-3,970
p-10-02-04-290-C	-	-	6335	-2,564,049	56.3	9	927	-/-	-85/-5,408

The results show that SMOLT is not able to outperform the commercial solver when applied to the majority of the small instances with 145 time periods. This is expected as the commercial solver obtains relatively tight gaps for these instances. That being said, for these instances, SMOLT identifies solutions with an objective value close to the commercial solver's solution in many cases, with four instances within 5%. SMOLT outperforms the commercial solver on all but one of the larger instances with 290 time periods. The commercial solver struggles to find high-quality or feasible solutions for these instances, and SMOLT can identify significantly better solutions in some cases. For example, for the instance **p-10-02-04-290-C**, SMOLT obtains an objective value of -2,564,049 while the commercial solver does not find a feasible solution. Furthermore, the gap obtained by the commercial solver on the instances with 290 time periods ranges between 26.5% and 457.0%, with an average of 101.1%. Note that this is the average gap for the instances for which the commercial solver manages to identify a feasible solution at all. SMOLT, on the other hand, is able to obtain a feasible solution for all instances with 290 time periods, and achieves a gap lower than 61.8% for all instances, with an average of 44.3%. This illustrates how well SMOLT scales compared to the commercial solver while still being capable of obtaining high-quality solutions.

Next, Table 8.5 also compares the solution time of the commercial solver and SMOLT. For most instances, SMOLT identifies its best-found solution long before the time limit of 7,200 seconds. Even for the instances where SMOLT does not beat the commercial solver, the solutions are found quicker, and in many cases only a fraction of the time limit of 7,200 seconds is needed. Also, for the instances where SMOLT beats the commercial solver, SMOLT finds a better solution than the commercial solver relatively quickly. For example, for the instances **p-05-02-04-290 B** and **C**, SMOLT beats the commercial solver in only eight seconds. On average, the beat time is 699 seconds, which is only 9.7% of the time limit of 7,200 seconds for the commercial solver.

8.2.2 Medium Instances

Table 8.6 presents SMOLT's results when applied to the medium-sized Mowi instances in Table 7.2. SMOLT was configured to terminate if no improvement was made in 30 minutes or if the time limit of three hours (10,800 seconds) was reached. The table is structured in three sections. As mentioned in Section 6.4, the initial solutions are empty, meaning that the vessels stay at their origin for the entire planning horizon. The first section presents the properties of the initial empty solution. The second section presents the final solution obtained by SMOLT before terminating. The two first sections consist of the columns "Lowest objective," "Spot [abs]," and "Spot share [%]". The two first columns of these sections indicate the objective value and the spot market cost associated with the solution, respectively. Note that the objective value can become positive in cases where the total costs, which includes spot costs and travel costs, exceed the total revenue from delivery. The last column, "Spot share [%]" indicates the spot costs' share of the solutions' total costs, i.e., the transportation and spot market costs. Lastly, the third section presents the differences between the two first sections.

As explained in Section 7.3.1, these instances have been generated by the authors. The problems are complex, and it is not guaranteed that it is possible to serve all demands at the farms with the available vessels. The spot market cost cannot be reduced to zero in such cases. Furthermore, due to the large instance sizes, the commercial solver will not identify any valuable solutions, making a comparison to the commercial solver's results worthless. Hence, it is difficult to assess the quality of SMOLT's solutions. However, the results in Table 8.6 allow us to draw some conclusions.

First, SMOLT can identify solutions with low spot market usage for all instances, and in several cases, the solutions do not rely on spot market deliveries at all. On average, the spot market cost only constitutes 1.8% of the total costs in the best solutions found by SMOLT. Furthermore, SMOLT improves the solution quality significantly between the first and last reported solution. On average, the objective is reduced by 728.5%. However, for some instances, the improvement is notably better. Lastly, as these instances are generated to imitate instances faced in day-to-day operations, the solution times must reflect this. All the solutions were found within a time limit of three hours, which in our opinion, is sufficient to be used in daily operations.

Table 8.6: SMOLT's performance on the medium sized Mowi instances.

	Empty solution			**Final solution**			**Difference**	
	Lowest objective	*Spot* [abs]	*Spot share* [%]	*Lowest objective*	*Spot* [abs]	*Spot share* [%]	*Objective* [%]/[abs]	*Spot share* [p.p.]
p-20-06-01-145-A	-867,966	1,826	100.0	-2,949,299	0	0.1	239/-2,081,333	-99.9
p-20-06-01-145-B	-458,270	9,017	100.0	-2,623,940	706	0.3	472/-2,165,669	-99.7
p-20-06-01-145-C	-1,340,163	55,704	100.0	-3,056,298	2,381	0.5	128/-1,716,134	-99.5
p-20-06-03-145-A	-700,676	5,605	100.0	-8,304,951	0	0.0	1,085/-7,604,275	-100.0
p-20-06-03-145-B	-2,286,464	18,365	100.0	-7,896,486	658	0.1	245/-5,610,022	-99.9
p-20-06-03-145-C	-254,513	237,830	100.0	-8,848,034	5,586	0.7	3,376/-8,593,521	-99.3
p-20-06-06-145-A	-1,463,838	8,016	100.0	-11,645,454	0	0.0	695/-10,181,616	-100.0
p-20-06-06-145-B	-2,012,943	47,543	100.0	-11,295,754	4,131	0.3	461/-9,282,810	-99.7
p-20-06-06-145-C	-857,924	281,489	100.0	-12,566,054	9,209	0.7	1,364/-11,708,129	-99.3
p-20-06-01-235-A	-170,993	184,478	100.0	-3,427,878	3,417	0.6	1,904/-3,256,885	-99.4
p-20-06-01-235-B	-420	147,269	100.0	-3,301,432	0	0.0	785,259/-3,301,011	-100.0
p-20-06-01-235-C	-846,474	225,171	100.0	-3,207,345	28,332	7.3	278/-2,360,870	-92.7
p-20-06-03-235-A	-1,881,910	473,052	100.0	-6,583,439	7,912	0.5	249/-4,701,529	-99.5
p-20-06-03-235-B	-436,280	435,348	100.0	-8,747,953	13,057	0.9	1,905/-8,311,672	-99.1
p-20-06-03-235-C	-118,062	690,373	100.0	-7,917,898	45,887	1.0	6,606/-7,799,836	-99.0
p-20-06-06-235-A	-1,248,134	704,728	100.0	-9,781,509	8,814	0.4	683/-8,533,375	-99.6
p-20-06-06-235-B	-2,496,304	504,013	100.0	-9,334,559	11,003	0.6	273/-6,838,254	-99.4
p-20-06-06-235-C	-1,937,649	817,591	100.0	-9,243,540	118,480	2.1	377/-7,305,890	-97.9
p-20-06-01-290-A	-408,211	295,822	100.0	-3,705,772	13,358	1.7	807/-3,297,560	-98.3
p-20-06-01-290-B	-1,404,903	207,634	100.0	-3,611,075	2,951	0.5	157/-2,206,171	-99.5
p-20-06-01-290-C	-174,604	352,445	100.0	-4,190,309	11,472	1.6	2,299/-4,015,704	-98.4
p-20-06-03-290-A	-2,348,811	678,722	100.0	-10,103,431	42,188	2.1	330/-7,754,619	-97.9
p-20-06-03-290-B	-256,331	718,591	100.0	-10,050,603	21,431	1.3	3,820/-9,794,272	-98.7
p-20-06-03-290-C	-3,602,836	684,780	100.0	-11,609,474	46,332	3.5	222/-8,006,637	-96.5
p-20-06-06-290-A	-940,586	1,196,023	100.0	-12,164,139	56,628	2.1	1,193/-11,223,553	-97.9
p-20-06-06-290-B	-535,836	1,083,599	100.0	-12,273,768	106,899	5.0	2,190/-11,737,932	-95.0
p-20-06-06-290-C	-1,316,728	1,147,057	100.0	-12,305,810	145,961	8.1	834/-10,989,082	-91.9
p-40-07-01-145-A	-944,546	3,613	100.0	-6,992,961	0	0.0	640/-6,048,415	-100.0
p-40-07-01-145-B	22,476	22,476	100.0	-6,703,296	0	0.0	-29,924/-6,725,772	-100.0
p-40-07-01-145-C	-768,749	113,840	100.0	-7,155,846	10,738	1.3	830/-6,387,097	-98.7
p-40-07-03-145-A	-2,561,077	9,180	100.0	-14,649,650	29	0.0	472/-12,088,573	-100.0
p-40-07-03-145-B	-1,508,689	75,121	100.0	-14,270,288	0	0.0	845/-12,761,599	-100.0
p-40-07-03-145-C	-2,679,997	288,756	100.0	-15,205,376	3,868	0.2	467/-12,525,379	-99.8
p-40-07-06-145-A	-2,476,389	12,673	100.0	-17,478,918	1,970	0.0	605/-15,002,528	-100.0
p-40-07-06-145-B	-1,509,477	91,801	100.0	-14,272,729	15,203	0.9	845/-12,763,252	-99.1
p-40-07-06-145-C	-1,895,721	378,396	100.0	-17,611,000	47,808	3.1	829/-15,715,278	-96.9
p-40-07-01-235-A	103,721	356,138	100.0	-7,549,749	21,477	1.8	-7,378/-7,653,470	-98.2
p-40-07-01-235-B	163,534	285,761	100.0	-7,147,930	17,897	1.3	-4,470/-7,311,464	-98.7
p-40-07-01-235-C	-102,317	399,806	100.0	-7,847,099	7,586	0.7	7,569/-7,744,782	-99.3
p-40-07-03-235-A	-57,351	868,908	100.0	-10,363,539	54,549	2.8	17,970/-10,306,187	-97.2
p-40-07-03-235-B	-895,264	712,850	100.0	-13,635,518	30,253	1.3	1,423/-12,740,253	-98.7
p-40-07-03-235-C	-1,712,316	928,012	100.0	-12,702,485	45,937	1.3	641/-10,990,169	-98.7
p-40-07-06-235-A	-2,451,637	1,180,497	100.0	-15,167,675	129,113	4.0	518/-12,716,037	-96.0
p-40-07-06-235-B	-1,110,399	1,071,210	100.0	-14,155,860	5,911	0.2	1,174/-13,045,461	-99.8
p-40-07-06-235-C	-2,260,937	1,322,148	100.0	-13,054,632	183,623	1.9	477/-10,793,694	-98.1
p-40-07-01-290-A	-1,019,132	523,222	100.0	-7,933,753	26,968	2.0	678/-6,914,621	-98.0
p-40-07-01-290-B	192,402	486,903	100.0	-7,926,901	513	0.0	-4,220/-8,119,304	-100.0
p-40-07-01-290-C	451,234	625,954	100.0	-8,028,173	49,883	3.9	-1,879/-8,479,407	-96.1
p-40-07-03-290-A	-1,385,150	1,304,394	100.0	-13,196,289	50,588	1.2	852/-11,811,139	-98.8
p-40-07-03-290-B	-1,593,087	1,180,381	100.0	-13,675,277	94,534	3.6	758/-12,082,190	-96.4
p-40-07-03-290-C	209,301	1,558,782	100.0	-11,570,204	115,416	1.3	-5,628/-11,779,506	-98.7
p-40-07-06-290-A	-2,051,844	1,790,781	100.0	-12,568,613	252,915	5.5	512/-10,516,768	-94.5
p-40-07-06-290-B	-3,124,048	1,675,381	100.0	-13,561,439	293,010	10.1	334/-10,437,390	-89.9
p-40-07-06-290-C	-2,050,770	1,996,785	100.0	-9,996,783	460,311	8.3	387/-7,946,012	-91.7
Average	**-1,173,779**	**564,368**	**100.0**	**-9,725,337**	**48,646**	**1.8**	**728/-8,551,558**	**-98.2**

8.2.3 Large Instances

Table 8.7 presents SMOLT's results when applied to the large-sized Mowi instances in Table 7.2. The table is structured exactly as Table 8.6 presenting the results for the medium-sized instances.

Table 8.7: SMOLT's performance on the large Mowi instances.

	Empty solution			**Final solution**			**Difference**	
	Lowest objective	*Spot* [abs]	*Spot share* [%]	*Lowest objective*	*Spot* [abs]	*Spot share* [%]	*Objective* [%]/[abs]	*Relative spot* [p.p.]
p-60-10-01-145-A	-1,001,852	5,933	100.0	-10,308,692	0	0.0	929/-9,306,840	-100.0
p-60-10-01-145-B	-2,463,023	29,183	100.0	-9,833,191	164	0.0	299/-7,370,168	-100.0
p-60-10-01-145-C	-1,019,859	228,721	100.0	-10,679,150	20,170	1.6	947/-9,659,291	-98.4
p-60-10-03-145-A	-2,938,398	13,440	100.0	-15,369,493	2,964	0.2	423/-12,431,095	-99.8
p-60-10-03-145-B	-741,601	123,298	100.0	-14,062,316	34,567	2.3	1,796/-13,320,714	-97.7
p-60-10-03-145-C	-2,778,186	488,702	100.0	-16,223,425	37,932	2.3	484/-13,445,239	-97.7
p-60-10-06-145-A	-4,389,866	19,726	100.0	-14,820,988	6,593	0.5	237/-10,431,121	-99.5
p-60-10-06-145-B	-4,103,549	176,359	100.0	-13,433,175	86,341	8.4	227/-9,329,626	-91.6
p-60-10-06-145-C	-3,361,969	616,487	100.0	-14,887,170	133,973	6.3	342/-11,525,200	-93.7
p-60-10-01-235-A	-1,465,306	491,791	100.0	-12,180,716	18,212	1.2	731/-10,715,409	-98.8
p-60-10-01-235-B	-1,061,062	466,182	100.0	-11,546,296	17,282	0.9	988/-10,485,234	-99.1
p-60-10-01-235-C	7,306	717,815	100.0	-11,831,667	53,161	2.3	-162,035/-11,838,974	-97.7
p-60-10-03-235-A	-2,269,871	1,248,192	100.0	-17,569,626	148,956	4.6	674/-15,299,755	-95.4
p-60-10-03-235-B	-1,908,493	1,061,286	100.0	-19,003,324	56,126	1.5	895/-17,094,831	-98.5
p-60-10-03-235-C	-4,502,252	1,400,993	100.0	-17,048,103	77,047	2.5	278/-12,545,851	-97.5
p-60-10-06-235-A	-2,077,641	1,838,086	100.0	-15,169,843	131,026	2.8	630/-13,092,202	-97.2
p-60-10-06-235-B	-2,732,559	1,532,091	100.0	-17,386,653	149,560	3.4	536/-14,654,093	-96.6
p-60-10-06-235-C	-2,562,763	2,118,907	100.0	-13,595,136	522,198	9.5	430/-11,032,373	-90.5
p-60-10-01-290-A	-444,164	882,392	100.0	-12,478,367	15,868	0.8	2,709/-12,034,203	-99.2
p-60-10-01-290-B	-473,931	771,359	100.0	-11,904,770	23,445	1.1	2,411/-11,430,838	-98.9
p-60-10-01-290-C	-258,109	1,082,034	100.0	-13,064,114	37,411	1.7	4,961/-12,806,004	-98.3
p-60-10-03-290-A	-1,383,227	1,949,387	100.0	-14,313,592	362,031	10.9	934/-12,930,365	-89.1
p-60-10-03-290-B	-927,736	1,778,338	100.0	-15,043,419	240,971	4.9	1,521/-14,115,682	-95.1
p-60-10-03-290-C	-67,874	2,346,350	100.0	-13,454,242	314,682	7.7	19,722/-13,386,367	-92.3
p-60-10-06-290-A	-738,220	2,876,955	100.0	-11,131,646	1,956,420	76.9	1,407/-10,393,425	-23.1
p-60-10-06-290-B	-2,916,268	2,657,772	100.0	-23,673,276	350,809	9.9	711/-20,757,007	-90.1
p-60-10-06-290-C	-2,846,091	3,137,522	100.0	-28,333,263	668,935	17.1	895/-25,487,172	-82.9
p-80-10-01-145-A	-1,527,907	8,479	100.0	-10,991,609	1,186	0.1	619/-9,463,701	-99.9
p-80-10-01-145-B	-572,018	73,478	100.0	-12,152,474	9,892	0.7	2,024/-11,580,456	-99.3
p-80-10-01-145-C	-1,203,740	277,734	100.0	-12,580,032	32,568	2.1	945/-11,376,292	-97.9
p-80-10-03-145-A	-2,550,553	23,208	100.0	-24,799,228	700	0.0	872/-22,248,674	-100.0
p-80-10-03-145-B	-1,773,978	240,523	100.0	-20,899,833	24,095	0.7	1,078/-19,125,855	-99.3
p-80-10-03-145-C	-2,417,985	795,760	100.0	-22,986,755	57,133	1.8	850/-20,568,769	-98.2
p-80-10-06-145-A	-2,660,077	32,059	100.0	-30,245,825	1,693	0.0	1,037/-27,585,747	-100.0
p-80-10-06-145-B	-2,811,277	312,691	100.0	-19,147,983	118,898	2.9	581/-16,336,706	-97.1
p-80-10-06-145-C	-1,206,452	950,584	100.0	-21,025,114	130,698	2.0	1,642/-19,818,662	-98.0
p-80-10-01-235-A	-866,093	764,902	100.0	-15,203,629	25,807	1.1	1,655/-14,337,536	-98.9
p-80-10-01-235-B	-2,126,892	588,726	100.0	-14,193,634	29,831	1.3	567/-12,066,742	-98.7
p-80-10-01-235-C	54,983	1,013,630	100.0	-15,179,048	83,633	4.2	-27,706/-15,234,032	-95.8
p-80-10-03-235-A	-2,164,141	2,046,267	100.0	-15,141,610	382,642	9.5	599/-12,977,469	-90.5
p-80-10-03-235-B	-939,121	1,920,029	100.0	-23,984,662	454,780	15.3	2,453/-23,045,541	-84.7
p-80-10-03-235-C	695,278	2,630,246	100.0	-15,373,946	641,971	13.1	-2,311/-16,069,224	-86.9
p-80-10-06-235-A	-3,081,728	2,729,840	100.0	-32,236,212	572,558	13.3	946/-29,154,484	-86.7
p-80-10-06-235-B	-2,160,621	2,543,514	100.0	-29,841,713	467,990	9.1	1,281/-27,681,091	-90.9
p-80-10-06-235-C	-1,956,674	3,278,714	100.0	-10,800,378	1,507,733	25.8	452/-8,843,704	-74.2
p-80-10-01-290-A	-597,543	1,240,904	100.0	-15,252,998	124,980	4.5	2,452/-14,655,454	-95.5
p-80-10-01-290-B	181,703	1,199,162	100.0	-12,777,616	174,968	8.3	-7,132/-12,959,320	-91.7
p-80-10-01-290-C	-422,194	1,455,015	100.0	-15,051,979	124,858	5.3	3,465/-14,629,785	-94.7
p-80-10-03-290-A	-598,078	3,334,064	100.0	-23,092,793	591,393	9.8	3,761/-22,494,715	-90.2
p-80-10-03-290-B	-1,461,812	3,087,296	100.0	-16,804,475	795,334	19.7	1,049/-15,342,663	-80.3
p-80-10-03-290-C	-337,863	3,799,989	100.0	-15,910,441	1,082,866	20.8	4,609/-15,572,577	-79.2
p-80-10-06-290-A	839,104	4,481,229	100.0	-12,196,978	2,965,559	80.4	-1,553/-13,036,082	-19.6
p-80-10-06-290-B	-2,065,149	4,067,761	100.0	-8,148,628	3,010,046	75.6	294/-6,083,479	-24.4
p-80-10-06-290-C	1,517,366	5,103,132	100.0	-14,792,324	3,091,120	65.8	-1,074/-16,309,690	-34.2
Average	**-1,548,889**	**1,445,523**	**100.0**	**-16,280,696**	**406,885**	**10.6**	**951/-14,731,806**	**-89.4**

Like the medium-sized instances, we observe a significant improvement from the initial empty and the final solution. The average spot market dependency is 10.6%, which is higher than for the medium instances. However, this is largely due to a few instances with large spot dependency in the final solution, such as **p-80-10-06-290-A**. Most of the large instances have a spot share similar in size to that of the medium instances. The instances contributing to the higher spot share average are especially those with 6 products, 290 time periods, and 60 and 80 nodes. The complexity of these instances is beyond any instances previously solved in the literature, and it is therefore not surprising that SMOLT's obtained solutions have a relatively high spot share. Still, for instances such as **p-80-10-01-290-C** and **p-80-10-01-235-C** solutions with a low spot share of 5.3% and 4.2%, respectively, are obtained. Even though these instances are not the largest, they are larger than typical MIRP instances found in the literature. Therefore, obtaining solutions with such low spot shares is considered promising.

8.3 MIRPLib Instances

This section presents the computational results of the test instances from the MIRPLib benchmark library. As mentioned in Section 7.2, there are some differences between the FFMIRP and the MIRPLib instances. To recall, the vessels must arrive at production ports empty and leave them fully loaded, and there is a semi-continuous requirement on the quantity being loaded and unloaded. As argued in Section 7.2, in reality, these restrictions are not necessary. The results from the Group 1 instances are therefore presented in two ways. Section 8.3.1 presents the Group 1 results with all requirements of MIRPLib respected, while Section 8.3.2 presents the results generated when the restrictions mentioned above are relaxed, referred to as the relaxed Group 1 instances. The results are presented in this manner for two reasons. Firstly, when comparing SMOLT's performance to the best-known solutions, all restrictions must be respected for a fair comparison. However, as mentioned in Section 7.2, relaxation of these constraints might allow us to construct more efficient routes. For MIP-based approaches, these additional constraints reduce the size of the search space and will typically allow a solver to more efficiently tackle the problem. Relaxing these constraints and presenting the results shows that SMOLT can still solve the problem instances efficiently and, in some cases, identify better solutions compared to the non-relaxed instances. Lastly, in Section 8.3.3 the results from the Group 1 instances with longer planning horizons than what has been solved previously by other approaches are presented.

A more exhaustive presentation of these results, including more elaborate visualizations, are available at `https://fiskeforgutane.github.io`.

8.3.1 Group 1 Instances

Table 8.8 presents the results from solving the MIRPLib Group 1 instances while respecting all restrictions. The leftmost column indicates the problem instance, as described in Table 7.3. The rest of the table is separated into two sections - the results from a planning horizon of 45 and 60 time periods, respectively. The column "SMOLT" indicates the objective value obtained by SMOLT, and "BKS" indicates the best-known solution. The column "Gap [%]" shows the relative difference between SMOLT's and the best-known solution's objective value, i.e., $1 - \frac{SMOLT}{BKS}$. The columns "Time" present the solution times for SMOLT and the current BKS, respectively.

In order to make these results comparable to earlier attempts at solving MIRPLib Group 1 instances, all runs were done using a single island running on a single thread. Do note that this represents a worst-case scenario for SMOLT's performance. Islanding plays an important role in maintaining a diverse set of candidate solutions. By effectively removing islanding, we drastically reduce the amount of diversity, increasing the risk of premature convergence to a suboptimal local minimum. In addition to this, we run SMOLT without restricting variables to be semi-continuous. These restrictions are instead added after SMOLT has converged to a solution. A consequence is that SMOLT might be able to find a solution that is valid without the semi-continuous restrictions, but invalid or of poorer quality when they are imposed. All of the results presented in Table 8.8

respect the semi-continuous constraints.

Table 8.8: SMOLT's performance on the MIRPLib Group 1 instances with 45 and 60 time periods. The dashed values indicate that SMOLT was unable to find a feasible solution within the time limit. The current best-known solutions were obtained by Papageorgiou et al. (2014b)[a], Friske and Buriol (2017)[b], Friske and Buriol (2018)[c], Friske and Buriol (2020)[d], and Friske et al. (2022)[e].

	$\|\mathcal{T}\| = 45$					$\|\mathcal{T}\| = 60$				
	SMOLT	*Time*	*BKS*	*Time*	*Gap [%]*	*SMOLT*	*Time*	*BKS*	*Time*	*Gap [%]*
LR1-1-DR1-3-VC1-V7a	-13,254	7,313	-13,272[a]	177	0.1	-16,655	3,531	-16,675[d]	444	0.1
LR1-1-DR1-4-VC3-V11a	-10,947	2,741	-11,243[b]	1,578	2.6	-12,841	5,686	-13,257[a]	7,139	3.1
LR1-1-DR1-4-VC3-V12a	-10,737	3,622	-10,766[e]	10,339	0.3	-	-	-11,040[a]	7,731	-
LR1-1-DR1-4-VC3-V12b	-9,047	2,354	-9,085[b]	1,942	0.4	-9,881	6,900	-10,053[a]	8,710	1.7
LR1-1-DR1-4-VC3-V8a	-5,025	2,849	-5,106[a]	4,609	1.6	-4,571	6,726	-5,191[a]	7,687	11.9
LR1-1-DR1-4-VC3-V9a	-6,645	2,131	-6,921[c]	1,220	4.0	-7,366	5,533	-7,552[a]	8,285	2.5
LR1-2-DR1-3-VC2-V6a	-9,902	3,072	-11,134[a]	5,963	11.1	-12,701	9,922	-13,532[e]	8,300	6.1
LR1-2-DR1-3-VC3-V8a	-11,878	2,481	-12,010[a]	7,634	1.1	-14,174	7,202	-14,652[e]	8,031	3.3
LR2-11-DR2-22-VC3-V6a	-9,561	3,772	-9,718[a]	8,149	1.6	-12,692	4,922	-12,745[a]	8,404	0.4
LR2-11-DR2-33-VC4-V11a	-	-	-14,017[a]	8,913	-	-	-	-15,387[a]	8,943	-
LR2-11-DR2-33-VC5-V12a	-	-	-18,524[e]	8,934	-	-	-	-22,948[c]	6,335	-
LR2-22-DR2-22-VC3-V10a	-	-	-24,985[c]	6,279	-	-	-	-32,627[a]	8,803	-
LR2-22-DR3-333-VC4-V14a	-	-	-21,952[a]	9,406	-	-	-	-26,873[a]	9,542	-
LR2-22-DR3-333-VC4-V17a	-	-	-22,294[a]	10,793	-	-	-	-27,000[a]	9,615	-

The results in Table 8.8 show that SMOLT is not able to obtain a feasible solution for all Group 1 instances. However, it solves 17 of the 28 instances and is within 3.1% of the BKS on average. Furthermore, when comparing the solution times, one observes significant differences between SMOLT and the BKS. For some instances, as **LR1-1-DR1-3-VC1-V7a**, SMOLT spends significantly more time than the method used for obtaining the current BKS. However, for other instances, as **LR1-2-DR1-3-VC3-V8a**, SMOLT's solution is within 1.1% of the BKS in less than 50% of the time used by the BKS method. Note that the criterion used for gradually expanding the horizon prevents SMOLT from ever converging to a solution in less than about half an hour for $|\mathcal{T}| = 45$. Changing the criterion or reducing the use of the gradually expanding horizon decomposition could lead to much faster convergence for some problems. However, this might come at the expense of solution quality.

However, as Table 8.8 shows, SMOLT is not able to solve 11 of the 28 instances. This is believed to be caused by two main reasons. Firstly, SMOLT is not designed to handle the additional constraints of the MIRPLib instances as they are irrelevant for FFMIRPs, while also negatively impacting the efficiency of SMOLT, as described in Appendix A.5.2. These constraints include that all vessels must leave production sites empty and arrive at them fully loaded and a semi-continuous constraint on the unloaded amounts. In fact, if the semi-continuous constraints, which represent an artificial restriction on the problem intended to simplify it, are relaxed, SMOLT obtains valid solutions for all instances. Secondly, the instances with no valid solution are the most challenging ones with several loading and discharging regions. As mentioned above, running on only one thread effectively removes islanding, and this is believed to impact the solution quality for these instances the most, as their complexity increases the risk of SMOLT converging to a suboptimal solution.

8.3.2 Relaxed Group 1 Instances

Table 8.9 presents SMOLT's results on the MIRPLib Group 1 instances with the semi-continuous loading and unloading restrictions and full capacity exploitation disabled. Furthermore, these results are generated by running 24 islands in parallel. These conditions are better suited for SMOLT, which is reflected in the results. The table is structured as Table 8.8, except that the solution times are not reported because when using several threads in parallel, the comparison to the BKS solution times is unfair. The results clearly show that SMOLT has no problem solving the relaxed instances, while it is considered more challenging for MIP-based approaches. This is an advantage as it enables SMOLT to find better solutions than those found when respecting all

restrictions. In fact, SMOLT identifies solutions that beat the current best-known solutions on 14 instances, with a 7.7% improvement on instance **LR2-11-DR2-33-VC4-V11a** with 60 time periods. This clearly illustrates the advantage of applying solution methods that do not require the introduction of artificial constraints in order to work effectively.

Table 8.9: SMOLT's performance on the relaxed Group 1 instances with 45 and 60 time periods. The bold values indicate new best-known solutions. The current best-known solutions were obtained by Papageorgiou et al. (2014b)[a], Friske and Buriol (2017)[b], Friske and Buriol (2018)[c], Friske and Buriol (2020)[d], and Friske et al. (2022)[e].

	$\|\mathcal{T}\| = 45$			$\|\mathcal{T}\| = 60$		
	SMOLT	*BKS*	*Gap [%]*	*SMOLT*	*BKS*	*Gap [%]*
LR1-1-DR1-3-VC1-V7a	-14,143	-13,272[a]	**-6.6**	-17,403	-16,675[a]	**-4.4**
LR1-1-DR1-4-VC3-V11a	-11,277	-11,243[b]	**-0.3**	-12,868	-13,257[a]	2.9
LR1-1-DR1-4-VC3-V12a	-10,752	-10,766[e]	0.1	-11,243	-11,040[d]	**-1.8**
LR1-1-DR1-4-VC3-V12b	-9,350	-9,085[b]	**-2.9**	-9,986	-10,053[a]	0.7
LR1-1-DR1-4-VC3-V8a	-5,112	-5,106[a]	**-0.1**	-5,271	-5,191[a]	**-1.5**
LR1-1-DR1-4-VC3-V9a	-6,707	-6,921[c]	3.1	-7,445	-7,552[a]	1.4
LR1-2-DR1-3-VC2-V6a	-11,142	-11,134[a]	**-0.1**	-13,632	-13,532[a]	**-0.7**
LR1-2-DR1-3-VC3-V8a	-11,981	-12,010[a]	0.2	-14,382	-14,652[a]	1.8
LR2-11-DR2-22-VC3-V6a	-9,922	-9,718[a]	**-2.1**	-12,927	-12,745[e]	**-1.4**
LR2-11-DR2-33-VC4-V11a	-13,907	-14,017[a]	0.8	-16,577	-15,387[a]	**-7.7**
LR2-11-DR2-33-VC5-V12a	-18,338	-18,524[e]	1.0	-22,452	-22,948[c]	2.2
LR2-22-DR2-22-VC3-V10a	-25,041	-24,985[e]	**-0.2**	-32,322	-32,627[a]	0.9
LR2-22-DR3-333-VC4-V14a	-22,176	-21,952[a]	**-1.0**	-25,966	-26,873[a]	3.4
LR2-22-DR3-333-VC4-V17a	-22,030	-22,294[c]	1.2	-23,932	-27,000[a]	11.4

8.3.3 Relaxed Group 1 Instances with Long Planning Horizons

Lastly, we present SMOLT's performance on the relaxed Group 1 instances with planning horizons of 90 and 120 time periods. As far as we know, no solutions for the original Group 1 instances have been reported previously for $|\mathcal{T}|> 60$. Despite relaxing the instances, SMOLT is, to the best of our knowledge, the first solution method attempting to solve any variant of the Group 1 instances with a longer planning horizon than 60 time periods. The table is structured as Table 8.9; however, instead of reporting the gap to the BKS, which in this case is SMOLT's solution, we also report the inventory violations. The violation is calculated as the cumulative inventory shortage (or overflow for production ports) throughout the planning horizon, i.e., a shortage of 10, 20, and 30 in time periods 1, 2, and 3, accumulates to a violation of 60. To understand whether the reported violation is large or small, we also report the violation as the share of the experienced violation if no products could be delivered neither from the vessels nor the spot market. This is what the column "Violation gap [%]" reports. The results colored in light gray are instances where SMOLT did not identify a solution with no violation.

Table 8.10: SMOLT's performance on the relaxed Group 1 instances with 90 and 120 time periods.

	$\|\mathcal{T}\| = 90$				$\|\mathcal{T}\| = 120$			
	SMOLT	*Violation*	*Violation gap* [%]	*BKS*	*SMOLT*	*Violation*	*Violation gap* [%]	*BKS*
LR1-1-DR1-3-VC1-V7a	-24,037	0.0	0.0	-24,037	-30,591	0.0	0.0	-30,591
LR1-1-DR1-4-VC3-V11a	-15,611	0.0	0.0	-15,611	-19,257	0.0	0.0	-19,257
LR1-1-DR1-4-VC3-V12a	-14,298	0.0	0.0	-14,298	-16,904	0.0	0.0	-16,904
LR1-1-DR1-4-VC3-V12b	-11,044	0.0	0.0	-11,044	-12,540	0.0	0.0	-12,540
LR1-1-DR1-4-VC3-V8a	-3,996	684.0	0.1	-3,996	-4,711	6921.0	0.5	-4,711
LR1-1-DR1-4-VC3-V9a	-8,235	0.0	0.0	-8,235	-8,886	0.0	0.0	-8,886
LR1-2-DR1-3-VC2-V6a	-18,292	0.0	0.0	-18,292	-23,165	0.0	0.0	-23,165
LR1-2-DR1-3-VC3-V8a	-18,864	0.0	0.0	-18,864	-23,618	0.0	0.0	-23,618
LR2-11-DR2-22-VC3-V6a	-16,904	75.0	0.0	-16,904	-21,231	891.0	0.1	-21,231
LR2-11-DR2-33-VC4-V11a	-18,459	0.0	0.0	-18,459	-23,621	3.0	0.0	-23,621
LR2-11-DR2-33-VC5-V12a	-27,047	22.0	0.0	-27,047	-34,180	333.0	0.0	-34,180
LR2-22-DR2-22-VC3-V10a	-39,963	0.0	0.0	-39,963	-55,449	0.0	0.0	-55,449
LR2-22-DR3-333-VC4-V14a	-29,366	71.0	0.0	-29,366	-36,287	779.0	0.0	-36,287
LR2-22-DR3-333-VC4-V17a	-18,140	56.0	0.0	-18,140	-23,204	830.0	0.0	-23,204

SMOLT can find a feasible solution with no violation in nine instances with a planning horizon of 90 time periods and eigth in the ones with a planning horizon of 120 time periods. For the instances where SMOLT cannot find a solution with no violation, the violation gap is minimal, meaning that almost all demand is covered. In all cases except one, SMOLT is able to identify a solution with a violation gap of less than 0.1%. The one case with a larger gap has a gap of 0.5%. Note that as no solutions have been reported for these instances previously, it is not given that a feasible solution exists for all of them.

8.4 Managerial Insights

The forecast for the Norwegian salmon farming industry is clear – according to Norsk Industri (2017), the Norwegian government has the ambition to increase the size of the salmon farming industry by a factor of five in the next three decades. Consequently, the actors in the industry will have to invest significantly in new infrastructure to keep up with the demand. These investments will include both new farms, vessels, and factories. While the primary purpose of SMOLT is to provide Mowi with a tool for making operational decisions on a day-to-day basis, it can also be used for decision support when long-term strategic decisions are being made.

This section presents some proposals on how SMOLT can be used to gain managerial insights – in particular, how Mowi can use SMOLT as a tool when deciding on future investments in infrastructure and how it can be used as a valuation tool for concessions. First, Section 8.4.1 discusses how SMOLT can be used in deciding the locations of future factories and storage facilities. Then, Section 8.4.2 presents a proposed way of using SMOLT to decide on when to invest in new vessels and what type of vessel should be bought. Finally, Section 8.4.3 discusses how SMOLT can contribute to the valuation of salmon concessions.

8.4.1 Factory and Storage Locations

When deciding the location of a new factory or storage facility, many factors play a role. However, in situations where Mowi is left with several possible options, SMOLT can be used to support decisions quantitatively. By including the new infrastructure in the problem and setting several potential locations, SMOLT can solve the different instances representing the alternatives, allowing for comparison.

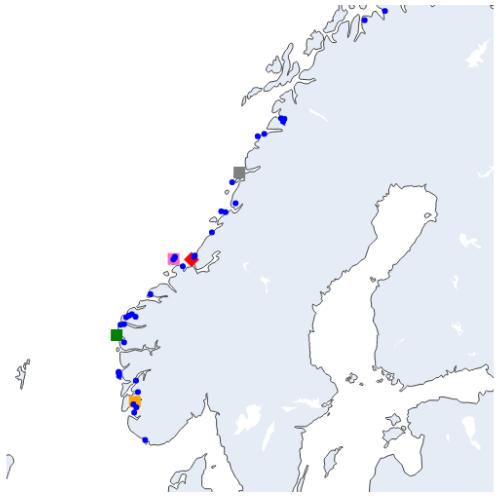


Figure 8.1: Scenario where four different new factory locations are analysed and compared. The red diamond represents the current factory location at Bjugn. The orange square represents *Location 1*, the green square represents *Location 2*, the pink square represents *Location 3*, and the gray square represents *Location 4*. Finally, the blue circles represent the farms.

Particularly when deciding on the locations of future factories, we believe such quantitative analyses can be valuable. Mowi currently only has one factory, and where future factories are located can play a major role in keeping the salmon feed part of the company profitable. Consider a scenario where Mowi has singled out four potential locations for their next factory, illustrated in Figure 8.1. We assume that a factory in each of the four locations has similar terms regarding operational factors, including the delivery of raw material, access to staff, and investment costs. In this scenario, SMOLT can be employed to compare the four locations. When comparing the locations, it is essential to consider different demand scenarios; one location might contribute to the lowest costs during a low-demand planning horizon, but another location becomes preferable as the demand increases. Hence, a simulation of different demand scenarios, all solved with SMOLT, can be applied to find the most robust location.

We conduct an analysis of the scenario described above on three different instances of medium size, more specifically on **p-40-07-01-145-C**, **p-40-07-03-145-C**, and **p-40-07-06-145-C**. For each instance, we generate four new instances where we replace one farm with a factory in each instance. The locations of the new factories correspond to the scenario illustrated in Figure 8.1. SMOLT is run three times for each instance, and the best objective is reported for each of them, along with the corresponding cost associated with purchases from the spot market. The results of the analysis are given in Table 8.11.

Table 8.11: Comparison of the objectives with added factories in different locations for a new factory using SMOLT.

	Location 1		**Location 2**		**Location 3**		**Location 4**	
	Objective	*Spot*	*Objective*	*Spot*	*Objective*	*Spot*	*Objective*	*Spot*
p-40-07-01-145-C	-7,140,892	2,248	-7,282,099	10,739	-6,986,010	14,714	-6,531,956	17,864
p-40-07-03-145-C	-16,025,255	12,876	-16,003,697	6,092	-15,206,808	13,318	-14,849,321	10,972
p-40-07-06-145-C	-24,322,843	34,174	-24,702,815	30,409	-24,155,659	35,118	-23,440,405	6,274
Average	**-15,829,663**	**16,433**	**-15,996,204**	**15,746**	**-15,449,492**	**21,050**	**-14,940,560**	**11,703**

Table 8.11 presents an analysis of how the objective is affected by adding the four proposed factory locations. From the analysis, we can see that **Location 2** is the most favorable location – it has the lowest average objective, and it finds the best solution in two of the three problem instances. **Location 1** is a close second and finds the best solution in the instance with three products. From an intuitive perspective, the results make sense – **Location 1** and **Location 2** are located south of the current factory at Bjugn in an area with high density of farms. Thus, better objectives can

be achieved by decreasing travel distances to the farms in the southern region with a new factory.

Further, we see that **Location 4** provides the lowest spot cost. This indicates that much of the spot dependency originates in the northern region in the other instances. The cost of serving the farms in the region is too high to be profitable. By adding a factory in the region, these farms can be served more cost-efficiently, and consequently, the spot dependency is reduced. However, the average objective is worse for **Location 4** than for the other three. This is a consequence of low total feed demand in the proximity of the added factory.

Similarly to the factory analysis above, SMOLT can be used in order to assess both existing and potential new storage locations[1]. We argue that analyses similar to the one discussed in this section are valuable for the management as the decisions to be made are of great importance, and the underlying information is complex to analyze manually. Without a simulation tool, assessing the impact of a structural change in the distribution network is challenging.

8.4.2 Vessel Investments

As the total feed demand increases, investments in new vessels will become necessary to satisfy the demand. Similarly, as with the factory locations, SMOLT can be used to simulate how an extra vessel in the fleet will affect the costs in different demand scenarios. This can be particularly interesting in high-demand scenarios, where Mowi is relying on external deliveries from the spot market today. For the people responsible for making investment decisions, these analyses can be used to find the right timing for introducing new vessels and how much capacity the new vessels should have.

More specifically, simulations can be run using forecasts of future production volumes to determine the degree of self-sufficiency given the current fleet. At some point, the costs of buying from external will exceed the costs associated with investing in a new vessel; by employing such analyses, Mowi can stay ahead of time and order new vessels well in advance.

8.4.3 Concession Bidding

The Norwegian salmon farming industry is regulated by concessions issued by the Ministry of Trade, Industry and Fisheries. The concessions are associated with a production region, and when new concessions are issued, a limited number becomes available in each region (Directorate of Fisheries, 2022). The concessions are sold at auctions, more specifically at simultaneous clock auctions. In a simultaneous clock auction, the price gradually increases until the demand equals the supply. Thus, reasonable valuations of the concessions are crucial for making good deals. These valuations depend on numerous factors – some factors are easy to calculate, while others are more intricate. One such intricate factor is how additional feed demand in a particular region affects the total distribution costs. In a region with a high density of farms, added demand is likely to affect the distribution costs less than in a region with fewer farms, and hence, a concession in the latter case should be valued lower.

When a concession is bought, Mowi has two options – the concession can be given to one farm in the region, or a new farm can be established. In the first case, SMOLT can be used to calculate the additional costs by simply adding feed demand at the particular farms corresponding to the increased production volume. The difference in distribution costs between the new demand scenario and the original one makes up for the additional distribution costs accrued by the concession. These additional distribution costs are likely to differ from region to region, and thus, quantifying them can contribute in valuating the concessions. Similarly, SMOLT can be used for evaluating new potential farm locations, if no existing farms have the capacity to include a new concession. In the following analysis, we assume that all farms have the capacity required.

[1] A storage location is here referring to the facilities used by Mowi for storing feed produced in periods with excess production rates. This is currently not implemented in SMOLT but is discussed in Chapter 10.

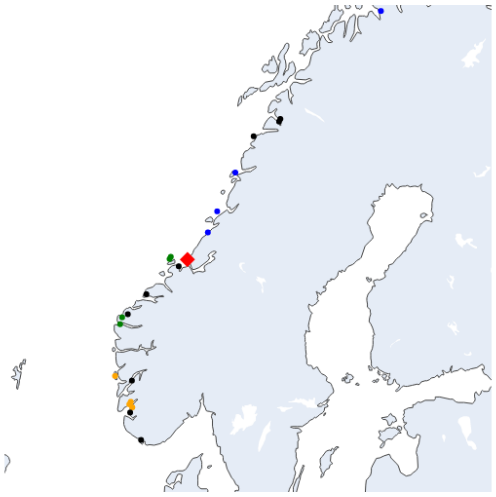


Figure 8.2: Scenario where four farms in each of the three regions are given extra consumption corresponding to a concession. The red diamond represents the current factory location at Bjugn. The orange circles represent the randomly chosen farms from **Region South**, the green circles the farms from **Region Middle**, and the blue circles the farms from **Region North**. Finally, the black circles represent the farms not chosen for additional consumption.

The Norwegian coastline is divided into 13 production regions, but we use a simplified representation to keep the analysis simple and easy to read. Instead of 13 regions, we have divided the coastline into three regions, equivalent to the regions presented in Section 7.3.1. The analysis is conducted using a single problem instance, namely **p-20-06-01-145-C**. In each region, four random farms are selected to be assigned additional feed consumption corresponding to that of a concession. An illustration of the problem is given in Figure 8.2. A new problem instance for each of the consumption scenarios is generated, resulting in a total of 12 problem instances. All 12 instances are run twice, and the lowest costs are reported. The results are given in Table 8.12.

Table 8.12: Analysis on the total costs when a concession is added in different regions.

	Region South		**Region Middle**		**Region North**	
	Total cost	*Spot*	*Total cost*	*Spot*	*Total cost*	*Spot*
Farm 1	382,888	257	364,140	257	353,736	2,262
Farm 2	467,051	257	369,877	257	374,205	5,001
Farm 3	482,553	257	382,274	2,797	399,848	350
Farm 4	503,932	3,008	391,060	257	434,516	3,474
Average	**459,106**	**945**	**376,838**	**892**	**390,576**	**2,772**

According to the results, a concession in **Region Middle** has the lowest total distribution cost. This makes sense as the factory is located at Bjugn in the region. Further, **Region South** has the highest marginal increase in total costs on average. As can be seen from Figure 8.2, three of the selected farms from the southern region are located relatively far away from the factory, making them expensive to serve. Surprisingly, **Region North** has an average cost almost as low as **Region Middle**. This is most likely due to three of the four selected farms, which constitute the three lowest costs, are located in proximity of the factory at Bjugn, as can be seen from Figure 8.2. Note that this analysis is simplified, and is intended to communicate how SMOLT *can* be used to conduct advanced analyses. The results are thus not a representation of how things are in the real-life scenario.

Chapter 9

Concluding Remarks

The salmon farming industry has been growing rapidly for the last decades and is expected to continue growing. As the industry becomes larger, the distribution problems faced by the actors become increasingly complex, and manual planning becomes intractable. Even though the industry as a whole is profitable, the margins in the salmon feed industry are low. Hence, optimized everyday operations are crucial. The complexity of manual planning and the low margins calls for automation of the logistics process. Today, the salmon farming company Mowi employs an order-based model for their feed distribution, influenced by manual decisions. As they have access to all relevant data for a vendor-managed inventory approach (VMI), this master's thesis focuses on an optimization algorithm that serves as a VMI planning tool. The VMI approach, in this case, a maritime inventory routing problem (MIRP), ensures that all fish farms maintain a stock level of all demanded feed types within user-specified limits. Thus, the manual aspects of the planning are removed in their entirety.

This thesis explores a particular type of the MIRP, namely the fish feed MIRP (FFMIRP) faced by Mowi. The problem at hand incorporates several complicating factors – Mowi has multiple unmixable products, the consumption rates vary over time, and the size of the distribution network is significantly larger than those explored by previous studies. We first propose an extension of the arc-flow model introduced by Song and Furman (2013) to incorporate all aspects relevant to the FFMIRP. The model uses a time-space network, where each node represents a combination of a port and a time period. A set of arcs between the nodes is generated for each vessel in the heterogeneous fleet based on vessel-specific properties. The problem is then solved as a flow problem over the arcs in the network, minimizing the distribution cost while delivering as much feed as possible throughout the planning horizon.

The FFMIRP is a complex problem, so commercial solvers cannot solve real-life instances using exact models. Hence, to solve the problem faced by Mowi, we have implemented a matheuristic – the Scalable Memetic Optimization algorithm with LP-based search Techniques (SMOLT). SMOLT exploits the structure of the problem by separating the routing from the quantity assignments. The routing is done entirely by a memetic algorithm (MA), a genetic algorithm augmented with local search techniques. On the other hand, a linear program (LP) is used for assigning quantities to all visits, given a routing solution from the MA.

The computational study indicates that the results generated by SMOLT on the instances based on Mowi data are promising. First, a comparison between SMOLT and a commercial solver on a set of small test instances is presented. As expected, the commercial solver outperforms SMOLT on the smallest instances. However, SMOLT obtains high-quality solutions, with four solutions within 5% of those of the commercial solver. Among the larger instances used to compare SMOLT to the commercial solver, SMOLT performs significantly better with an average gap of 44.3% compared to 101.1% for the commercial solver. Further, the results from the larger instances indicate that adequate solutions are found in a reasonable time, indicating that SMOLT scales well.

Finally, a computational study on the MIRPLib benchmark instances introduced by Papageorgiou

et al. (2014c) is given to further strengthen the indications that SMOLT can discover good solutions. Despite not being designed with MIRPLib Group 1 in mind, SMOLT is able to find high-quality solutions for most of these instances. In the fully constrained instances, including a semi-continuous constraint on the loading and unloading quantities, and a requirement to always leave production ports fully loaded and return to them empty, SMOLT struggles to find feasible solutions in some of the instances. However, in the instances it obtains feasible solutions, the average gap from the best-known solutions is only 3.1%. When these artificial constraints are relaxed, it consistently produces better solutions than previously presented in the literature. It finds 14 solutions with a better objective than the best-known solutions out of 28 MIRPLib Group 1 instances. Furthermore, it is able to find feasible or nearly-feasible solutions for all relaxed instances with a planning horizon of 90 and 120 time periods.

This master's thesis contributes to the research on the MIRP in two ways. An extended mathematical model is formulated and implemented in a commercial solver, incorporating a combination of aspects that has not been explored previously. Further, a novel matheuristic is developed and implemented to be scalable to large real-life instances.

Chapter 10

Future Work

This chapter discusses potential future improvements and extensions of the work presented in this master's thesis. The discussion is twofold – we first present an extension of SMOLT to incorporate further aspects from the real-life scenario faced by Mowi and propose a way to integrate the algorithm into day-to-day operations. Then, we discuss potential improvements to the current implementation.

During high-demand seasons, the production capacity of feed is a limiting factor for Mowi to be self-sufficient. Consequently, temporary storage facilities store feed produced in seasons with excess production. The current implementation of SMOLT does not incorporate this aspect; however, introducing hybrid ports can solve it. The hybrid ports have no consumption or production rates and can be loaded at and unloaded from. By associating a revenue for delivering to the ports, vessels will be incentivized to do so, and in periods with insufficient production, the vessels can load at the ports. Further, to allow SMOLT to have actual operational value for Mowi, it must be integrated into a holistic system. This would require continuously updated and complete data and integrating SMOLT in an application dedicated to operators.

When it comes to the current implementation of SMOLT, we have identified some ideas that could be worth exploring. As of today, the bottleneck in terms of run time is the LP that assigns quantities. Consequently, we have considered implementing a sparse version of the current LP, which would only construct the variables and constraints relevant to the specified routing solution. For example, we only need to define a vessel's load for the time periods when it is docked at a port since that is the only time the load may change. Every time something in a routing solution changes, the LP is solved, and a quantity is requested. If we can reduce the number of variables optimized, we would also reduce the run time of SMOLT.

Further, we have not considered any particular tailored decomposition approach suitable for the problem at hand. This could, for example, include clustering the ports similarly to what is proposed in the study by Papageorgiou et al. (2014b). The ports can then be aggregated into clusters; each cluster contains aggregated information regarding the ports in it. SMOLT's routing and quantity assignments can be applied to the aggregated sets to reduce the complexity. Finally, a dedicated algorithm can do the routing within the clusters, and quantity assignments to ports can be made.

The current parameters used are not necessarily well-tuned. We have conducted some easily accessible trial-and-error analyses on some of the most critical parameters, like the population size. However, conducting a more rigorous parameter tuning can improve the performance of SMOLT further and can thus be an idea for future work.

Finally, the current implementation of the Mowi problem does not consider the production side of the real-life problem. As discussed in the study by Brekkå et al. (2022), the production rates at the factory are not unlimited. Thus, the instances would be even more realistic by including this in the problem instances. As a matter of fact, the logic for dealing with production rates is already implemented in SMOLT, as this was necessary for the MIRPLib instances. The only thing required for it to work would be further data on production rates.

Bibliography

Agra, A., Andersson, H., Christiansen, M., and Wolsey, L. (2013). A maritime inventory routing problem: Discrete time formulations and valid inequalities. *Networks*, 62(4):297–314.

Agra, A., Christiansen, M., Delgado, A., and Hvattum, L. M. (2015). A maritime inventory routing problem with stochastic sailing and port times. *Computers Operations Research*, 61:18–30.

Agra, A., Christiansen, M., Delgado, A., and Simonetti, L. (2014). Hybrid heuristics for a short sea inventory routing problem. *European Journal of Operational Research*, 236(3):924–935.

Agra, A., Christiansen, M., Hvattum, L. M., and Rodrigues, F. (2018). Robust optimization for a maritime inventory routing problem. *Transportation Science*, 52(3):509–525.

Agra, A., Christiansen, M., Ivarsøy, K. S., Solhaug, I. E., and Tomasgard, A. (2017). Combined ship routing and inventory management in the salmon farming industry. *Annals of Operations Research*, 253(2):799–823.

Angilletta, Michael J., J., Steury, T. D., and Sears, M. W. (2004). Temperature, Growth Rate, and Body Size in Ectotherms: Fitting Pieces of a Life-History Puzzle1. *Integrative and Comparative Biology*, 44(6):498–509.

Austreng, E., Storebakken, T., and Åsgård, T. (1987). Growth rate estimates for cultured atlantic salmon and rainbow trout. *Aquaculture*, 60(2):157–160.

Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., Mack, R. G., and Prutzman, P. J. (1983). Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces*, 13(6):4–23.

BioMar Group A/S (2021). 2021 annual report.

Bjelland, A. T., Borgen, A., and Wold, S. (2021). A genetic greedy construction heuristic for the fish feed maritime inventory routing problem.

Brekkå, I. and Randøy, S. (2021). A decomposition-based adaptive large neighborhood search heuristic for the fish feed production routing problem.

Brekkå, I., Randøy, S., Fagerholt, K., Thun, K., and Vadseth, S. T. (2022). The fish feed production routing problem. *Computers Operations Research*, 144.

Chakraborty, A. (2021). Present bias.

Christiaens, J. and Vanden Berghe, G. (2020). Slack induction by string removals for vehicle routing problems. *Transportation Science*, 54(2):417–433.

Christiansen, M. (1999). Decomposition of a combined inventory and time constrained ship routing problem. *Transportation Science*, 33(1):3–16.

Christiansen, M. and Fagerholt, K. (2009). *Maritime Inventory Routing Problems*, pages 1947–1955. Springer US, Boston, MA.

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

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

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2014). Thirty years of inventory routing. *Transportation Science*, 48(1):1–19.

Dauzère-Pérès, S., Nordli, A., Olstad, A., Haugen, K., Koester, U., Myrstad, P., Teistklub, G., and Reistad, A. (2007). Omya hustadmarmor optimizes its supply chain for delivering calcium carbonate slurry to european paper manufacturers. *Interfaces*, 37:39—-51.

De, A., Kumar, S. K., Gunasekaran, A., and Tiwari, M. K. (2017). Sustainable maritime inventory routing problem with time window constraints. *Engineering Applications of Artificial Intelligence*, 61:77–95.

De Jong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, USA. AAI7609381.

Directorate of Fisheries (2021). Biomassestatistikk etter fylke.

Directorate of Fisheries (2022). Tildelingsprosessen.

Diz, G., Hamacher, S., and Oliveira, F. (2019). A robust optimization model for the maritime inventory routing problem. *Flexible Services and Manufacturing Journal*, 31:675–701.

Engineer, F. G., Furman, K. C., Nemhauser, G. L., Savelsbergh, M. W., and Song, J.-H. (2012). A branch-price-and-cut algorithm for single-product maritime inventory routing. *Operations Research*, 60(1):106–122.

EY (2020). The norwegian aquaculture analysis 2020.

Fodstad, M., Stremersch, G., Hecq, S., Uggen, K., Rømo, F., and Lium, A.-G. (2010). Lngscheduler: A rich model for coordinating vessel routing, inventories and trade in the lng supply chain. *Journal of Energy Markets*, 3:31–64.

Friske, M. W. and Buriol, L. S. (2017). A relax-and-fix algorithm for a maritime inventory routing problem. In Bektaş, T., Coniglio, S., Martinez-Sykora, A., and Voß, S., editors, *Computational Logistics*, pages 270–284, Cham. Springer International Publishing.

Friske, M. W. and Buriol, L. S. (2018). Applying a relax-and-fix approach to a fixed charge network flow model of a maritime inventory routing problem. In Cerulli, R., Raiconi, A., and Voß, S., editors, *Computational Logistics*, pages 3–16, Cham. Springer International Publishing.

Friske, M. W. and Buriol, L. S. (2020). A multi-start algorithm and a large neighborhood search for a maritime inventory routing problem. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Friske, M. W., Buriol, L. S., and Camponogara, E. (2022). A relax-and-fix and fix-and-optimize algorithm for a maritime inventory routing problem. *Computers Operations Research*, 137:270–284.

Grønhaug, R. and Christiansen, M. (2009). Supply chain optimization for the liquefied natural gas business. In Nunen, J. A., Speranza, M. G., and Bertazzi, L., editors, *Innovations in Distribution Logistics*, pages 195–218, Berlin, Heidelberg. Springer Berlin Heidelberg.

Grønhaug, R., Christiansen, M., Desaulniers, G., and Desrosiers, J. (2010). A branch-and-price method for a liquefied natural gas inventory routing problem. *Transportation Science*, 44(3):400–415.

Handeland, S. O., Imsland, A. K., and Stefansson, S. O. (2008). The effect of temperature and fish size on growth, feed intake, food conversion efficiency and stomach evacuation rate of atlantic salmon post-smolts. *Aquaculture*, 283(1):36–42.

Hartvigsen, R. (2019). Inventory managemement, scheduling and routing in fish feed distribution.

Haugland, M. G. and Thygesen, S. (2017). Use of clusters in a route generation heuristic for distribution of fish feed.

Hemmati, A., Hvattum, L. M., Christiansen, M., and Laporte, G. (2016). An iterative two-phase hybrid matheuristic for a multi-product short sea inventory-routing problem. *European Journal of Operational Research*, 252(3):775–788.

Hemmati, A., Hvattum, L. M., Fagerholt, K., and Norstad, I. (2014). Benchmark suite for industrial and tramp ship routing and scheduling problems. *INFOR Information Systems and Operational Research*, Volume 52:28–38.

Hemmati, A., Stålhane, M., Hvattum, L. M., and Andersson, H. (2015). An effective heuristic for solving a combined cargo and inventory routing problem in tramp shipping. *Computers Operations Research*, 64:274–282.

Jiang, Y. and Grossmann, I. E. (2015). Alternative mixed-integer linear programming models of a maritime inventory routing problem. *Computers Chemical Engineering*, 77:147–161.

Misra, S., Kapadi, M., and Gudi, R. D. (2020). Hybrid time-based framework for maritime inventory routing problem. *Industrial & Engineering Chemistry Research*, 59(46):20394–20409.

Mowi (2019). Salmon farming industry handbook 2019.

Mowi (2021a). Integrated annual report 2021.

Mowi (2021b). Salmon farming industry handbook 2021.

Norsk Industri (2017). Veikart for havbruksnæringen.

Papageorgiou, D., Cheon, M.-S., Nemhauser, G., and Sokol, J. (2014a). Approximate dynamic programming for a class of long-horizon maritime inventory routing problems. *Transportation Science*, 0:870–885.

Papageorgiou, D., Keha, A., Nemhauser, G., and Sokol, J. (2014b). Two-stage decomposition algorithms for single product maritime inventory routing. *INFORMS Journal on Computing*, 26:825–847.

Papageorgiou, D. J., Cheon, M.-S., Harwood, S., Trespalacios, F., and Nemhauser, G. L. (2018). Recent progress using matheuristics for strategic maritime inventory routing. In *Modeling, computing and data handling methodologies for maritime transportation*, pages 59–94. Springer.

Papageorgiou, D. J., Nemhauser, G. L., Sokol, J., Cheon, M.-S., and Keha, A. B. (2014c). Mirplib – a library of maritime inventory routing problem instances: Survey, core model, and benchmark results. *European Journal of Operational Research*, 235(2):350–366. Maritime Logistics.

Rakke, J. G., Andersson, H., Christiansen, M., and Desaulniers, G. (2015). A new formulation based on customer delivery patterns for a maritime inventory routing problem. *Transportation Science*, 49(2):384–401.

Ronen, D. (2002). Marine inventory routing: Shipments planning. *The Journal of the Operational Research Society*, 53(1):108–114.

Sanghikian, N., Martinelli, R., and Abu-Marrul, V. (2021). A hybrid vns for the multi-product maritime inventory routing problem. In Mladenovic, N., Sleptchenko, A., Sifaleras, A., and Omar, M., editors, *Variable Neighborhood Search*, pages 111–122, Cham. Springer International Publishing.

Siswanto, N., Wiratno, S. E., Rusdiansyah, A., and Sarker, R. (2019). Maritime inventory routing problem with multiple time windows. *Journal of Industrial and Management Optimization*, 15(3):1185–1211.

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

Speranza, M. and Archetti, C. (2014). A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, 2:223–246.

Stålhane, M., Rakke, J. G., Moe, C. R., Andersson, H., Christiansen, M., and Fagerholt, K. (2012). A construction and improvement heuristic for a liquefied natural gas inventory routing problem. *Computers & Industrial Engineering*, 62(1):245–255.

Statistics Norway (2021). Fastlandseksport, etter produksjonsfylke og varegruppe (sitc-basert) (mill. kr) (f) 2009 - 2021.

Uggen, K. T., Fodstad, M., and Nørstebø, V. S. (2013). Using and extending fix-and-relax to solve maritime inventory routing problems. *Top*, 21(2):355–377.

Vidal, T. (2022). Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Computers Operations Research*, 140:105643.

Whitley, D., Rana, S., and Heckendorn, R. B. (1999). The island model genetic algorithm: On separability, population size and convergence. *Journal of computing and information technology*, 7(1):33–47.

Appendix A

LP Model for Quantity Assignments

In this appendix, we present the linear mathematical model used for assigning quantities in SMOLT in its entirety.

A.1 Sets and Indices

$i \in \mathcal{I}^C$	–	Set of consumption ports
$i \in \mathcal{I}^P$	–	Set of production ports
$i \in \mathcal{I}$	–	Set of all ports, $\mathcal{I} = \mathcal{I}^C \cup \mathcal{I}^P$
$v \in \mathcal{V}$	–	Set of vessels
$t \in \mathcal{T}$	–	Set of time periods in the planning horizon
$p \in \mathcal{P}$	–	Set of different products

A.2 Parameters

A.2.1 Constraint-related Parameters

L^0_{vp}	–	The initial load of product $p \in \mathcal{P}$ on vessel $v \in \mathcal{V}$
$\overline{L}_v$	–	The total capacity of vessel $v \in \mathcal{V}$
S^0_{ip}	–	The initial inventory of product $p \in \mathcal{P}$ at port $i \in \mathcal{I}$
$\overline{S}_{ip}$	–	Total capacity of product $p \in \mathcal{P}$ at port $i \in \mathcal{I}$
D_{itp}	–	The consumption/production of product $p \in \mathcal{P}$ at port $i \in \mathcal{I}$ in time $t \in \mathcal{T}$.
I_i	–	Indicator of the port type. 1 if the node is a production port, and -1 if the node is a consumption port
F^{MAX}_{it}	–	The maximum amount that can be loaded/unloaded on/from a single vessel in port $i \in \mathcal{I}$ in time $t \in \mathcal{T}$
β_{it}	–	The maximum amount of all products that can be bought from the spot market by port $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$
β^{TOT}_i	–	The maximum amount of all products that can be bought from the spot market by port $i \in \mathcal{I}$ throughout the planning horizon

A.2.2 Objective-related Parameters

C^S_{it} – The unit cost of buying from the spot market for port $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$

R_{it} – The revenue generated per unit delivery to port $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$

τ – Scaling factor used for punishing excessive inventories in production ports and shortage in consumption ports

ε – Scaling factor used for making early deliveries more preferable

A.3 Variables

x_{ivtp} – The quantity loaded/unloaded from/to port $i \in \mathcal{I}$ from vessel $v \in \mathcal{V}$ of product $p \in \mathcal{P}$ at time $t \in \mathcal{T}$

s_{itp} – The current stock of product $p \in \mathcal{P}$ at port $i \in \mathcal{I}$ at the *beginning* of time period $t \in \mathcal{T}$

l_{vtp} – The current stock of product $p \in \mathcal{P}$ on vessel $v \in \mathcal{V}$ at the *beginning* of time period $t \in \mathcal{T}$

w_{itp} – excess inventory/shortage at port $i \in \mathcal{I}$ at the *beginning* of time period $t \in \mathcal{T}$ of product $p \in \mathcal{P}$

α_{itp} – amount bought from / sold to the spot market by port $i \in \mathcal{I}$ at the *beginning* of time period $t \in \mathcal{T}$ of product $p \in \mathcal{P}$

A.4 Model Formulation

A.4.1 Objective

The objective consists of four terms. One represents the revenue, one represents the advantage of delivering early, one represents the cost of buying/selling to the spot market, and the final term represents the punishment that occurs when inventory violations occur. The objective is given in Equation (A.4.1).

$$\max z = \sum_{i\in\mathcal{I}}\sum_{v\in\mathcal{V}}\sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}} (R_{it} - \varepsilon t)\, x_{ivtp} - \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}} \left(C^S_{it}\alpha_{itp} + \tau w_{itp}\right) \tag{A.4.1}$$

A.4.2 Port inventory

Constraints (A.4.2) set the initial inventory in all ports.

$$s_{i0p} = S^0_{ip}, \quad i \in \mathcal{I}, p \in \mathcal{P} \tag{A.4.2}$$

Constraints (A.4.3) ensure that the inventories are correctly updated in all time periods for all ports.

$$s_{itp} = s_{i,t-1,p} + I_i \left(D_{i,t-1,p} - \sum_{v\in\mathcal{V}} x_{iv,t-1,p} - \alpha_{itp} \right), \quad i \in \mathcal{I}, t \in \mathcal{T}\backslash\{0\}, p \in \mathcal{P} \tag{A.4.3}$$

Constraints (A.4.4) ensure that the inventories remain within the capacity limits plus the current excess inventory for all production ports.

$$0 \le s_{itp} \le \overline{S}_{ip} + w_{itp}, \quad i \in \mathcal{I}^P, t \in \mathcal{T}, p \in \mathcal{P} \tag{A.4.4}$$

Constraints (A.4.5) ensure that the inventories are held between the shortage and the capacity for all consumption ports.

$$-w_{itp} \leq s_{itp} \leq \overline{S}_{ip}, \quad i \in \mathcal{I}^C, t \in \mathcal{T}, p \in \mathcal{P} \tag{A.4.5}$$

A.4.3 Vessel inventory

Constraints (A.4.6) set the initial inventory for all vessels.

$$l_{i0p} = L^0_{vp}, \quad v \in \mathcal{V}, p \in \mathcal{P} \tag{A.4.6}$$

Constraints (A.4.7) ensure that the inventories are correct throughout the planning horizon for all vessels.

$$l_{itp} = l_{i,t-1,p} + I_i \left(\sum_{i \in \mathcal{I}} x_{iv,t-1,p} \right), \quad v \in \mathcal{V}, t \in \mathcal{T} \backslash \{0\}, p \in \mathcal{P} \tag{A.4.7}$$

Constraints (A.4.8) ensure that the total inventory onboard a vessel is within the capacity limits of the vessel at all times.

$$\sum_{p \in \mathcal{P}} l_{vtp} \leq \overline{L}_v, \quad v \in \mathcal{V}, t \in \mathcal{T} \tag{A.4.8}$$

A.4.4 Loading and unloading

Constraints (A.4.9) ensure that the total loaded/unloaded amount at all ports are within the rate constraints for that particular port in all time periods.

$$\sum_{p \in \mathcal{P}} x_{ivtp} \leq F^{\text{MAX}}_{it}, \quad i \in \mathcal{I}, v \in \mathcal{V}, t \in \mathcal{T} \tag{A.4.9}$$

Constraints (A.4.10) make sure that the amount bought from / sold to the spot market in all time periods is within the limit for that particular port in that time period.

$$\sum_{p \in \mathcal{P}} \alpha_{itp} \leq \beta_{it}, \quad i \in \mathcal{I}, t \in \mathcal{T} \tag{A.4.10}$$

Constraints (A.4.11) make sure that the amount bought from / sold to the spot market in throughout the planning horizon is is within the available amount for all ports.

$$\sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}} \alpha_{itp} \leq \beta^{\text{MAX}}_i, \quad i \in \mathcal{I} \tag{A.4.11}$$

A.4.5 Non-Negativity Constraints

$$x_{ivtp} \geq 0, \quad i \in \mathcal{I}, v \in \mathcal{V}, t \in \mathcal{T}, p \in \mathcal{P} \tag{A.4.12}$$

$$l_{vtp} \geq 0, \quad v \in \mathcal{V}, t \in \mathcal{T}, p \in \mathcal{P} \tag{A.4.13}$$

$$w_{itp} \geq 0, \quad i \in \mathcal{I}, t \in \mathcal{T}, p \in \mathcal{P} \tag{A.4.14}$$

$$\alpha_{itp} \geq 0, i \in \mathcal{I}, t \in \mathcal{T}, p \in \mathcal{P} \tag{A.4.15}$$

A.5 Extensions

In this section, we present some extensions to the basic LP model. The extensions are mainly developed for solving compatibility issues with the MIRPLib benchmark instances.

A.5.1 Travel at Capacity

To make the results of SMOLT comparable to the benchmark results on the MIRPLib instances, there was in particular one thing we had to incorporate. As mentioned in Section 7.3, the MIRPLib instances have constraints forcing all vessels to leave production ports fully loaded, and to return to the production ports empty (hereafter referred to as the "travel at capacity" constraints). This is a simplification of the general MIRP problem as this is not a hard constraint in reality, and it is not implemented in SMOLT by default. That being said, good solutions often fully exploit the vessels' capacities and are therefore likely to respect the travel at capacity constraints even if they are not explicitly defined. In fact, SMOLT tends to produce solutions that either respect or nearly respect the travel at capacity constraints, simply because such solution characteristics are favourable. However to further increase the attractiveness of solutions respecting these constraints, we have added a penalty term punishing solutions that do not. This extension is presented in this section.

First, we have to introduce additional sets, parameters and some variables.

Sets

$t \in \mathcal{T}_v^P$ – Set of time periods vessel $v \in \mathcal{V}$ arrives at a production port during the planning horizon

$t \in \mathcal{T}_v^C$ – Set of time periods vessel $v \in \mathcal{V}$ arrives at a consumption port right after a production port visit during the planning horizon

$t \in \mathcal{T}_v$ – $\mathcal{T}_v = \mathcal{T}_v^P \cup \mathcal{T}_v^C$

Parameters

σ^C – Weight for adjusting the "travel at capacity" violations at consumption ports

σ^P – Weight for adjusting the "travel at capacity" violations at production ports

Variables

γ_{vt} – The "travel at capacity" constraint violation by vessel $v \in \mathcal{V}$ when arriving at a port in time period $t \in \mathcal{T}_v$

The sets $\mathcal{T}_v^P$ and T_v^C are calculated before the LP model is optimized. As known, upon running, the routing solution is already fixed from the MA. Thus, before optimizing the quantities, we already know when the vessels arrive at production ports and leave them, and can calculate the set.

$$\gamma_{vt} = \overline{L}_v - \sum_{p \in \mathcal{P}} l_{vtp}, \quad v \in \mathcal{V}, t \in \mathcal{T}_v^C \tag{A.5.1}$$

Further, we can simply use l_{vtp} to punish the vessel when arriving at production ports with load as this should be 0 to avoid any punishment.

We add the two "punishment terms" to the objective, impose a non-negativity constraint on γ_{vt}, and get the final extended model.

$$\max z = \sum_{i \in \mathcal{I}} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}} (R_{it} - \varepsilon t) x_{ivtp} - \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}} (C_{it}^S \alpha_{itp} + \tau w_{itp}) - \sum_{v \in \mathcal{V}} \Big(\sum_{t \in \mathcal{T}_v^C} \sigma^C \gamma_{vt} + \sum_{t \in \mathcal{T}_v^C} \sum_{p \in \mathcal{P}} \sigma^P l_{vtp} \Big) \tag{A.5.2}$$

$$\gamma_{vt} \geq 0, \quad v \in \mathcal{V}, t \in \mathcal{T} \tag{A.5.3}$$

A.5.2 Semi-continuous Restrictions on Loading and Unloading

The MIRPLib instances require the loading and unloading quantities to be semi-continuous. This can easily be incorporated into the LP presented above by converting the continuous l variables to be semi-continuous. However, this will cause the LP to become a MIP, which is significantly more time-consuming to solve. As a result, this would reduce the efficiency of SMOLT drastically, and the semi-continuous restrictions were therefore not implemented directly in SMOLT. To handle these constraints, SMOLT can be applied as if they did not exist. The routing solution obtained by SMOLT is then given to an extended version of the LP that enforces the semi-continuous constraints that calculates the quantities. If it is impossible to satisfy the semi-continuous constraints given the routing solution, the solution is infeasible. This approach is not optimal, however, as SMOLT was not originally designed to solve the MIRPLib instances, and because the semi-continuous constraints are not enforced in reality, it was considered acceptable.

A.5.3 Compartment Assignment

The Mowi instances provide a further complexity over the MIRPLib instances by requiring feed to be carried in distinct silos, each of which is only capable of carrying a single type of feed at a time. In order to adhere to these restrictions, we add the following to the model.

Sets

$c \in \mathcal{C}_v$ – Set of compartments onboard vessel $v \in V$

Parameters

C_c^C – Capacity of compartment $c \in \mathcal{C}_v$

Variables

ζ_{vtpc} – Binary variable saying whether compartment $c \in \mathcal{C}_v$ aboard vessel $v \in \mathcal{V}$ is assigned to carry product $p \in P$ at time $t \in T$

The load carried on a vessel is required to fit into a set of assigned compartments. This is enforced by Equation (A.5.4).

$$l_{tvp} \leq \sum_{c \in \mathcal{C}_v} C_c^C \zeta_{vtpc}, \quad v \in \mathcal{V}, t \in \mathcal{T}, p \in \mathcal{P} \tag{A.5.4}$$

Each compartment is only allowed to carry at most one product type. This is enforced by Equation (A.5.5).

$$\sum_{p \in \mathcal{P}} \zeta_{vtpc} \leq 1, \quad v \in \mathcal{V}, t \in \mathcal{T}, c \in \mathcal{C}_v \tag{A.5.5}$$

Lastly, we put binary constraints on the assignment variables ζ_{vtpc}. This is enforced by Equation (A.5.6).

$$\zeta_{vtpc} \in \{0,1\}, \quad v \in \mathcal{V}, t \in \mathcal{T}, c \in \mathcal{C}_v, p \in \mathcal{P} \tag{A.5.6}$$

Note that these constraints only enforce that it should be possible to carry each load by assigning some combination of the available compartments to different products. It does not put any constraints on how this assignment can change over time. As such, a solution is allowed to reorder compartment assignment at each time step. Keeping track of compartment assignment over time would require us to index the quantity unloaded and loaded by compartment, which would significantly increase the size of the model.

Appendix B

Details: Slack Induction by String Removal

Our adaptation of SISR to the MIRP mainly differs in how it considers proximity when removing strings. While the original study by Christiaens and Vanden Berghe (2020) considers the closest nodes by distance, we instead use a combination of time and space. This chapter describes the aspects of SISR that was left out from the main text for the sake of brevity. All of these aspects are largely unchanged from the original SISR (Christiaens and Vanden Berghe, 2020).

B.1 Deciding on the Number and Length of Strings to Remove

The number of strings to remove and how long each string should be is stochastic, but controlled by two parameters: L^{MAX} and $\overline{c}$. L^{MAX} sets an upper limit for the size of removed strings, while $\overline{c}$ controls the average number of visits removed. In combination, these allow us to adjust how many strings to remove as well as the length of each string.

With L^{MAX} and $\overline{c}$ given, the maximum size of the removed strings l_s^{MAX}, the number of strings to be removed k_s, and the length l_t of a string taken from vessel plan t is given by:

$$l_s^{\text{MAX}} = \min\left(L^{\text{MAX}}, \text{AVERAGE VESSEL PLAN LENGTH}\right) \tag{B.1.1}$$

$$k_s = \left\lfloor U\left(1, \frac{4\overline{c}}{1 + l_s^{MAX}}\right)\right\rfloor \tag{B.1.2}$$

$$l_t = \left\lfloor U\left(1, \min\left(|t|, l_s^{\text{MAX}}\right) + 1\right)\right\rfloor \tag{B.1.3}$$

where $U(a, b)$ is a number drawn from the continuous uniform distribution between a and b.

B.2 "Split String" Procedure

In Christiaens and Vanden Berghe (2020), the authors present two string removal procedures: "string" and "split string". This work has only adapted the "string" procedure to the MIRP; the "split string" procedure was not implemented.

Appendix C

Discarded Mutations

Several other mutation operators were implemented and tested in addition to the ones mentioned in Section 6.4.3. During testing, these were found to provide little or no additional benefit to solution speed and quality, and were therefore discarded. However, for completeness, we give a summary of them here. See Table C.1. The implementations can be found on our GitHub.

Table C.1: Summary of mutations that were discarded from SMOLT.

Mutations	**Description**
Best move	Remove the most expensive visit in terms of distance from a random route and do a greedy reinsertion in the *same* route
Relocate	Remove the most expensive visit in terms of distance from a random route and do a greedy reinsertion in *any* route
Reduced cost	Give more time before/after the visit that has the best reduced cost w.r.t. quantity delivered
Replace node	Choose a random visit, and replace the node of the visit with the node that gives the best fitness
Time setter	Given that the order of node visits remain unchanged, use a continuous LP to decide quantities and arrival times at each node
Vessel swap	Swap the routes of two vessels

Aksel Borgen, Anders Bjelland and Sjur Wold

A Memetic Algorithm With Optimal Quantity Assignments for the Fish Feed Maritime Inventory Routing Problem

NTNU
Norwegian University of
Science and Technology