



Norwegian University of
Science and Technology

A New Optimization-based Algorithm for a Maritime Inventory Routing Problem

Elise Foss

Trine Nord Myklebust

Industrial Economics and Technology Management

Submission date: June 2016

Supervisor: Marielle Christiansen, IØT

Co-supervisor: Henrik Andersson, IØT

Norwegian University of Science and Technology

Department of Industrial Economics and Technology Management

Problem description

The purpose of this thesis is to develop optimization model(s) and method(s) to solve a multi-product maritime inventory routing problem (MIRP) with undedicated compartments. A problem within a tactical planning horizon will be considered taking ship routing and scheduling, inventory management and the allocation of products on board the ships into consideration. Due to the complexity of the problem, the focus in this thesis will be to develop suitable solution method(s).

Main contents:

- Description of the problem
- Overview and discussion of relevant literature
- Development of mathematical model(s) of the underlying problem and suitable solution method(s)
- Implementation of mathematical model(s) and method(s) using commercial software
- Testing and discussion of model(s) and method(s)
- An evaluation of the model(s) and solution method(s) developed

Preface

This master thesis is the final result of the work for our Master in Science degree at the Norwegian University of Science and Technology. The degree specialization is Applied Economics and Operations Research at the Department of Industrial Economics and Technology Management. The thesis was written during the spring semester of 2016, and is a continuation of our work for the project report in the fall semester 2015. The work done for the project report has resulted in a paper submitted revised to Lecture Notes in Computer Science, Springer Verlag (Foss et al., 2016). The submitted paper can be found in Appendix A.

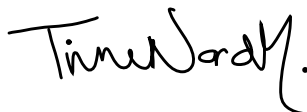
The purpose of this thesis is two folded. First, we introduce a formulation for a multi-product maritime inventory routing problem with a particular focus on the handling of allocation of products on the ships. Second, we present a new optimization based algorithm for solving the highly complex multi-product MIRP formulated.

We would like to thank our supervisors Marielle Christiansen and Henrik Andersson for their valuable insights and helpful guidance, but also their sincere enthusiasm about our work. We have been lucky to work with two true experts on the area of maritime transportation.

Trondheim, June 09th, 2016



Elise Foss



Trine Nord Myklebust

Abstract

Maritime transportation has a long tradition of taking a dominant role in the global trade. Remarkable improvements in the efficiency of maritime transportation have been made in the last 50 years, but still significant improvements can be made by improving the routing and scheduling of ships through the use of operations research (OR).

This thesis considers the problem of routing bulk tankers to minimize costs while managing the inventory in ports. Multiple unmixable products are transported and the allocation of products to undedicated compartments on board the ships is an important aspect of the problem. One aim of the thesis is to contribute to the literature on maritime inventory routing problems (MIRPs) by including the dynamic allocation of products to undedicated compartments on board the ships. An arc-flow formulation of the problem is proposed together with several types of valid inequalities and tightening constraints. The model has been tested on three differently sized test cases. The valid inequality imposed on the minimum number of compartments needed in each port was the most efficient valid inequality.

The formulation proposed proved difficult to solve with an exact solution method within a reasonable time. We propose a new optimization based algorithm for solving the multi-product MIRP. The algorithm is an iterative matheuristic, and it utilizes the structure of the problem. In a construction phase, the routing component of the problem is extracted and solved independently based on the series of valid inequalities designed. When the solution of the routing problem is known, the remaining part of the problem is solved with the sequence of port visits from the routing problem fixed. If the problem is not feasible given the current routes, new routes are generated utilizing information from the infeasibilities. The iterations continue until a feasible solution is found. An improvement heuristic is utilized to improve the discovered feasible solution.

By decomposing the problem, the size of each of the two subproblems is reduced remarkably. The effectiveness of the exact solution method decreases with the size of the test case. Simultaneously, the effectiveness of the matheuristic increases with the size of the test case. On average, the construction phase of the matheuristic returns high quality solutions in significantly shorter time than the exact solution method. The solutions are further improved in the improvement phase. The proposed matheuristic represents a remarkable improvement in both efficiency and solution quality when solving the complex MIRP.

Sammendrag

Sjøtransport har lenge hatt en dominerende rolle i det globale handelsmarkedet. Til tross for at sjøtransportnæringen har opplevd signifikante forbedringer de siste 50 årene, finnes det fortsatt et stort forbedringspotensialet på flere områder. Både ruting og timeplanlegging av skip har potensiale til å bli effektivisert ved hjelp av operasjonsanalyse.

Denne masteroppgaven tar for seg et problem hvor både ruteplanleggingen av bulk tankskip, samt lagerstyring av et flertalls produkter i havner skal optimeres. Problemet optimeres ved å minimere kostnader. Flere produkter som ikke kan blandes transporteres av tankskipet og allokeringen av disse produktene til ikke dedikerte tanker er en essensiell del av problemet. Et av målene med masteroppgaven er å bidra til litteraturen på kombinerte lagerstyrings- og ruteplanleggingsproblemer ved å inkludere dynamisk allokering av produkter til ikke-dedikerte tanker på skipet. En arc-flow formulering, samt en serie med gyldige ulikheter, er designet for dette problemet. Modellen har blitt testet med en eksakt løsningsmetode på tre datasett av forskjellig størrelse. Den gyldige ulikheten på minste antall tanker nødvendig i hver havn for hvert produkt presterte best og hadde størst effekt på kjøretiden.

Til tross for utviklingen av gyldige ulikheter, er modellen for kompleks til å kunne løses innen rimelig tid med en eksakt løsningsmetode. En optimeringsbasert algoritme ble utviklet for å kunne løse modellen mer effektivt. Algoritmen er en iterativ matheuristikk som utnytter strukturen til problemet. I en konstruksjonsfase, trekkes selve ruteplanleggingen ut av problemet og løses uavhengig ved hjelp av de gyldige ulikhetene som er utviklet. Resten av problemet løses så med ruteløsningen fiksert. Hvis løsningen ulovlig gitt rutene fra ruteproblemene, genereres nye ruter i ruteproblemene med informasjon om ulovlighetene ved nåværende løsning. Det iterative søket fortsetter til algoritmen finner en lovlig løsning til det originale problemet. En forbedringsheuristikk benyttes til å forbedre den lovlige løsningen.

Ved å dekomponere problemet reduseres hvert av de to delproblemene signifikant i størrelse. Effektiviteten til den eksakte løsningsmetoden reduseres med størrelsen på datasettet. Samtidig øker effektiviteten til matheuristikken med størrelsen på datasettet. Konstruksjonsfasen returnerer løsninger av høy kvalitet på mye kortere tid enn den eksakte løsningsmetoden. Løsningene blir videre forbedret av forbedringsheuristikken. Matheuristikken som er utviklet har vist seg å være en bemerkelsesverdig bedre løsningsmetode for den komplekse modellen enn den eksakte løsningsmetoden.

Contents

Problem description	i
Preface	iii
Abstract	v
Sammendrag	vi
1 Introduction	1
1.1 The shipping industry	1
1.2 Motivation	5
1.3 The thesis problem and purpose	6
1.4 Organization of the thesis	7
2 Problem Description	8
3 Related Literature	11
3.1 Maritime inventory routing problems	12
3.2 MIRPs with multiple products and allocation	16
3.3 Matheuristics as a solution approach	17
4 Model Description and Valid Inequalities	25
4.1 Model assumptions	25
4.2 Multi-product MIRP-UC formulation	27
4.3 Valid inequalities	32
5 Solution Methods	41
5.1 Matheuristic solution method	41
5.2 Construction phase: The routing problem	43
5.3 Construction phase: The inventory problem	46
5.4 Construction phase: Handling of infeasibilities	50
5.5 Improvement phase	63
6 Computational Study	67
6.1 Exact solution method	67
6.2 Matheuristic solution method	75
6.3 Practical implications	104

7	Concluding Remarks and Further Research	113
7.1	Conclusion	114
7.2	Further research	115
	Bibliography	116
A	ICCL paper, 2016	I

List of Figures

1.1	Transport volume of seaborne trade from 1990 to 2013	2
1.2	World fleet by vessel type 2014	3
1.3	World tonnage orders by vessel type	4
2.1	Example of the routing of two ships between five ports	9
2.2	Example of the allocation of three products to a ship with two compartments	10
4.1	Example of a ship route consisting of sailing s , waiting w , and operating o	26
4.2	Illustration of variables	29
4.3	Development of sailing cost over the planning horizon. The total sailing cost in the IP solution is 31 640, while the sailing cost in the LP solution is 1 314	33
4.4	Illustration of excess production and consumption over the planning horizon	35
4.5	Determine the time interval for the occurrence of the first visit	37
4.6	Determine the time interval for the occurrence of the second visit	38
4.7	Determine the time interval for the occurrence of the last visit	39
5.1	Flow chart representing the iterative matheuristic scheme	42
5.2	Illustration of the functionality of the inventory problem	47
5.3	Illustration of Strategy 1 functionality for a consumption port violation . .	52
5.4	Illustration of Strategy 3 functionality for a consumption port violation . .	55
5.5	Illustration of issue with multiple sources	58
5.6	Flow chart presenting the matheuristic when Strategy 5 is included	62
5.7	Safety inventory of routing problem	62
5.8	Illustration of the Intra ship improvement operator	64
5.9	Illustration of the Inter ship improvement operator	65
6.1	Discovery of feasible solutions in Mosel Xpress-MP	69
6.2	Development of objective value in the routing and inventory problem in each iteration. Dotted/solid line represents the routing/inventory problem objective value.	92
6.3	Illustration of small test case solution	105
6.4	Illustration of routing solution for <i>Ship 4</i> in the large test case	106
6.5	Comparison of routing solution when different lengths of planning horizons are applied	108

6.6	Illustration of construction phase solution with the greedy strategy. <i>Ship 1</i> marked in orange and <i>Ship 2</i> marked in blue. The red tab illustrates a violation with the violation product given inside the parenthesis	109
6.7	Illustration of construction phase solution with the valid strategy. <i>Ship 1</i> marked in orange and <i>Ship 2</i> marked in blue. The red tab illustrates a violation with the violation product given inside the parenthesis	110
6.8	Comparison of construction phase input route and the resulting route from executing the Intra operator	111
6.9	Comparison of Intra operator input route and the resulting route from executing the Inter operator	112

List of Tables

3.1	Classification of literature considering matheuristics	20
6.1	Notation for exact solution method test instances and valid inequalities . .	68
6.2	Problem size of exact solution method instances	70
6.3	Results of small-, medium- and large-sized test cases with different valid inequalities. Running time 5000 seconds	71
6.4	Results on combination of valid inequalities for all test cases. Running time 5000 seconds	73
6.5	Notation for matheuristic solution method strategy settings	77
6.6	Parameter testing: First iteration, Greedy strategy	80
6.7	Parameter testing: First iteration, Valid strategy	81
6.8	Benchmark from exact solution method results	83
6.9	Test results of construction phase test instances for the small test cases . .	84
6.10	Test results of construction phase test instances for the medium test cases	87
6.11	Test results of construction phase instances for the large test cases	88
6.12	Overall average construction phase results for the different strategy settings	90
6.13	Development of the size of the routing problem	92
6.14	Scalability results of the greedy strategy	93
6.15	Parameter testing: Inter operator	96
6.16	Results from testing the Intra, Inter and Combination improvement operators	99
6.17	Average improvement phase results of the three chosen strategy settings . .	101
6.18	Average improvement phase results of the different improvement operators	102
6.19	Average matheuristic test results	103

Chapter 1

Introduction

The purpose of this chapter is to first give a brief overview of relevant aspects of the maritime transportation industry, as well as providing an introduction to the problem modelled in this thesis. Further, we also provide a motivation and purpose for studying the thesis problem.

1.1 The shipping industry

1.1.1 Maritime transportation

Maritime transportation is a prominent player in the global economy through its dominating role in the global trade industry (Michel & Noble, 2008). As a result of the free market forces in the global trading industry, the maritime transportation segment has managed to achieve a cost efficient structure. By providing a low-cost movement of goods around the world, the shipping industry has contributed to expanding the global economy. According to AON (2012), 90 % of all goods traded across borders are moved by the shipping industry. In 2013, 9.6 billion tons of cargo was transported using seaborne transportation on a fleet of size 1.69 billion dwt. By 2015 the fleet size had increased to 1.75 billion dwt (AON, 2012). Figure 1.1 shows the historical development in volume transported by sea from 1990 to 2013, corresponding to a 6% average yearly increase. Preliminary estimates presented by UNCTAD (2015) indicated that the volume of world seaborne shipments expanded with 3,4% in 2014.

With the global economical growth as main driver of demand, the drivers of revenue are also highly dependent on the global economical state. The global financial crisis in 2007-2009 strongly impacted the maritime transportation industry by slowing down demand for maritime shipments as the economics worldwide deteriorated (AON, 2012). The ongoing drop in the oil price since June 2014 has also affected shipping and seaborne trade. The drop affects the industry indirectly through changes in the areas of activity and changes in the sector generating demand for maritime transport services (UNCTAD, 2015). However, the overall growth in world GDP, merchandise trade and seaborne shipments was expected to continue at a moderate pace in 2015 (UNCTAD, 2015).

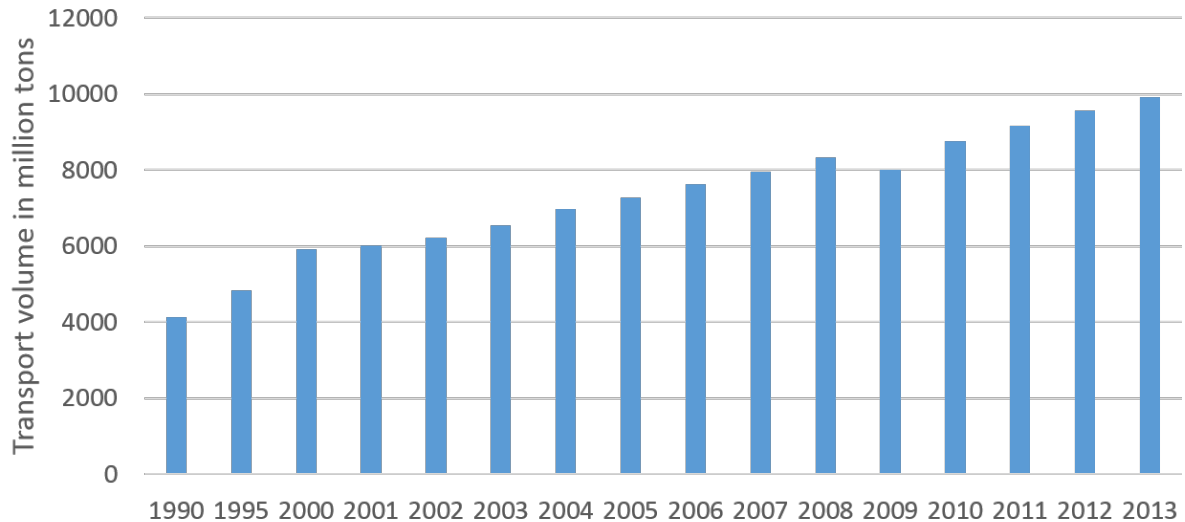


Figure 1.1: Transport volume of seaborne trade from 1990 to 2013

The cost structure of the industry is dependent on multiple factors, including the size of the vessels and distance of travel. The maritime transportation industry is both labor- and capital-intensive. The principal expenses are the costs and maintenance of the ships, salaries, equipment, and fuel (AON, 2012). As the revenue side of the profit function suffers from low oil prices, the cost segment is positively affected. A lower oil price leads to lower fuel and consequently transportation costs. This again reduces the expenditures and rates paid by the ship operators. The reduction in costs can in turn stimulate the demand for maritime transport services and increase the seaborne cargo flows (UNCTAD, 2015).

Another important aspect impacting the business of the operators is the investment activity. The investment activity in the maritime transportation industry is also correlated with the global economical state and the oil price. According to AON (2012), the maritime transport industry tends to follow boom-and-bust patterns. In good times, the industry tends to be overly optimistic causing an over-building of new vessels. The result is an over-supply of carrying capacity and decreased rates. Following the optimistic times with over-building of ships is a drop-off period where production of new ships is reduced. As a result, demand surpasses the supply and when demand rises again there is a mismatch between supply and demand that causes prices to spike.

1.1.2 Segments of the shipping market

There are three large international shipping market segments, namely the tanker, dry bulk and the container market (AON, 2012). Tankers mainly transport "wet" cargos that are often energy related, such as crude oil and petroleum products like LNG. The dry-bulk shipping market consists principally of the six cargo types; iron, ore, grain, coal, bauxite/alumina and phosphate. The market also includes a variety of other dry bulk products, with scrap metal as an example. Finally, the container market primarily, as the name suggests, includes products that can be shipped as containerized cargo (UNCTAD, 2015).

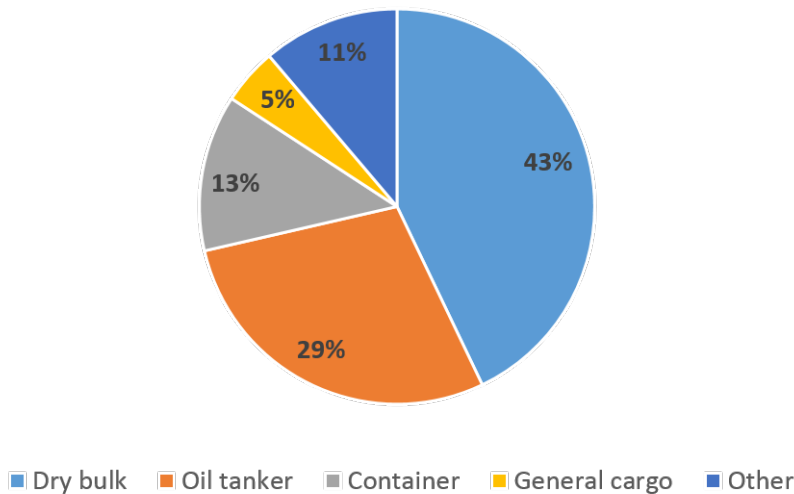


Figure 1.2: World fleet by vessel type 2014

Figure 1.2 displays the world fleet by vessel type in 2014 (UNCTAD, 2015). Evidently, dry bulk has the highest market share with 43% of the vessels in the markets, while tankers is the second largest with 29% market share. Other cargo ships and barges include general cargo ships, reefer ships, roll on-roll off (Ro-Ro) vessels, car carriers, forest-product carriers, barge carriers, heavy-lift ships, dry and tank barges, and articulated tug and barge units. Many of these are specialized vessels designed to carry specific types of cargoes.

1.1.3 Modes of operation

It is possible to distinguish between three different modes of operation in the maritime transportation industry, namely liner, tramp and industrial shipping. Christiansen et al. (2007) make a comparison between liner operations and bus lines, because both operate on a predetermined schedule. An example of liner operations is container shipping. Tramp shipping, however, can be compared with a taxicab as the ships follow the available cargoes and can service both contracted and spot market cargoes (Christiansen et al., 2007). The operators of both liner and tramp operations have the objective to maximize profit. The aspiration of industrial operators are different from the two other modes of operation. In industrial shipping operations, the operators usually both own the cargoes to be shipped and control the means of transportation. Consequently, the objective is always to minimize the cost of transportation and handling of the cargoes. Industrial and tramp shipping normally transport liquid or dry bulk cargoes shipped in large quantities. Figure 1.3 shows the historical development of tonnage orders over the last 14 years. As can be seen, bulk carriers stand for almost 50% of the orders in 2014.

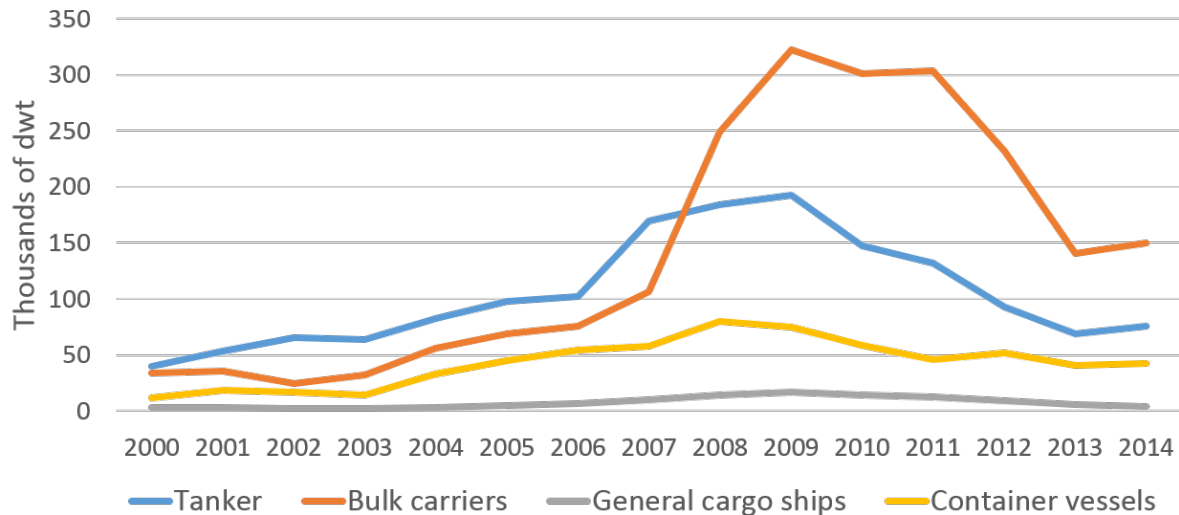


Figure 1.3: World tonnage orders by vessel type

1.1.4 Levels of planning

Shipping companies are faced with three levels of decisions; strategic, tactical and operational, depending on the planning horizon used (Christiansen et al., 2007). The strategic decisions are long term decisions with a time span of several years. These decisions include problems like fleet size and mix, port and facility location, and ship design problems. Strategic decisions lay the foundation for the tactical and operational decisions. The tactical planning level has a shorter time horizon, usually limited to one year. It includes problems like ship routing and scheduling and inventory management problems. Lastly, the operational planning level includes problems like sailing speed, pickup of spot cargoes and allocation of products to compartments on the ships. The decisions are made on a daily or weekly basis. Operational short-term planning problems are used when uncertainty in the operational environment is high, the situation is dynamic or when decisions have short-term impact. Due to the dependency across decision levels, it is essential for a shipping operator to make good decisions on all levels (Christiansen et al., 2007).

1.1.5 Geographical characteristics

Maritime transportation also differ in the geographical characteristics of the shipping routes. Shipping routes are normally classified as either short-sea, deep-sea or coastal and inland waterways. The type and size of the vessels used vary with the geographical characteristics (Christiansen et al., 2007). Short-sea shipping often use smaller container ships to transport goods over relatively short distances. Deep-sea shipping covers inter-continental cross-ocean transportation legs and utilize larger sized vessels. The deep-sea shipping segment can thus enjoy the benefits of economies of scale (Christiansen et al., 2007). Hemmati et al. (2016) emphasize that in contrary to deep-sea shipping, short sea shipping consisting of relatively short voyage legs compared to the time spent in ports. Lastly, coastal and inland waterways are mainly used by barges (Christiansen et al., 2007).

1.1.6 Maritime inventory routing problems

A maritime inventory routing problem (MIRP) is defined as a planning problem considering both the ships' routing and scheduling as well as the inventory management at one or both ends of the maritime transportation legs. MIRPs are considered to be industrial planning problems where the mainstay are liquid or dry bulk cargoes that are shipped in large quantities. MIRPs are often relevant in the context of maritime supply chain management when the inventory management and the routing and scheduling of ships have to be coordinated simultaneously (Christiansen & Fagerholt, 2014). For instance, vertically integrated companies are often responsible for their own internal transportation. It is an increasing trend that supply chains take advantage of the possibility of synchronizing production and inventories at the ports (Christiansen et al., 2004).

1.2 Motivation

As already stated, maritime transportation is an integral part of global trade. Maintaining time- and cost-efficient transportation systems are important factors in the world's economic development. Ship routing and scheduling is the major determinant of the fleet productivity (Christiansen et al., 2013). Thus, the fleet utilization is at the core of the overall efficiency of the shipping industry. Improvement in time efficiency of maritime transportation operations is an obvious effect of an increase in the fleet utilization. Further, the building of ships impose a major capital investment and daily operating costs of a ship can be tens of thousands of dollars. With a higher degree of fleet utilization, high investments in possibly redundant ship capacity and high daily costs can be saved. It is also worth mentioning that an increase in fleet utilization can help reduce the negative environmental impact of maritime transportation due to the decrease of unnecessary use of the fleet. From this, it is clear that the potential benefits from the use of decision support systems (DSS) in ship scheduling are significant (Christiansen et al., 2004).

The focus on maritime supply chain optimization and the use of MIRPs in operations research (OR) has increased during the last two decades. From 2002-2007, five new research papers was published on the topic maritime inventory management (Christiansen et al., 2013). The next five years, from 2007-2011, this number had more than doubled with 11 new research papers published. Since 2011 until now, several additional papers have been published exploring this topic further. Coordinating the planning challenges of ship routing and inventory management with the use of MIRPs has proven to give monetary benefits, flexibility in services and improved robustness (Christiansen & Fagerholt, 2014). Until now, the focus in most of the literature addressing maritime inventory routing has been on single-product MIRPs. Lately, the attention of the research community has shifted towards multi-product MIRPs. Christiansen et al. (2013) highlight this and propose the allocation of products to the ship's compartments as a relevant issue for further research.

Due to the growth in world seaborne trade and the increase in the use of DSS in the maritime transportation industry, operations research (OR) in shipping is a field that will continue to grow.

1.3 The thesis problem and purpose

The purpose of this thesis is to give further attention to maritime inventory routing, more specifically, a multi-product MIRP with undedicated compartments (multi-product MIRP-UC). The problem studied is a MIRP considering the transportation of multiple products while managing the inventory in ports and the allocation of products to compartments. The background for the development of the model is solely theoretical and is not grounded in a real operation. To our knowledge, combining multiple products with partial loading, undedicated compartments and allocation as a decision variable, is new to the literature. Most of the literature that do exist on MIRPs with multiple products, simplifies the allocation of products by assigning them to dedicated compartments or disregarding allocation completely. This contribution is the first, to our knowledge, that use undedicated compartments dynamically and without the assumption of empty compartments when returning to the production ports.

The MIRP is an operational and tactic decision problem and we consider short planning periods of 2-4 weeks. The combination of inventory management and ship routing and scheduling makes the MIRP a very complex problem to solve (Christiansen et al., 2013). Extending the basic MIRP to include multiple products and dynamic allocation to undedicated compartments increase the complexity even more. As for all complex optimization models, a trade off must be made between the solution speed of the problem and the reality representation of the model (Nygreen et al., 1998).

As the multi-product MIRP is highly complex, it is essential to develop a solution method appropriate for the structure of the problem. In this thesis, we develop an optimization based algorithm that can handle the high complexity of the multi-product MIRP. The currently existing literature on MIRPs has only been able to solve small instances using exact solution methods. Simultaneously, much interesting work has been done on developing heuristics for this type of problem. Equivalently, much promising and relevant work has been conducted in the overlapping field between exact solution methods and heuristic approaches, namely matheuristics. In later years, the interest for hybrid heuristics and matheuristic have increased (Boschetti et al., 2009). Currently, no extensive research has been done on matheuristics for multi-product MIRPs. Consequently, to pursuit a development of a matheuristic for the thesis problem is both constructive for the progression in research as well as an interesting and exciting topic to work with.

1.4 Organization of the thesis

The thesis is organized as follows. In Chapter 2, we give a detailed problem description for our multi-product MIRP-UC. In Chapter 3, we review relevant literature that exists on MIRPs today, with extra focus on models considering multiple products and allocation. Literature on alternative and specialized solution methods for MIRPs are also reviewed. This is done to get an overview of the different modelling choices that need to be made and how similar problems have been solved before. Chapter 4 contains the model description, including model assumptions, modelling choices, and the formulation developed along with the valid inequalities designed to tighten the formulation. Following is Chapter 5 where a matheuristic as an alternative solution method is presented and thoroughly described. The results from the computational study on both the exact and the matheuristic solution method are presented in Chapter 6. The chapter include both a thorough technical study as well as a discussion on practical implications. Finally, in Chapter 7, we conclude the master thesis and make suggestions for further research.

Chapter 2

Problem Description

The multi-product MIRP-UC in this thesis considers the transportation of multiple unmixable products and the allocation of the products to undedicated compartments on board the ships. The problem can be considered in context of the bulk market segment and is especially relevant for the shipping of liquid bulk products. It is a short-sea transportation problem with a planning horizon of 2-4 weeks. The problem is solved with respect to the ship routing and scheduling, inventory management in ports and the allocation of products to compartments on each ship. The objective is to minimize the total costs of the ship routing and loading/unloading operations.

Minimization of costs. The objective is to minimize the total costs consisting of the following components:

- Sailing costs between ports
- Operating costs in port
- Waiting costs outside a port
- Loading/unloading unit costs in port

Since the operator is responsible for both the inventory in ports and the routing of the ships, the inventory holding costs are ignored.

Ports. The problem considers the transportation of multiple products between the set of ports. The distribution of products can be characterized as a many-to-many distribution network, meaning that a ship can pick-up and deliver products to all the ports. A port can both produce and consume any number of products, but each port either produce or consume a product over the planning horizon. There are no fixed origin or destination ports in the route of a ship. Each port has a berth capacity restricting the number of ships operating in the port simultaneously.

Inventory. All ports have storages with inventory limits. The fleet of ships must be scheduled such that inventory in all ports are within its limits at all times. There is one separate storage for each product in each port. If a port neither produces nor consumes a

given product during the planning horizon, the product is not handled by that port and the port does not have a storage of that product. The inventory limits are constant upper and lower bounds specified for each port and each product. The initial inventory for each product in all ports is known.

Routing. Each ship has an initial start position either in a port or a point at sea. The sailing time between all ports is known for all ships. A ship is not allowed to visit a port (i.e sail to a port) without operating in that port. When a ship arrives at a port, it may wait outside the port before starting to load and unload products. In this thesis, we use the word operate when referring to the activity of loading or unloading products during a ship's port visit. Waiting outside a port may be necessary if there is no available berth capacity in the port, or to better time the start of operation with the inventory levels in the port. However, once the ship has started to operate in a port, the ship is not allowed interrupt the operation and wait. When a ship has finished all operating activities in a port, it must immediately sail to its next destination port. Hence, the ship is not allowed to wait prior to departure for the next port. Figure 2.1 shows an example of a routing solution for two ships.

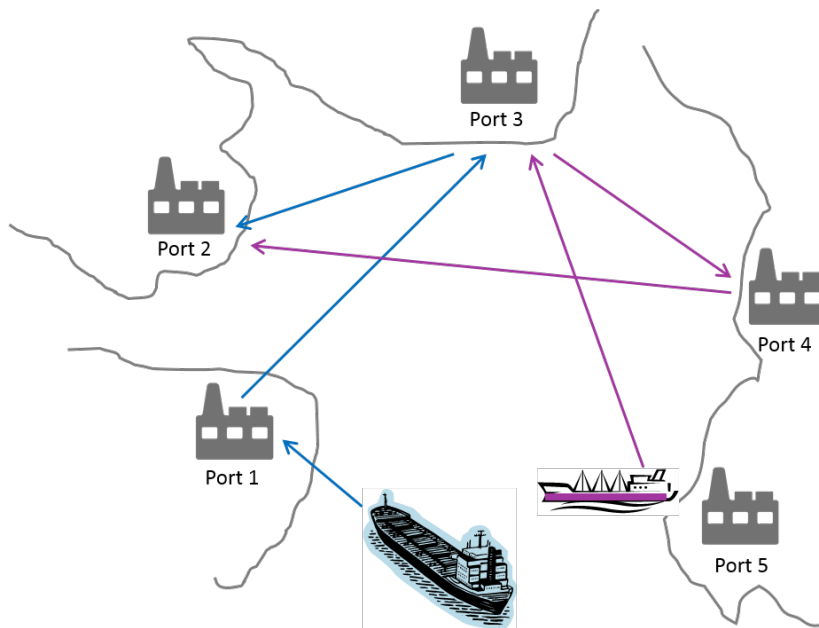


Figure 2.1: Example of the routing of two ships between five ports

Heterogeneous ships. The fleet consists of a set of ships that may differ with respect to the following characteristics:

- Size
- Speed
- Cost structure
- Set of compartments and their capacities

Allocation. The ships can carry a selection of products, possibly all. In addition, each ship has a given number of undedicated compartments where products can be allocated. The compartments can vary in size and each have a maximum capacity. The different products that are transported cannot be mixed, and thus a compartment can only contain one product at a time. The capacity of a compartment in a ship is often large compared to the quantity loaded or unloaded in a given port, and hence partial loading and unloading is allowed. This implies that unloading a product from a compartment does not necessarily mean that the compartment is emptied. Similarly, if a compartment has available capacity it can be loaded with more of the same product that it currently contains. However, if a compartment is emptied at a port, any product can now be loaded into the compartment. Allocation of products, loading/unloading in port and the possibility of partial unloading is illustrated in Figure 2.2.

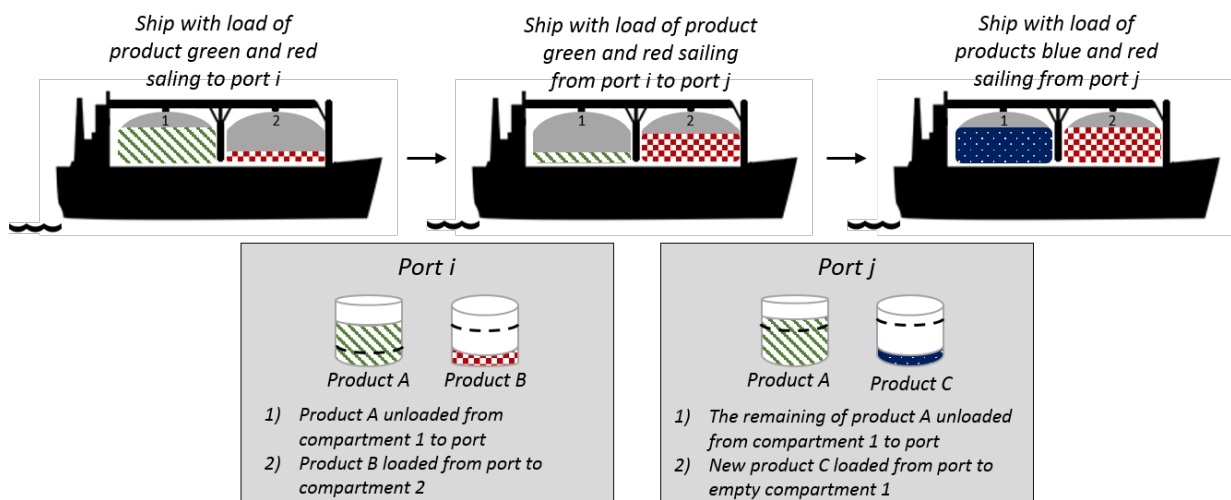


Figure 2.2: Example of the allocation of three products to a ship with two compartments

When a product is loaded into a compartment, it continues to stay in that compartment during sailing and waiting outside ports, until it is unloaded in a different port. This means that no reallocation of products between compartments can take place between operating times in the ports. In a port, however, a product can be reallocated by first unloading the product from the ship to the port and then loading it back on the ship into a different compartment, given that the port has a storage for the product.

Loading. At the start of each schedule, the initial load of a product in every compartment in each ship is known. All loading and unloading operations take place when a ship operates in a port. All ships and ports have a maximum loading capacity. When a ship visits a port, the binding loading capacity under operation is the lowest of the ship's capacity and the port's capacity.

Chapter 3

Related Literature

Both the industry and academia interest for the use of DSS in MIRPs have increased in the later years, possibly due to the proved possibilities of monetary benefits (Christiansen et al., 2013). As a result, several papers on MIRPs with similarities to the problem introduced in this thesis have been published. The purpose of this chapter is to first provide an overview of the literature related to the formulation of MIRPs. Following is a presentation of relevant literature on specialized solution methods for MIRPs and problems of similar structure.

In Section 3.1, a selection of papers on MIRPs is presented. The papers are classified according to the dimensions of a MIRP that we consider most essential. Many of these dimensions are also used in the presentation of the literature on general inventory routing problems in Andersson et al. (2010). However, additional dimensions have been added here to better reflect the problem depicted in this thesis. The classification is introduced to help facilitate an organized discussion around the similarities and differences of the different models and applications presented in the papers. Following is Section 3.2, included with the intention of presenting the most important and relevant related literature for this thesis in greater detail. Here, the handling of multiple products and the allocation of products are in focus. Lastly, Section 3.3 presents literature relevant for the development of solution methods for MIRPs. A relevant classification scheme for matheuristics, as first presented by Archetti and Speranza (2014), is also provided.

The increase in interest for MIRPs has resulted in an increase in the number of publications in the field. Over the years, several literature studies have been conducted to present and organize the available literature. Recently, two relatively exhaustive surveys have been conducted on literature in the area of maritime transportation optimization, by Christiansen et al. (2013) and Andersson et al. (2010). The former presents relevant literature and provides four basic models to illustrate different dimensions of the maritime transportation domain. Andersson et al. (2010) focus on the industrial aspects of combined inventory management and routing in both maritime and road-based transportation. Both surveys are conducted with the purpose of giving a comprehensive literature review of the current state of the research.

This literature review does not have the intention of being an exhaustive and comprehensive presentation of all relevant literature. Its aim is to review a representative selection of papers depicting general MIRPs as well as papers directly related to our problem and corresponding solution method.

3.1 Maritime inventory routing problems

A MIRP is essentially a combination of a routing problem and an inventory management problem. The flexibility when designing a MIRP is great. According to Andersson et al. (2010), MIRPs have no predefined clear definitions or well-defined assumptions. As a result, each new paper published, consider a new version of a MIRP. This, in addition to the inclusion of industrial aspects and specific cases, opens for a large variance in the underlying problems, problem formulations and the modelling decisions that are made. Consequently, there are great difficulties associated with the classification of these papers. We consider the following dimensions to be the most important when classifying our selection of papers:

- length of the planning horizon
- time scheme used
- the structure of the distribution network
- fleet characteristics
- number of products handled
- handling of multiple products
- handling of inventory
- nature of production/consumption rates

Table 3.1 shows the classification of the selected papers with respect to the defined dimensions. The table is not exhaustive and problems similar to the ones included are left out. The papers presented in the table are selected on the basis of two reasons; first, coverage of the selected dimensions, and second, well cited and known publications in its field. Even though the problem presented in this thesis is not directly linked to a specific industrial application, we have included both papers with real life industrial applications and papers with a theoretical aim to give a representative overview of the MIRP literature.

Planning horizon

MIRPs are normally considered to be tactical problems, however, loading operations to and from a ship are considered to be operational decisions (Christiansen et al., 2007). The length of the planning horizon depends on the level of planning. A few weeks would be considered a normal length planning horizon for tactical/operational problems such as

MIRPs. In this thesis, we have classified a short planning horizon as approximately four weeks, a medium horizon as a few months and a long horizon as one year or longer. Differences in the planning horizon originate from the structure of the specific problem solved. Long sailing legs normally implicate a longer planning horizon. Most of the papers listed in Table 3.1 address short sea inventory problems with short sailing legs compared to time used in port. As can be seen, different lengths of planning horizons have been chosen for the different problems, however all lie in the short-medium range. Hemmati et al. (2016) and Agra et al. (2014) solve the planning problem with a time horizon ranging from one to two months and several months respectively. Al-Khayyal and Hwang (2007) on the other hand, consider a very short time horizon of only 10 days due to the focus on operations in port.

Paper	Horizon	Time	Dist. network	Fleet	Products	Allocation	Inventory	Prod/con rate
Our problem	Short	Discrete	Many-to-many	Hetero.	Multi.	Undedicated comp.	Min-max stock levels	Varying
Al-Khayyal and Hwang (2007)	Short	Continuous	Many-to-many	Hetero.	Multi.	Dedicated comp.	Min-max stock levels	Fixed
Agra et al. (2013)	Short	Discrete	Many-to-many	Hetero.	Sing.	N/A	Min-max stock levels	Varying
Christiansen (1999)	Short	Continuous	Many-to-many	Hetero.	Sing.	N/A	Min-max stock levels	Fixed
Siswanto et al. (2011)	Short/ medium	Continuous	Many-to-many	Hetero.	Multi.	Undedicated comp.	Min-max stock levels	Fixed
Hemmati et al. (2016)	Medium	Continuous	Many-to-many	Hetero.	Multi.	No comp.	Min-max stock levels	Fixed
Agra et al. (2014)	Medium	Continuous	Many-to-many	Hetero.	Multi.	Dedicated comp.	Min-max stock levels	Fixed
Ronen (2002)	Short	Discrete	One-to-one	Hetero.	Multi.	No comp.	Safety stock	Varying
Persson and Göthe-Lundgren (2005)	Short/ Medium	Discrete	Many-to-many	Hetero.	Multi.	No comp.	Soft limits	Variable
Dauzère-Pérés et al. (2007)	Short	Discrete	One-to-many	Hetero.	Multi.	No comp.	Min-max stock levels	Varying

Time

MIRPs are a combination of ship routing, scheduling, and inventory management. With ship scheduling, the aspect of time is introduced. Time can either be treated as discrete or continuous. With a discrete time scheme, the aspect of time is modelled with time periods and an index representing time is included on all relevant variables and parameters. A continuous time scheme, on the other hand, represents time with a continuous decision variable. Both Agra et al. (2013) and Ronen (2002) employ varying production and consumption rates, a complicating factor overcome with the use of a discrete time scheme. A continuous time scheme is chosen by e.g. Christiansen (1999), Al-Khayyal and Hwang (2007) and Siswanto et al. (2011), all of which consider the production and/or consumption rates to be fixed during the planning horizon.

Distribution network

The transportation of products between ports can be done with either a one-to-one, one-to-many or many-to-many distribution network. In Ronen (2002), bulk products are routed with a one-to-one distribution network where a vessels voyage has a single loading port and a single discharging port. Dauzère-Pérès et al. (2007) have a one-to-many distribution network as the problem describes the transportation of products from one processing plant (depot) to several consumption ports. As opposed to road-based transportation, there is usually no central facility in maritime transportation. As a result, a many-to-many distribution network is the most common structure. Hence, ships can load and unload in any number of ports, and there is no fixed start- or end-point to each route (Andersson et al., 2010). A many-to-many distribution can be found in Christiansen (1999) where ammonia is transported between multiple production- and consumption harbours, as well as in Al-Khayyal and Hwang (2007) and Agra et al. (2013), among others.

Fleet

The fleet used to distribute the products can either be homogeneous or heterogeneous. In road-based transportation problems, the fleet of vehicles often consists of identical, homogeneous vehicles. However, in maritime transportation, the ships usually differ on a number of characteristics making the fleet heterogeneous. In vehicle routing problems (VRPs) the products are often transported as cargoes that can easily be stocked together. This is not necessarily the case in maritime transportation problems where the products handled often need to be compartmentalized or carried in tanks. Bulk products can impose special requirements on the compartments of the ships carrying these products. All of the example literature included in Table 3.1, have a fleet of heterogeneous ships. The ships can vary in size, speed, compartment structure, costs etc.

Products

A MIRP considers the transportation of either a single product or multiple products. Multiple products are in particular present in the transportation of (liquid) bulk products. Both Agra et al. (2013) and Christiansen (1999) consider the transportation of a single product. Ronen (2002) was the first to introduce the transportation of multiple products. Since then several multi-product MIRPs have been modelled. For example Al-Khayyal and Hwang (2007) modifies the model from Christiansen (1999) to account for multiple products.

Allocation

When a MIRP considers the distribution of multiple products, the problem can be expanded to include the allocation of products. Due to the nature of the products that are

shipped, they often cannot be mixed and must be allocated to different compartments. Up until now, little research has addressed this issue. Ronen (2002), Persson and Göthe-Lundgren (2005), Dauzère-Pérès et al. (2007) and Hemmati et al. (2016) all have models with multiple products, but disregard the allocation of products into compartments on board the ship. Al-Khayyal and Hwang (2007) consider the pick-up and delivery of multiple products using ships with multiple dedicated compartments. Agra et al. (2014) also assume that products must be dedicated to the different tanks of the ship. Siswanto et al. (2011) relaxes the problem from Al-Khayyal and Hwang (2007) to consider an assignment of multi-undedicated compartments to products. Undedicated compartments is also utilized by Christiansen et al. (2011).

Inventory

In maritime inventory routing, problems must be solved without violating inventory and storage limitations in each port. Ronen (2002) faces uncertainties in demand and production and must maintain safety stocks of each product at the origins and destinations. Persson and Göthe-Lundgren (2005) consider the shipment planning of bitumen from three refineries to a number of depots. The refineries have a lower safety stock level and a maximum inventory level for each product, while the depots are modelled with soft inventory limits where a deviation from the target inventory level is penalized with a cost. Siswanto et al. (2011) and Hemmati et al. (2016) among others, use hard minimum and maximum inventory limits for all products at all origins and destinations.

Production and consumption rate

The production and consumption rates at the origin and destination ports can have three states, namely fixed, varying or variable. With a fixed production/consumption rate, the rate is equal during the planning horizon. Opposite, with varying production/consumption rate, the ports have rates that vary with time. Variable production implies the inclusion of a new decision dimension, i.e. the production/consumption level is modelled as a decision variable. Whether the production/consumption rate is fixed or varying is usually correlated with time scheme used. With a continuous time scheme, production/consumption rates are normally given as a parameter and is fixed over the planning horizon. Christiansen (1999) and Siswanto et al. (2011), as well as Hemmati et al. (2016) and Agra et al. (2014) present fixed production/consumption rates combined with continuous time. A discrete time scheme enables a varying production/consumption rate over the planning period by keeping the rates constant in each time period. Ronen (2002) assumes varying production, while Dauzère-Pérès et al. (2007) assume consumption varying with time. Persson and Göthe-Lundgren (2005) employ a variable production level. The production decision is part of the model and the problem is thus composed of a generation of both shipment plans and processing schedules.

3.2 MIRPs with multiple products and allocation

In this section we look at how the papers essential for the development of this thesis handle multiple products and the allocation problem. Hence, the focus of this section is only the papers that consider multiple products and propose a method of handling the allocation problem. Consequently, all papers avoiding the allocation problem are not included in this section.

The incorporation of multiple products in the traditional inventory routing problem introduces new challenges. Which ports handle which products, which products can be handled simultaneously and which ships can transport which products are just some of the considerations that need to be reviewed. Similarly to the problem proposed in this thesis, Hemmati et al. (2016) have chosen not to enforce any restrictions on the combinations of products and ports, i.e. all products can be both consumed and produced in any ports. Al-Khayyal and Hwang (2007), Siswanto et al. (2011) and Agra et al. (2014) on the other hand have a set of production ports and a set of consumption ports which in a greater degree limit the flow of products. In this way, they remove the challenges of modelling both loading and unloading activities in the same port, reducing the problem size. In both Hemmati et al. (2016) and Al-Khayyal and Hwang (2007), ships are allowed to load/unload different products simultaneously in the same port, but handling the same products by different ships in the same ports simultaneously is not allowed. Agra et al. (2014) have solved this issue by restricting the ports to only have one ship operating at a time and thus removing the possibility of different ships handling the same product simultaneously in the same port. Siswanto et al. (2011) impose that each ship can only handle one product at a time. Sequential handling and no parallel handling of products for a ship reduce the size of the problem. Further on, challenges with multiple products in MIRPs can be found in pulp industry (Andersson, 2011) and (Bredstrom et al., 2005), grain industry (Bilgen & Ozkarahan, 2007), liquid bulk industry (Ronen, 2002) and calcium carbonate industry (Dauzère-Pérès et al., 2007).

A natural extension to the introduction of multiple products is the issue with how these products are to be loaded on board the ship. In the context of multi-product MIRPs, it is normal to consider bulk products that need to be transported in compartments due to their unmixable nature. Both Agra et al. (2014) and Al-Khayyal and Hwang (2007) as well as Li et al. (2010) assume the products to be unmixable and are thus forced to address the problem of allocating products to compartments. They solve the problem by defining each compartment to be dedicated to a specific product, introducing a limitation on which products that can be carried by each compartment of a ship. This implies that reallocation of the products into other compartments is impossible. The use of dedicated compartments is the most used method of solving the allocation problem. Siswanto et al. (2011) and Christiansen et al. (2011) on the other hand, introduce the use of undedicated compartments. The paper by Siswanto et al. (2011) defines an undedicated compartment to be a compartment that can take any product, but it can only store one product at a time. Introducing undedicated compartments relaxes the models utilizing dedicated com-

partments as a result of a greater flexibility in the allocation of products. In the event of an empty compartment, any product can be loaded to that compartment. Siswanto et al. (2011) have imposed a limitation that all compartments in the ship must be emptied before returning to a production port to be reloaded and thus the danger of mixing products in the same compartment during the shipment is removed. There still does not exist much literature on the allocation problem and the papers mentioned here are about the only ones addressing the problem. In Hemmati et al. (2016), even though the dimension of multiple products is included, the allocation into compartments is disregarded. The lack of compartments in the ships implies that the only capacity constraint that needs to be respected is the total ship capacity. This approach is also used by Ronen (2002), Dauzère-Pérès et al. (2007) and Persson and Göthe-Lundgren (2005) among others.

3.3 Matheuristics as a solution approach

In this section we present and review literature that consider matheuristics as a solution approach to inventory routing problems (IRPs).

The class of IRPs includes optimization problems that differ on numerous characteristics, but they all simultaneously consider a routing and an inventory management component of an optimization problem (Berttazzi & Speranza, 2012). VRPs, that consider only the routing part of IRPs, are known to be computationally very hard to solve. IRPs are known to be even harder to solve (Berttazzi & Speranza, 2012). The current leading methodology for solving VRPs with an exact algorithm is a branch-and-price-and-cut (Archetti, 2014). For IRPs, there have been few attempts to find the optimal solution and mostly heuristic algorithms have been presented in the literature (Berttazzi & Speranza, 2012).

Both classical heuristics and metaheuristics have shown good results for solving VRPs, but they are often over-engineered to the extent that the flexibility is reduced (Archetti, 2014). For a long time, the design of exact and heuristic algorithms have proceeded in parallel. However, in recent years a new class of heuristics have emerged with increasing popularity, so-called matheuristics. Matheuristics are heuristic algorithms which embed the optimal solution of mathematical programming models, typically mixed integer linear problems MILPs (Archetti, 2014). The interest for matheuristics has been driven partly by the improvements in computational effectiveness of exact commercial solvers and the availability of such software (Berttazzi & Speranza, 2012). It is expected that matheuristics will become increasingly popular and successful as a solution method. By both making use of mathematical programming formulations and exploiting the specific structure of a problem, matheuristics bring us closer to the goal of solving large-sized instances close to optimality (Archetti & Speranza, 2014).

Even though there has been a significant increase in the number of papers devoted to matheuristics for solving routing problems the last decade, the available literature is still limited. In Archetti and Speranza (2014), an overview of literature on matheuristics pro-

posed to solve routing problems is given. In our literature review we have included all of the papers listed in this survey that consider IRPs and that we consider relevant. The papers presented include both land-based routing problems and maritime ship routing problems. We have also included papers not presented in the survey which we regard as relevant for our thesis, some of which are published after the publication of the survey.

3.3.1 Classification of matheuristics

In Archetti and Speranza (2014), the proposed literature is classified in three different classes; decomposition approaches, improvement heuristics, and column generation based approaches. The recent literature mainly focus on improvement heuristics and column generation based approaches. However, the survey shows that the decomposition approaches are still popular for complex problems such as IRPs.

Decomposition approach

The underlying idea of the decomposition approach is to divide the problem into smaller and favorably simpler problems such that each subproblem can be solved using an exact solution method (Archetti & Speranza, 2014). In order for it to be called a matheuristic, at least one subproblem must be solved with a mixed integer linear programming (MILP). This approach is particularly suitable for integrated problems such as IRPs. In IRPs, it is natural to decompose the different decision components. According to Archetti and Speranza (2014), decomposition approaches have also proved to behave better than other heuristic algorithms for IRPs. The class of decomposition approaches is further divided into the following subclasses by Archetti and Speranza (2014).

- Cluster-first route-second. Divides the two main decisions. First customers are assigned to vehicles according to clusters. Subsequently, the sequence of the customers for each vehicle is decided. The latter part of the problem is typically solved using heuristics.
- Two-phase approaches. The problem is decomposed into two phases where each phase is solved separately. Includes all two-phase matheuristics, except from the cluster-first route-second scheme.
- Partial optimization approaches. MILPs are used to handle a part of the problem otherwise handled by heuristics. Routing decisions are typically handled by heuristics, while the quantity decisions are solved using MILPs.
- Rolling horizon approaches. The decisions must be taken over a period of time and the time horizon is divided into smaller periods.

Improvement heuristics

In the class of improvement heuristics, the matheuristics use a MILP to improve a solution found by a different heuristic approach (Archetti & Speranza, 2014). This approach is popular due to the fact that the MILP can be used to improve any solution no matter

which heuristic is implemented. Archetti (2014) further classify the improvement heuristics approach into the following two categories.

- One-shot approach. The MILP is used as a final improvement procedure and is solved once after a solution is obtained by a heuristic. This approach is typically preferred when the MILP is difficult and complex to solve.
- Iterated approach. The MILP is integrated in the heuristic scheme and is used as an operator inside a searching procedure.

Column generation based approach

In the column generation based approaches, branch-and-price and/or column generation is used to build heuristic solutions. Most of these approaches use heuristic schemes to generate columns. The success of exact algorithms for branch-and-price has made this approach more and more popular in recent years.

Classification of the included literature

In Berttazzi and Speranza (2012), the authors present different classes of matheuristics for solving inventory routing problems. They separate the different matheuristics by how the mathematical model is embedded in the heuristic scheme; by solving sub-problems, solving parts of an instance, restricting the search space or exploring neighborhoods.

Evidently, there are numerous opportunities of integrating MILPs with heuristics. In the following presentation of literature related to the thesis problem, we use the three classes presented above as well as the subclasses proposed by Archetti and Speranza (2014) to classify the included literature. Table 3.1 presents all the papers included in this literature review, classified after type of scheme. It should be noted that we have chosen to classify each paper in only one subclass, the one we find most fitting, even though the research presented might include components from multiple subclasses. This becomes evident when we review the literature in greater detail.

3.3.2 Applications

Decomposition approaches

Cluster first - route second. Campbell and Savelsbergh (2004) consider an IRP with a long time horizon. A cluster first- route second approach is the chosen framework for the design of the matheuristic. The problem is decomposed into two phases due to the long time horizon. Initially in the first phase, the customers are clustered by geographical location. This is a measure to reduce the size of the problem. However, the main focus of the first phase is the decision on which day to serve which cluster of customers, along with a suggestion on delivery quantities. These decisions are made by solving a MIP over the full planning horizon. In the second phase, the time horizon is shortened and vehicle routes are constructed using an insertion construction heuristic. The underlying idea is

Table 3.1: Classification of literature considering matheuristics

Paper	Decomposition				Improvement		Col. gen.
	Cluster-first route-second	Two- phase	Partial opt.	Rolling horizon	One Shot	Iterative	
Campbell and Savelsbergh (2004)	✓						
Guerrero et al. (2013)	✓						
Yu et al. (2008)		✓					
Halvorsen-Weare and Fagerholt (2013)		✓					
Coelho et al. (2011)			✓				
Coelho et al. (2012)			✓				
Cóccola and Méndez (2015)			✓				
Rakke et al. (2011)				✓			
Agra et al. (2014)				✓			
Savelsbergh and Song (2008)					✓		
Stålhane et al. (2012)					✓		
Archetti et al. (2013)					✓		
Song and Furman (2013)					✓		
Agra et al. (2016)					✓		
Archetti et al. (2012)						✓	
Maraš et al. (2013)						✓	
Hemmati et al. (2016)						✓	
Aghezzaf et al. (2006)							✓
Raa and Aghezzaf (2008)							✓
Raa and Aghezzaf (2009)							✓

that the delivery quantities and time specified in the first phase are always good in a long-time perspective, but not necessarily in the short term. Thus, these decisions are treated as suggestions when finding a solution in phase two (Campbell & Savelsbergh, 2004). A rolling horizon framework is used and the solutions from the second phase is used to update the input of the first phase in the next iteration.

Like Campbell and Savelsbergh (2004), Guerrero et al. (2013) employ a cluster first-route second approach. However, Guerrero et al. (2013) consider a more complex type of IRP called inventory location routing problem (ILRP). An ILRP can be described as a multiple supplier IRP combined with supplier plant location decisions. Thus, the objective is to minimize the standard IRP costs, as well as location costs from the location decision. Similarly to the first phase strategy proposed by Campbell and Savelsbergh (2004), a MILP formulation is used to determine the distribution plan. In Guerrero et al. (2013) however, the first phase also determines the location of the supplier plants. Following is the second phase where a randomized routing heuristic is implemented. The second phase optimizes the routing decisions attempts to improve the location decisions, similarly to Campbell and Savelsbergh (2004). This is done through a local search followed by an intensification procedure that evaluates the interaction between the inventory and routing decisions made in the different phases.

Two-phase. Yu et al. (2008) consider an IRP with split deliveries and propose an approximate model to quickly find near-optimal solutions to the problem. The model is solved

with a Lagrangian relaxation decomposition scheme which decompose the problem into an inventory subproblem and a routing subproblem. The inventory subproblem is solved using a MILP formulation, while the routing subproblem is solved with a minimum cost flow algorithm. Based on the solution of the subproblems, a feasible solution to the original model can be constructed by solving a series of assignment problems. The constructed solution is further improved by a local search heuristic, and like Campbell and Savelsbergh (2004), a rolling horizon framework is incorporated.

Similar as Yu et al. (2008), Halvorsen-Weare and Fagerholt (2013) decompose their problem into two phases, but here in a routing-first, inventory-second scheme. Halvorsen-Weare and Fagerholt (2013) consider an LNG problem where the goal is to make an annual delivery program (ADP). By first disregarding the berth and inventory constraints, they decompose the problem into one or more routing problems and a feasibility scheduling problem. A multi-start local search heuristic is used to solve the routing subproblems. Subsequently, the feasibility problem is solved with a MIP. It can be noted that a distinguishing factor between Halvorsen-Weare and Fagerholt (2013) and Yu et al. (2008) is the sequence in decisions made. While Yu et al. (2008) make the inventory decision first, Halvorsen-Weare and Fagerholt (2013) turn it around and make the routing decision first.

Partial optimization. Coelho et al. (2011) use an optimization based adaptive large neighborhood search heuristic (ALNS) to solve an IRP with transshipment. In each iteration of the ALNS, heuristics are used to explore the neighborhood and manipulate the vehicle routes. A partial optimization scheme is used by iteratively fixing the routing decisions and optimizing the remaining part of the problem. The remaining problem of determining delivery quantities and transshipment moves is solved through a network flow algorithm. A similar approach is used by the same authors in Coelho et al. (2012) to solve a multi-vehicle IRP with consistency requirements. Here, two subproblems are solved exactly. The first subproblem, Delivery Quantities (DQ), optimizes the delivery quantities and it is solved every time a new routing solution is computed by the ALNS. The second subproblem is called Solution Improvement (SI) and is solved every θ iterations or whenever a new best solution is found. The iterative scheme proposed, where restricted parts of the problem are optimized in turn, is in line with the definition of partial optimization.

Cóccola and Méndez (2015) introduce a new MILP-based approach for handling large-scale ship routing and scheduling problems. Note that this paper only consider routing decisions, however it included due to its relevance for the thesis problem. Like Coelho et al. (2011, 2012), a partial optimization approach is used, solving smaller sub problems in an iterative solution procedure. However, Cóccola and Méndez (2015) propose a different, systematic decomposition strategy. At first, the solution approach is decomposed by individually scheduling each ship tour. To gradually build a feasible solution, highly constrained versions of the model are solved by fixing a subset of the binary variables. In this way, the number of decisions is maintained at a reasonable level, and computational efficiency of the MILP branch and bound solution procedures can be improved. The procedure terminates when all ship schedules are feasible.

Rolling Horizon. Rakke et al. (2011) propose a rolling horizon heuristic (RHH) for creating an ADP to solve an LNG problem. The RHH solves the problem by iteratively solving the MIP for shorter sub-horizons using branch-and-bound. Each shortened time interval is divided into a frozen period, a central period and a forecast period. The variables in the frozen period are fixed to the values they were assigned in the central period of the previous time interval. A requirement on integrality is enforced in the central period, but relaxed in the forecast period. The ADP is then further improved by applying an improvement heuristic on a modified version of the mathematical problem. By using the feasible ADP as a starting point, the number of variables in the formulation can be limited by fixing some values and, hence, focus the search effort to a more restricted area of the solution space. This part of the solution process corresponds to the policy-based matheuristics mentioned by Bertazzi and Speranza (2012) which restrict the search space according to simplification policies.

Agra et al. (2014) present a matheuristic composed of different known approaches used when solving MIRPs. As Rakke et al. (2011), Agra et al. (2014) utilize the rolling horizon scheme to decompose the problem into shorter sub-horizons. Agra et al. (2014) also propose local branching and a feasibility pump procedure. Local branching is included in the matheuristic to improve a given feasible solution by searching for local optimum when restricting the number of variables that can change its value. The feasibility pump procedure is a scheme to find an initial feasible solution and it is based on the iterative rounding of fractional variables. The authors have included this procedure to more efficiently find an initial feasible solution.

Improvement heuristics

One-shot improvement. Savelsbergh and Song (2008), Song and Furman (2013), Stålhane et al. (2012), Archetti et al. (2013) and Agra et al. (2016) all utilize a one-shot approach in their proposed improvement matheuristics. Savelsbergh and Song (2008) use a randomized greedy heuristic (RGH) to produce a complete schedule for their IRP with continuous moves. They also propose an integer programming based optimization algorithm to improve portions of that schedule. Thus, following the RGH is an improvement phase that optimizes the schedules of a subset of the vehicles while keeping the schedules of the remaining vehicles fixed. This is done iteratively, each time optimizing the schedules for different vehicles. This can be viewed as a neighborhood search scheme that relies on integer programming to explore neighborhoods. This idea is further developed by Song and Furman (2013). They propose a similar optimization-based large neighborhood search procedure to solve a MIRP. After obtaining a feasible initial solution using a branch-and-cut algorithm, small sub-problems are constructed by fixing some values. The sub-problems can be characterized as MIPs and are now solved separately to improve the initial solution. This process is continued until no further improvements can be found.

Stålhane et al. (2012) study the same problem as Rakke et al. (2011), namely to create ADPs for an LNG inventory routing problem. The matheuristic consists of a construction and improvement heuristic (CIH), followed by an intensification phase. The CIH is a multi-start, first-descent, local search heuristic that by greedy insertion constructs a set of solutions. The intensification phase is the one-shot improvement strategy where the authors use a MILP. A subset of the variables in the original problem is fixed to the value found by the construction heuristic, and a branch-and-bound algorithm is used to optimize the value of the free variables. Archetti et al. (2013) also propose a matheuristic where the scheme is finalized with solving a MILP model using branch-and-bound to improve the solution. The problem considers the delivery of free newspapers from a production plant to subway, bus or tram stations. The problem is decomposed into two phases, and is thus also an example of the two-phase decomposition scheme discussed earlier. A MILP model is used to create a delivery plan in the first phase, while a heuristic constructs the routes in the second phase. The improvement phase, where a MILP is utilized, aims to reduce the number of trips by changing delivery quantities from the first phase.

Agra et al. (2016) consider a MIRP for the salmon farming industry and propose a method for improving the efficiency of the branch-and-bound algorithm used on the MIRP. This is done by a one-shot approach where the branch-and-cut is run for a given time, and then algorithms are used to either improve the feasible solution that is obtained or build a feasible solution if none is obtained in the branch-and-cut algorithm.

Iterative improvement. Hemmati et al. (2016), Archetti et al. (2012), and Maraš et al. (2013) use an iterative improvement heuristic approach. Hemmati et al. (2016) propose a two-phase heuristic to determine routes and schedules for a shipping company. By transforming the inventories into sets of cargoes, they convert the cargo and inventory routing problem into a pure cargo routing problem. The improvement heuristic iterates between solving the cargo routing problem for a given set of cargo sizes and updating the cargo sizes to obtain a new cargo routing problem. Archetti et al. (2012), on the other hand, propose a heuristic, called HAIR, that combines a tabu search scheme with an intensification technique consisting of solving a sequence of MILPs. As an iterative improvement heuristics approach, the intensification is applied every time the current solution is improved in order to improve it further. Maraš et al. (2013) propose three different matheuristics for exploring the neighborhood. The matheuristics are based on a local and variable neighborhood branching technique, and a variable neighborhood decomposition search, respectively.

Column generation based approach

Aghezzaf et al. (2006), Raa and Aghezzaf (2008, 2009) all use a column generation based approach for solving long term IRPs. All three papers differ from the other literature included in that they consider an IRP with deterministic and constant customer demand rates. Hence, they are defined as cyclical distribution problems. Therefore, the solutions of the problems can be given in the form of long-term cyclical distribution patterns for the

vehicles. The concept of multi-tours is introduced, which is distribution patterns consisting of vehicles performing multiple tours with possibly different frequencies. The same optimal cycle is repeated for a long period and the solution is a repetitive multi-tour for each vehicle.

Aghezzaf et al. (2006) propose a formulation where binary variables are used to represent vehicle routes. A column generation based approximation method is developed which utilizes a heuristic algorithm to generate columns (multi-tours). In Raa and Aghezzaf (2008) and Raa and Aghezzaf (2009) an extension of this solution approach is applied, also exploiting the concept of multi-tours and integrating several heuristic procedures within a column generation framework.

Chapter 4

Model Description and Valid Inequalities

In this chapter, we present the model of this thesis. In Section 4.1, we introduce all assumptions necessary for the design of the model. Section 4.2 presents the model formulation. The following section, Section 4.3, introduces the proposed valid inequalities.

4.1 Model assumptions

When modelling the multi-product MIRP-UC, a series of modelling choices and assumptions have to be considered. The first choice to be made for any optimization problem containing some aspect of time is whether to use a time-discrete or time-continuous scheme. The scheme should be chosen based on the characteristics of the problem to be modelled. We have chosen to use a time-discrete formulation in this thesis to overcome the complexity of varying production and consumption. The time is discretized into a number of uniform time periods over the planning horizon. When using a time-discrete formulation, it is important to consider the balance between the length of the planning horizon and the size of the time periods to both achieve the desired detail level and maintain a tolerable size on the problem.

In this thesis, a schedule consists of a geographical route, a sequence of ports, together with the time periods of when the ship waits and operates in that port (Agra et al., 2013). Each ship only has one schedule during the planning horizon. This implies that if the schedule of a ship ends prior the end of the planning horizon the ship cannot be utilized later in that planning horizon.

Each ship is assumed to have three types of modes; sailing, waiting, and operating. With a time-discrete formulation, sailing time, waiting time, and operating time can be expressed as an integer multiple of a time period. The number of time periods needed for each ship to sail between two ports is assumed to be known and is used as input to the model. In a schedule, there is no limitation on the number of times a ship can visit each port. An example of a ship route in the discrete time frame is presented in Figure 4.1

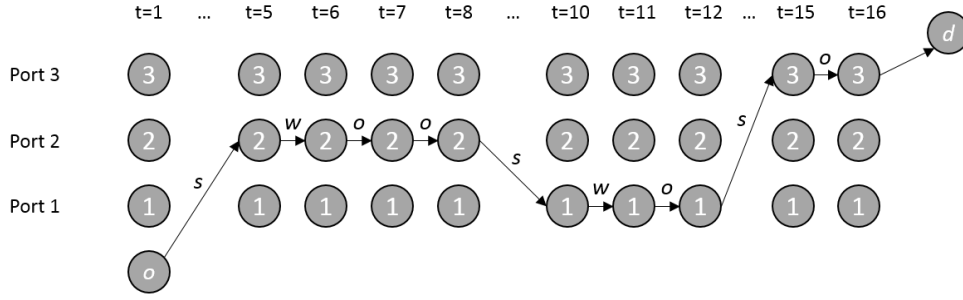


Figure 4.1: Example of a ship route consisting of sailing s , waiting w , and operating o

Figure 4.1 shows the routing of a ship between three ports. In time period one, $t = 1$, the ship sails directly from its origin node, o , to *Port 2*, where it arrives in time period five. In *Port 2*, the ship waits for one time period and then it operates in two time periods before it sails to *Port 1* in time period eight. The ship arrives in *Port 1* in time period ten, waits in time period 11 and operates in time period 12. Finally, the ship sails to *Port 3* and operates in one time period before the schedule ends in time period 16 when the ship sails to the destination node, d . Note that waiting is only done prior to operating in each port, as specified in the problem description.

Each ship has an initial starting position either in a port or a point at sea. The initial position, denoted $o(v)$, is known and is the starting point of the schedule of ship v . If a ship's initial position is in a port, the ship can start its schedule with operation, waiting or sailing. If the ship is located at a point at sea however, it must immediately sail to a port. The end of a schedule is modelled with an artificial ending node, called $d(v)$.

Since we are using a time-discrete model, each ship has a sailing-, waiting-, and operating-cost defined as a unit cost per time period used on an activity. We are also using an additional operation cost defined to be a unit cost per quantity loaded/unloaded in a port. The fixed operation cost per time period ensures that a ship minimizes the time used in a port, while the variable operation cost makes sure that a ship does not load/unload more of a product than what is necessary.

All production- and consumption quantities are assumed to be deterministic. Even though the production- and consumption quantities can vary over the planning horizon, they are assumed constant in each time period for each product. In each port, as long as the number of ships berthed does not exceed berth capacity, we assume that all ships can operate simultaneously. Hence, equipment and manpower limitations in the port are accounted for in the berth capacity measure.

Since we have assumed no technical limitations in a port, the only factor restricting the quantity loaded/unloaded in a port is the loading/unloading capacity to the ship and port. In other words, each ship can load/unload as much of one product or multiple products within one time period as long as it does not exceed the loading/unloading capacity. We also assume that no costs are associated with the time used on switching between loading/unloading different products because we consider the switching time to be insignificant

compared to the length of a time period.

Even though the assumption of unmixable products is at the core of this problem, the cost associated with time necessary to wash and clean a compartment during the switching of products in a compartment is not taken into account. We assume that the washing of compartments can be done without a decisive cost compared to the other cost components due to the similar nature of the products. Hence, extensive cleaning is not necessary and the time used on washing is insignificant compared to the length of a time period.

In general, MIRPs are considered continuous planning problems without a predetermined ending. To be able to model these problems however, the ships are assigned schedules limited to a finite planning horizon. Thus, it is natural to assume that end conditions must be imposed to ensure that the solution over the finite planning horizon adheres with the solution over the infinite planning horizon. However, such problems are often re-optimized prior to the end of the planning horizon to enable a rolling horizon that models the continuous operations of the fleet. Considering this, no predetermined end conditions on the inventory and loading variables are necessarily needed. Introducing end conditions can also bias the results considering that we have a relatively short planning horizon. Due to these arguments we have chosen not to include any end conditions on the inventory variables. When considering the routing variables, we assume that a ship's schedule can end, i.e. it can travel to $d(v)$, at any time during the planning horizon without having to empty its compartments. Potential costs associated with an early finish of a ship's schedule is not included in the model. Hence, we have not included any end conditions for the loading variables.

4.2 Multi-product MIRP-UC formulation

The following section presents the mixed integer linear program that model the multi-product MIRP-UC in this thesis. The model is originally based upon the work of Agra et al. (2013), but significant modifications have been made to account for multiple products and undedicated compartments. It is also the continuation of the work conducted on the model in Foss et al. (2016).

Sets

\mathcal{V}	Set of all ships
\mathcal{K}	Set of all products
\mathcal{K}_v	Set of all products carried by ship v
\mathcal{C}_v	Set of all compartments in ship v
\mathcal{N}	Set of all ports
\mathcal{T}	Set of all time periods

Indices

i, j	ports
v	ships
c	compartments
k	products
t	time periods
$o(v)$	origin node of ship v
$d(v)$	artificial destination node of ship v

Parameters

C_{ijv}^T	Cost of sailing from port i to port j with ship v
C_v^W	Cost of waiting outside a port per time period for ship v
C_{iv}^O	Cost of operating in port i per time period for ship v
C_{ivk}^Q	Cost of loading/unloading one unit of product k for ship v in port i
T_{ijv}	Sailing time from port i to port j for ship v
\bar{T}	Number of time periods in the planning horizon
B_{it}	Berth capacity in port i in time period t
\bar{Q}_v^V	Upper loading/unloading capacity of ship v per time period
\bar{Q}_i^P	Upper loading/unloading capacity of port i per time period
L_{vck}^0	Initial load of product k in compartment c on board ship v
\bar{K}_{vc}	Capacity of compartment c in ship v
D_{ikt}	Consumption of product k in port i in time period t
P_{ikt}	Production of product k in port i in time period t
S_{ik}^0	Initial inventory level of product k in port i
\underline{S}_{ik}	Lower inventory limit in port i for product k
\bar{S}_{ik}	Upper inventory limit in port i for product k

Variables

$$x_{ijvt} \begin{cases} 1, & \text{if ship } v \text{ sails from port } i \text{ starting in the beginning of period } t \\ & \text{directly to port } j \\ 0, & \text{otherwise} \end{cases}$$

$$w_{ivt} \begin{cases} 1, & \text{if ship } v \text{ waits outside port } i \text{ in time period } t \\ 0, & \text{otherwise} \end{cases}$$

$$o_{ivt} \begin{cases} 1, & \text{if ship } v \text{ operates in port } i \text{ in time period } t \\ 0, & \text{otherwise} \end{cases}$$

l_{vckt} load on board ship v of product k in compartment c at the end of time period t

q_{ivckt}^L quantity of product k loaded to compartment c by ship v from port i in time period t

q_{ivckt}^U quantity of product k unloaded from compartment c by ship v to port i in time period t

s_{ikt} inventory level in port i of product k at the end of time period t

$$y_{vckt} \begin{cases} 1, & \text{if compartment } c \text{ in ship } v \text{ contains product } k \text{ at the end of} \\ & \text{time period } t \\ 0, & \text{otherwise} \end{cases}$$

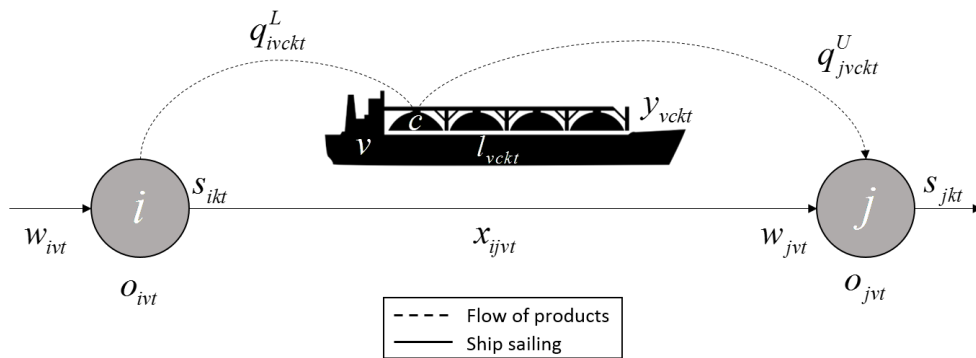


Figure 4.2: Illustration of variables

Objective function

$$\begin{aligned}
 \text{minimize} \quad & \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{N} \cup o(v)} \sum_{j \in \mathcal{N} \cup d(v)} \sum_{t \in \mathcal{T}} C_{ijv}^T x_{ijvt} + \sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} C_v^W w_{ivt} + \\
 & \sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} C_{iv}^O o_{ivt} + \sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}_v} \sum_{k \in \mathcal{K}_v} \sum_{t \in \mathcal{T}} C_{ivk}^Q (q_{ivckt}^L + q_{ivckt}^U)
 \end{aligned} \tag{4.1}$$

The objective function is presented in (4.1). It minimizes the sum of the transportation costs, waiting costs outside the ports, operation costs in the ports and the variable unit cost for loading and unloading activities.

Constraints

Routing constraints

$$\sum_{j \in \mathcal{N} \cup d(v)} x_{o(v)jv1} + o_{o(v)v1} + w_{o(v)v1} = 1 \quad v \in \mathcal{V} \tag{4.2}$$

$$\sum_{i \in \mathcal{N} \cup o(v)} \sum_{t \in \mathcal{T}} x_{id(v)vt} = 1 \quad v \in \mathcal{V} \tag{4.3}$$

$$\begin{aligned}
 \sum_{j \in \mathcal{N} \cup o(v)} x_{jiv(t-T_{jiv})} + w_{iv(t-1)} + o_{iv(t-1)} = \\
 \sum_{j \in \mathcal{N} \cup d(v)} x_{ijvt} + w_{ivt} + o_{ivt}
 \end{aligned} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \tag{4.4}$$

$$o_{iv(t-1)} \leq \sum_{j \in \mathcal{N} \cup d(v)} x_{ijvt} + o_{ivt} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \tag{4.5}$$

$$o_{iv(t-1)} \geq \sum_{j \in \mathcal{N} \cup d(v)} x_{ijvt} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \tag{4.6}$$

$$\sum_{v \in \mathcal{V}} o_{ivt} \leq B_{it} \quad i \in \mathcal{N}, t \in \mathcal{T} \tag{4.7}$$

Constraints (4.2) and (4.3) ensure that all ship schedules have a beginning and an end, i.e. from start-node $o(v)$ to the artificial end-node $d(v)$, respectively. If a ship travels directly from $o(v)$ to $d(v)$, the ship is not used and is idle during the entire planning horizon. Constraints (4.4) are the ship flow conservation constraints. The flow conservation ensures that in each time period of a ship's schedule, the ship either sails, waits or operates. Constraints (4.5) restrict a ship to only have the option to wait when it arrives at the port,

i.e. prior to operation. Constraints (4.6) enforce operations in a port, i.e. a ship cannot leave a port prior to operating, while constraints (4.7) restrict the number of ships in a port in the same time period to not exceed the berth capacity of the port.

Loading and unloading constraints

$$\sum_{k \in \mathcal{K}_v} \sum_{c \in \mathcal{C}_v} (q_{ivckt}^L + q_{ivckt}^U) \leq \min\{\bar{Q}_v^V, \bar{Q}_i^P\} \cdot o_{ivt} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (4.8)$$

$$l_{vck(t-1)} + \sum_{i \in \mathcal{N}} q_{ivckt}^L - \sum_{i \in \mathcal{N}} q_{ivckt}^U - l_{vckt} = 0 \quad v \in \mathcal{V}, c \in \mathcal{C}_v, k \in \mathcal{K}_v, t \in \mathcal{T} \quad (4.9)$$

$$l_{vck0} = L_{vck}^0 \quad v \in \mathcal{V}, c \in \mathcal{C}_v, k \in \mathcal{K}_v \quad (4.10)$$

$$\sum_{k \in \mathcal{K}_v} y_{vckt} \leq 1 \quad v \in \mathcal{V}, c \in \mathcal{C}_v, t \in \mathcal{T} \quad (4.11)$$

$$l_{vckt} \leq \bar{K}_{vc} y_{vckt} \quad v \in \mathcal{V}, c \in \mathcal{C}_v, k \in \mathcal{K}_v, t \in \mathcal{T} \quad (4.12)$$

Constraints (4.8) ensure that a ship can only load/unload when it is operating in a port and define the upper limit on the total quantity loaded/unloaded. Constraints (4.9) represent the load balance for each ship, while Constraints (4.10) define the initial load of every product in every compartment for each ship. Constraints (4.11) ensure that only one product can be in each compartment in the same time period. The load capacity of each compartment is given in Constraints (4.12), which also enforce the binary variable y_{vckc} to be active when there is a load in a compartment.

Inventory constraints

$$s_{ik(t-1)} + \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}_v} q_{ivckt}^U + P_{ikt} = D_{ikt} + \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}_v} q_{ivckt}^L + s_{ikt} \quad i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{T} \quad (4.13)$$

$$\underline{S}_{ik} \leq s_{ikt} \leq \bar{S}_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{T} \quad (4.14)$$

$$s_{ik0} = S_{ik}^0 \quad i \in \mathcal{N}, k \in \mathcal{K} \quad (4.15)$$

Constraints (4.13) are the inventory balance for all ports and products. Constraints (4.14) define lower and upper inventory limits for each product in every port. Lastly, Constraints (4.15) define the initial inventory of each product.

Binary and Non-negativity constraints

$$q_{ivckt}^L, q_{ivckt}^U \geq 0 \quad i \in \mathcal{N}, v \in \mathcal{V}, c \in \mathcal{C}_v, k \in \mathcal{K}_v, t \in \mathcal{T} \quad (4.16)$$

$$l_{vckt} \geq 0 \quad v \in \mathcal{V}, c \in \mathcal{C}_v, k \in \mathcal{K}_v, t \in \mathcal{T} \quad (4.17)$$

$$x_{ijvt} \in \{0, 1\} \quad i \in \mathcal{N}, j \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (4.18)$$

$$w_{ivt}, o_{ivt} \in \{0, 1\} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (4.19)$$

$$y_{vckt} \in \{0, 1\} \quad v \in \mathcal{V}, c \in \mathcal{C}_v, k \in \mathcal{K}_v, t \in \mathcal{T} \quad (4.20)$$

4.3 Valid inequalities

The purpose of this section is to introduce different ways of strengthening the model formulation introduced in Section 4.2. This is done through the use of valid inequalities.

In a general integer program $\max\{cx, x \in X\}$, where $X = \{x : Ax \leq B, x \in Z_+^n\}$, an inequality $Gx \leq G^0$ is a valid inequality for $X \subseteq R^n$ if $Gx \leq G^0$ for all $x \in X$ (Stålhane, 2015b). In words, a valid inequality is said to be an inequality that does not remove any integer feasible solutions from the feasible area. A valid inequality can be called a cut if the inequality removes the optimal solution of the linear programming (LP) relaxation of the integer programming (IP) problem. The wanted result of adding valid inequalities is an improved optimistic bound for the IP problem, and thus a problem that might be faster to solve. The valid inequalities cannot impose any alterations to the IP problem and the optimal integer solution must remain the same after adding valid inequalities.

Since one purpose of a valid inequality is to remove parts of the feasible region that does not contain integer feasible solutions, we studied the LP relaxation of the problem to identify how the model structure was altered when removing the integer requirements. By doing this we could evaluate which parts of the model that has the highest potential for improvement. When removing the integer requirement, the binary constraints on the binary variables are removed and the former binary variables are assigned values continuously between 0 and 1. The binary variables o_{ivt} , w_{ivt} and x_{ijvt} are all part of the objective function and since the objective function is minimized, the model assigns the variables values as close to 0 as possible while also satisfying the constraints in the model. When separating the terms of the objective function in both the IP problem and the LP relaxation, the sailing cost was the cost with the greatest difference between the two solutions. An example that illustrates the difference between the sailing cost from the IP solution and the LP relaxation of the basic model using the large-sized test case is portrayed in Figure 4.3.

By pushing the values of the routing variable x_{ijvt} closer to 1, the difference between the

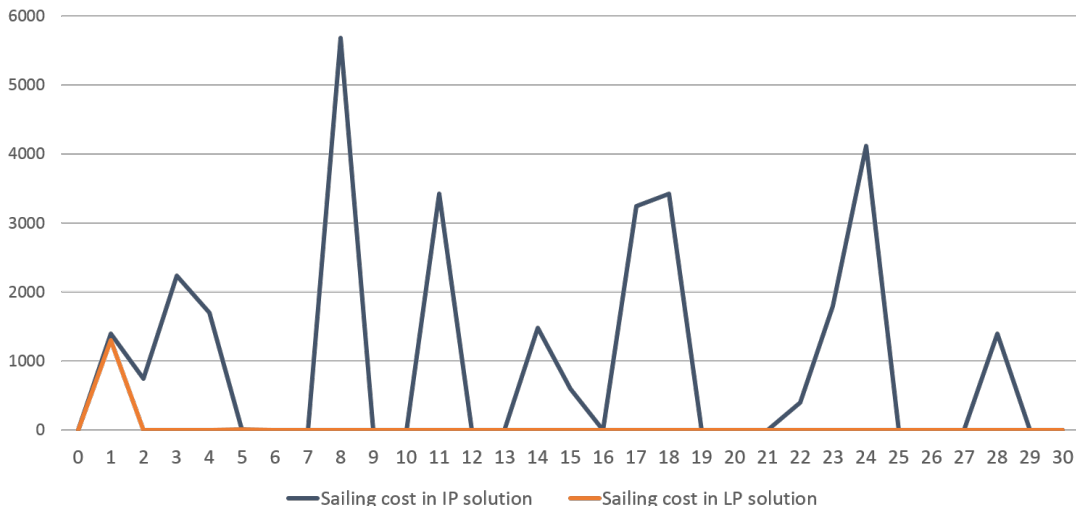


Figure 4.3: Development of sailing cost over the planning horizon. The total sailing cost in the IP solution is 31 640, while the sailing cost in the LP solution is 1 314

sailing cost in the IP and LP solutions decreases. This can be achieved by introducing valid inequalities involving x_{ijvt} . The reason why the routing variable is of special interest is that it lacks a lower bound and little forces the routing variable to be positive. When evaluating the difference in the LP and IP solutions for the operation variable o_{ivt} , on the other hand, the difference between the LP and IP solution is not as significant. This is because the ships still need to satisfy and handle excess demand and supply, and as a result o_{ivt} is pushed up. Hence, the routing variables x_{ijvt} are in focus when developing valid inequalities.

4.3.1 Minimum number of visits with ship capacity sequence

Agra et al. (2013), among others, address valid inequalities that give a lower bound on the number of visits a port needs during the planning period for single-product MIRPs. Rakke et al. (2014) introduce a way of strengthening these types of valid inequalities for MIRPs for LNG. Andersson et al. (2015) attempt to further strengthen the lower bound. Andersson et al. (2015) introduce a ship capacity sequence to avoid the generalization done when the maximum ship capacity is used to calculate the lower bound for the entire planning horizon. This subsection describes the same type of valid inequalities as introduced in Agra et al. (2013) with the strengthening introduced in Andersson et al. (2015), adjusted to a multi-product MIRP-UC model.

We start by defining a time interval as a subinterval of the planning horizon. A ship capacity sequence is defined over a time interval and is built upon two major building blocks. These building blocks are the maximum number of times each ship can visit a port during the planning horizon and the capacity of each ship.

Maximum number of visits. For a given time interval length, the maximum number of

times each ship can visit each port must be calculated. Since we have no predefined routes, loading ports or fixed number of operation hours in each port, the maximum number of visits each ship can have to a port i is calculated based on two valid assumptions; (1) each ship only travel back and forth from port i to its nearest port after the initial visit to port i and (2) each ship only operate for one period in each port visit. For this set of assumptions, the maximum number of visits ship v can make to port i in time interval T' , V_{iv}^{MAX} , is calculated in (4.21).

$$V_{iv}^{MAX} = \left\lceil \frac{|T'| - T_{jiv}}{2 * T_i^{MIN} + 2} \right\rceil \quad i \in \mathcal{N}, v \in \mathcal{V} \quad (4.21)$$

$|T'|$ is the length of time interval T' , and T_i^{MIN} is the sailing time for ship v from port i to its nearest port. The index j in T_{jiv} denotes the ship's position at the beginning of the time interval. In the case where the interval starts in the first time period, j is equivalent to $o(v)$. Equation (4.21) takes the length of the time interval minus the time it takes to travel to port i the first time and divides it with the number of time periods that is necessary to complete one visit. According to the stated assumptions, this is equal to two times the sailing time from port i to the nearest port, plus two operation periods, one in each port.

Ship capacity sequence. The ship capacity sequence, defined for each port i , gives the maximum amount of products that can be loaded or unloaded in a port in m visits during a time interval. First, the ship capacity of the largest ship is added cumulatively to the capacity sequence a number of times equal to the maximum number of visits defined in Equation (4.21). The same follows for the rest of the ships, in descending order based on capacity. The length of the ship capacity sequence is equal to the total number of visits to port i from all ships v in the time interval. The ship capacity sequence of port i is denoted \overline{K}_i^V .

$$\overline{K}_i^V = \{\overline{K}_{i0}^V, \overline{K}_{i1}^V, \dots, \overline{K}_{im}^V\} \quad i \in \mathcal{N} \quad (4.22)$$

The dynamics of the creation of the ship capacity sequence can more easily be illustrated with an example. Assume a fleet of two ships where the largest ship in the fleet has a capacity of 200 and that it can visit port i a maximum of three times in time interval T' . The other ship has a capacity of 100 and can visit port i a maximum of two times. Port i 's ship capacity sequence would then be equal to $\overline{K}_i^V = \{200, 400, 600, 700, 800\}$.

Excess production and consumption. The excess production of product k in port i , $e_{ikT'}^P$, and the excess consumption of product k in port i , $e_{ikT'}^D$, during time interval T' , are defined in (4.23) and (4.24), respectively. Figure 4.4 illustrates the calculation of the excess production and consumption.

$$e_{ikT'}^P = \sum_{t \in T'} P_{ikt} + s_{ik(T'-1)} - \overline{S}_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K} \quad (4.23)$$

$$e_{ikT'}^D = \sum_{t \in T'} D_{ikt} - s_{ik(T'-1)} + \underline{S}_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K} \quad (4.24)$$

Since a ship can handle both excess consumption and production in the same visit, the lower bound on visits for produced and consumed products cannot be added together. On this note, the maximum of $e_{ikT'}^P$ and $e_{ikT'}^D$ aggregated over product, $e_{iT'} = \max\{\sum_{k \in \mathcal{K}} e_{ikT'}^P, \sum_{k \in \mathcal{K}} e_{ikT'}^D\}$, are used as the restricting quantity in the inequalities.

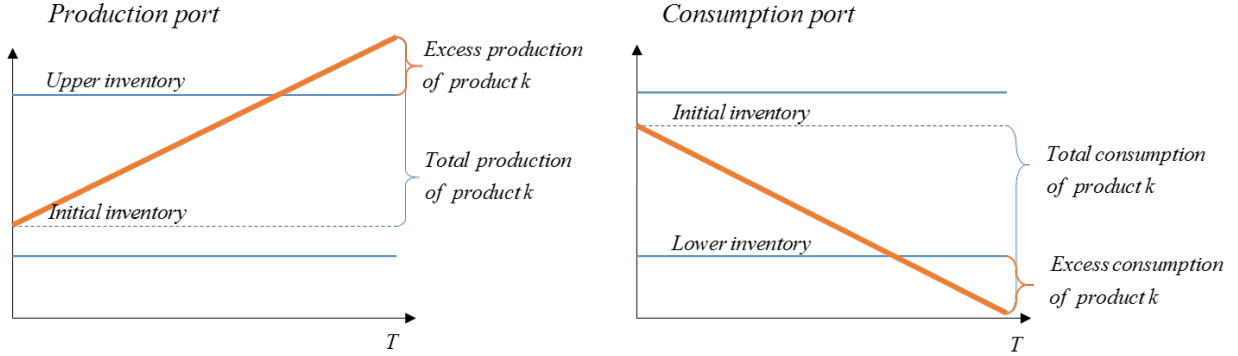


Figure 4.4: Illustration of excess production and consumption over the planning horizon

It is important to note that when using time intervals, the initial stock level is not necessarily fixed. It depends on the start of the time interval. In the case where the time interval starts in the first time period, the initial stock level is S_{ik}^0 , as can be seen in Figure 4.4. In all other cases, the initial stock level is represented by $s_{ik}(T'-1)$. As the initial inventory is known to be S_{ik}^0 when the time interval T' start in $t=1$, the minimum number of visits needed to serve the excess level can be pre-calculated. Let p_i be the first position in the ship capacity sequence corresponding to a capacity high enough to cover $e_{iT'}$. Hence, p_i corresponds to the minimum number of visits needed. In all other cases, the incoming inventory is a variable and this simplification is impossible. The valid inequalities for time interval T' are defined by (4.25) or (4.26) depending on the starting period of the time interval.

$$\sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in T'} x_{jivt} \geq p_i \quad i \in \mathcal{N} \quad (4.25)$$

$$\sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in T'} x_{jivt} \geq \frac{e_{iT'} + (m-1)\bar{K}_{im}^V - m\bar{K}_{i(m-1)}^V}{\bar{K}_{im}^V - \bar{K}_{i(m-1)}^V} \quad i \in \mathcal{N}, 1 < m < |\bar{K}_i^V| \quad (4.26)$$

Valid inequalities (4.25) and (4.26) give a lower bound on the number of ships that needs to be routed to port i in time interval T' . In the case where the time interval starts in $t = 1$, valid inequalities (4.25) are always stronger than (4.26). Valid inequalities (4.26) calculate a lower bound for all number of visits m that can be done during the time interval, where only the highest lower bound is restricting. The fraction on the right hand side of the inequality calculates the number of visits needed by dividing the excess production or consumption by the ship capacity applicable in the m 'th visit. In each of the m iterations, the excess production or consumption is adjusted for the amount of the product that can be handled by ships with a higher capacity in the last $m - 1$ visits.

4.3.2 Minimum number of compartments per product with compartment capacity sequence

A different way of formulating the minimum number of visits constraints is to give a lower bound on the number of compartments that needs to be routed to a port to cover the excess production or consumption of a product. This is an extension of the valid inequality presented by Andersson et al. (2015) adapted to account for both multiple products and a heterogeneous set of compartments.

A compartment capacity sequence is designed equivalently to the ship capacity sequence, using compartment capacities. The sequence is created for all products k and ports i using the compartment capacity and the number of visits for each ship from Equation (4.21). It is written as $\bar{C}_{ik}^V = \{\bar{C}_{ik0}^V, \bar{C}_{ik1}^V, \dots, \bar{C}_{ikm}^V\}$. The sequence is created in the exact same manner as in Equation (4.22) only on compartment level instead of ship level.

The excess production, $e_{ikT'}^P$, and consumption, $e_{ikT'}^D$, is calculated in (4.23) and (4.24) respectively. However, when using compartments, the total excess production or consumption is not aggregated over products since two products cannot share the same compartment like two products can share the same ship. However, production and consumption products can still be handled during the same visit. Hence, the applicable production/consumption amount is always the maximum of $e_{ikT'}^P$ and $e_{ikT'}^D$, denoted $e_{ikT'}$. As presented in Section 4.3.1, if the time interval starts in $t = 1$, excess production/consumption is a parameter and the minimum number of compartments needed can be precalculated. Let p_{ik} represent the first position in the ship compartment capacity sequence sufficient to cover $e_{ikT'}$. Then, p_{ik} is the minimum number of compartments needed for each port, product combination. The valid inequalities for time interval T' giving the lower bound on the number of compartments of ships in V_k that has to be routed to a port i for handling product k are presented in (4.27) and (4.28).

$$\sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}_k} \sum_{t \in T'} N_v^c x_{jivt} \geq p_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K} \quad (4.27)$$

$$\sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}_k} \sum_{t \in T'} N_v^c x_{jivt} \geq \frac{e_{ikT'} + (m-1)\bar{C}_{im}^V - m\bar{C}_{i(m-1)}^V}{\bar{C}_{im}^V - \bar{C}_{i(m-1)}^V} \quad i \in \mathcal{N}, k \in \mathcal{K}, 1 < m < |\bar{C}_i^V| \quad (4.28)$$

N_v^c is the number of compartments on board ship v . By multiplying N_v^c with the routing variable x_{jivt} on the left hand side of the equation, we know the number of compartments in port i if ship v is routed to that port. The sum of compartments in a port must always be greater than the minimum number of compartments required by product k .

4.3.3 Minimum number of operation periods

In continuation of the above idea of minimum number of visits and compartments, the purpose of the following inequality is to give a bound on the minimum number of operation periods in each port. A similar type of valid inequality for minimum number of operation periods formulated for single-product inventory problems can be found in e.g. Agra et al. (2013). Compared to the valid inequalities of this type stated in Agra et al. (2013), we have extended it to account for multiple products and it can be applied on ports that can both produce and consume different products.

Excess production, $e_{ikT'}^P$, and consumption, $e_{ikT'}^D$ is calculated by (4.23) and (4.24) for time interval T' . However, for a lower bound on the operation periods in a port, the excess production and consumption for the entire planning horizon is needed. This is calculated equivalently as (4.23) and (4.24), however the initial stock level is always S_{ik}^0 and the set of time periods used is T . The excess production and consumption for the entire planning horizon is denoted e_{ik}^P or e_{ik}^D respectively. The minimum number of operation periods required by each port is equivalent to the sum of operation periods required by each product. Since the operation variable is not connected to product handled during operation, it is not possible to impose any direct conditions on minimum number of operation periods for each product.

Valid inequalities (4.29) enforce a lower bound on the number of operation periods needed in each port.

$$\sum_{v \in \mathcal{V}} \sum_{t \in T} o_{ivt} \geq \left\lceil \sum_{k \in \mathcal{K}} \frac{e_{ik}^P + e_{ik}^D}{\min\{\bar{Q}_i^P, \max\{\bar{Q}_v^V; v \in \mathcal{V}\}\}} \right\rceil \quad i \in \mathcal{N} \quad (4.29)$$

4.3.4 Timing of first (F), second (S) and last (L) visit to a port

The previously presented valid inequalities only constrain the number of visits or operating periods needed for the entire planning horizon. Now, we present a type of valid inequality that identifies the timing of both the first visit and the second visit in each port to be able to adhere the inventory limits.

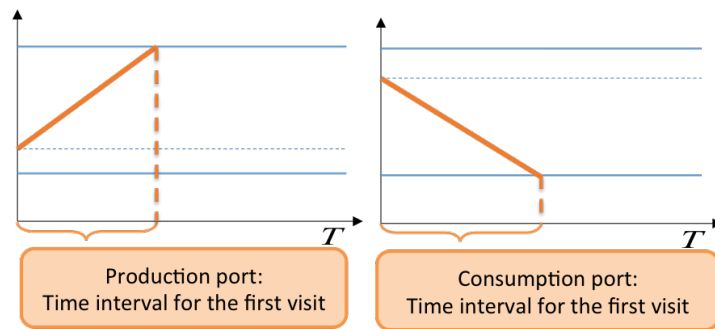


Figure 4.5: Determine the time interval for the occurrence of the first visit

In order to identify the time interval for the first visits, the last time period before the inventory limits are violated must be calculated. The method of calculating the time interval for the first visit is illustrated in Figure 4.5. We calculate the last time period feasible for the first visit for each product-port combination and it is denoted Z_{ik}^1 .

```

if (Port  $i$  is a consumption port for Product  $k$ ) then
  if  $S_{ik}^0 - \sum_{y \in \mathcal{T} | y \leq t} D_{iky} \leq \underline{S}_{ik}$   $i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{T}$  then
     $Z_{ik}^1 \leftarrow t$ 
  end if
else if (Port  $i$  is a production port for Product  $k$ ) then
  if  $\sum_{y \in \mathcal{T} | y \leq t} P_{iky} + S_{ik}^0 \geq \bar{S}_{ik}$   $i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{T}$  then
     $Z_{ik}^1 \leftarrow t$ 
  end if
end if
    
```

A binary parameter F_{ik} is introduced to keep account of whether or not port i needs a visit for a product k during the planning horizon. Thus, F_{ik} is one if Z_{ik}^1 is smaller than the total length of the planning horizon and zero otherwise. Given this, it is possible to construct a valid inequality stating that F_{ik} visit(s) must happen prior to Z_{ik}^1 . The valid inequality is presented in Constraints (4.30) and it states that at least one ship must be routed to and arrive to port j prior to Z_{jk}^1 .

$$\sum_{i \in \mathcal{N} \cup o(v)} \sum_{v \in \mathcal{V}_k} \sum_{\substack{t \in \mathcal{T} \\ t < Z_{jk}^1}} x_{ijvt-T_{ijv}} \geq F_{jk} \quad j \in \mathcal{N}, k \in \mathcal{K} \quad (4.30)$$

Alternatively, the constraint above can be designed using the operation variable o_{jvt} . Constraints (4.31) state that a ship must operate at least F_{jk} timeperiod(s) in port j in time period Z_{jk}^1 or before.

$$\sum_{v \in \mathcal{V}_k} \sum_{\substack{t \in \mathcal{T} \\ t \leq Z_{jk}^1}} o_{jvt} \geq F_{jk} \quad j \in \mathcal{N}, k \in \mathcal{K} \quad (4.31)$$

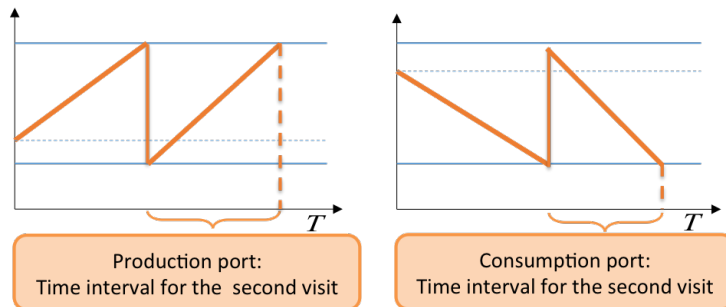


Figure 4.6: Determine the time interval for the occurrence of the second visit

We also want to determine the timing of a potential second visit. A second visit can be calculated given the assumption that a first visit happened in the port in the last period feasible for a first visit, and that the inventory was either filled up/emptied or visited by the largest ship. The method for determining the time interval for a second visit to a port for a given product is illustrated in Figure 4.6. Given these assumptions and the illustration, the last feasible time period for a second visit to port i for product k , denoted Z_{ik}^2 , is calculated.

```

if (Port  $i$  is a consumption port for Product  $k$ ) then
    if  $\min\{C_k, \bar{S}_{ik}\} - \sum_{y \in \mathcal{T} | z_{ik}^1 \leq y \ \& \ y \leq t} D_{iky} \leq \underline{S}_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{T} | t \leq Z_{ik}^1$  then
         $Z_{ik}^2 \leftarrow t$ 
    end if
else if (Port  $i$  is a production port for Product  $k$ ) then
    if  $\sum_{y \in \mathcal{T} | z_{ik}^1 \leq y \ \& \ y \leq t} P_{iky} + \min\{C_k, \underline{S}_{ik}\} \geq \bar{S}_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{T} | t \leq Z_{ik}^1$  then
         $Z_{ik}^2 \leftarrow t$ 
    end if
end if
    
```

As for handling the first visit, we need a binary parameter, A_{ik} , to keep account of the visits to port i for each product k . Equivalently to F_{ik} , A_{ik} is only one if a second visit is needed during the planning horizon. Given that A_{ik} is one, the second visit is required to occur between time Z_{ik}^1 and Z_{ik}^2 . The corresponding valid inequality is presented in Constraints (4.32). As for the first visit, this can also be written in terms of the operation variable. This is presented in Constraints (4.33).

$$\sum_{i \in \mathcal{N} \cup o(v)} \sum_{v \in \mathcal{V}_k} \sum_{\substack{t \in \mathcal{T} \\ Z_{jk}^1 < t < Z_{jk}^2}} x_{ijvt} - T_{ijv} \geq A_{jk} \quad j \in \mathcal{N}, k \in \mathcal{K} \quad (4.32)$$

$$\sum_{v \in \mathcal{V}_k} \sum_{\substack{t \in \mathcal{T} \\ Z_{jk}^1 < t \leq Z_{jk}^2}} o_{jvt} \geq A_{jk} \quad j \in \mathcal{N}, k \in \mathcal{K} \quad (4.33)$$

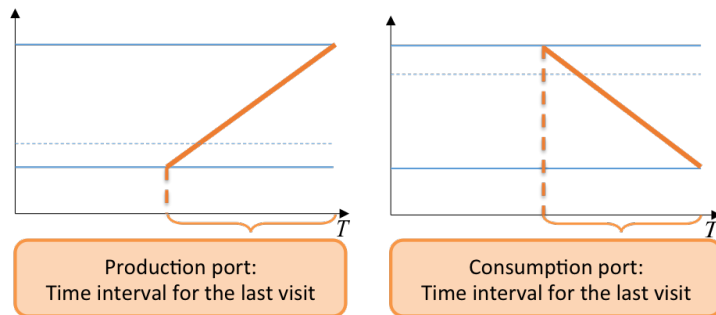


Figure 4.7: Determine the time interval for the occurrence of the last visit

Finally, it is also possible to derive the timing of a time interval for a last visit to a port for a given product. In other words, how long before the final time period of the planning

horizon can the port be visited for the last time and also secure that the inventory limits are not violated at the end of the planning horizon. This is illustrated in Figure 4.7. Given that the inventory is at the extreme limit in the last time period, in which earlier time period is the inventory at the opposite extreme. That time period is equivalent to the start of the time interval for the last visit and it is denoted Z_{ik}^3 for port i , product k . It is derived using the algorithm presented below.

```

if (Port  $\mathbf{i}$  is a production port for Product  $\mathbf{k}$ ) then
  if  $\bar{S}_{ik} - \sum_{g \in \mathcal{T} | g \leq |T| - t + 1} P_{ikg} \leq \underline{S}_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{T}$  then
     $Z_{ik}^3 \leftarrow |T| - t + 1$ 
  end if
else if (Port  $\mathbf{i}$  is a consumption port for Product  $\mathbf{k}$ ) then
  if  $\sum_{g \in \mathcal{T} | g \leq |T| - t + 1} D_{ikg} + \underline{S}_{ik} \geq \bar{S}_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{T}$  then
     $Z_{ik}^3 \leftarrow |T| - t + 1$ 
  end if
end if
    
```

As explained for both the first and the second visit, L_{ik} is equal to one if Z_{ik}^3 is greater than one for port i , product k . Given that L_{ik} is greater than one, a visit is required to occur during the time interval starting in Z_{ik}^3 and ending at the last time period of the planning horizon. The constraints using the routing variable are presented in Constraints (4.34) and the constraints with the operation variable are presented in Constraints (4.35).

$$\sum_{i \in \mathcal{N} \cup o(v)} \sum_{v \in \mathcal{V}_k} \sum_{\substack{t \in \mathcal{T} \\ t \geq Z_{jk}^3}} x_{ijvt-Tijv} \geq L_{jk} \quad j \in \mathcal{N}, k \in \mathcal{K} \quad (4.34)$$

$$\sum_{v \in \mathcal{V}_k} \sum_{\substack{t \in \mathcal{T} \\ t \geq Z_{jk}^3}} o_{jvt} \geq L_{jk} \quad j \in \mathcal{N}, k \in \mathcal{K} \quad (4.35)$$

Chapter 5

Solution Methods

Basic MIRPs are considered to be highly complex problems. This complexity increases with the addition of multiple products and with the use of undedicated compartments to handle the allocation of products. With this level of complexity, an exact solution method is likely to be too time consuming even with a model tightened by valid inequalities. Hence, we propose an alternative solution method to investigate the potential of this problem. In this chapter we propose a matheuristic solution method to solve our MIRP-UC. Boschetti et al. (2009) define matheuristics as heuristic algorithms made by the inter-cooperation of metaheuristics and mathematical programming techniques. However, matheuristics can also be viewed in broader terms as any method where mathematical programming techniques play an important role, but there is no proof of optimality (Stålhane, 2015a).

In this chapter we present our matheuristic solution method designed to solve the MIRP-UC for short sea problems. In Section 5.1, we introduce the overall scheme and idea behind the matheuristic. The following three sections, Sections 5.2-5.4 present the components of the matheuristic in greater detail. Section 5.5 introduces and elaborates on the improvement component of the matheuristic.

5.1 Matheuristic solution method

The matheuristic is composed of two separate phases. First, a construction phase is executed to find a feasible solution to the problem. The construction phase terminates once a feasible solution is found. Following is an improvement phase where the solution is improved through the use of an improvement heuristic. Figure 5.1 depicts the proposed matheuristic.

To design the construction phase, a decomposition strategy is used and the original problem is decomposed into a routing problem and a scheduling- and inventory management problem. Several types of decomposition methods was introduced in Section 3.3. We classify the proposed matheuristic as a partial optimization algorithm, since the problem is decomposed into smaller components and solved iteratively. In this thesis, the idea recognized is that the routing problem and the scheduling- and inventory management problem does not necessarily have to be solved together. The routing problem is solved to generate

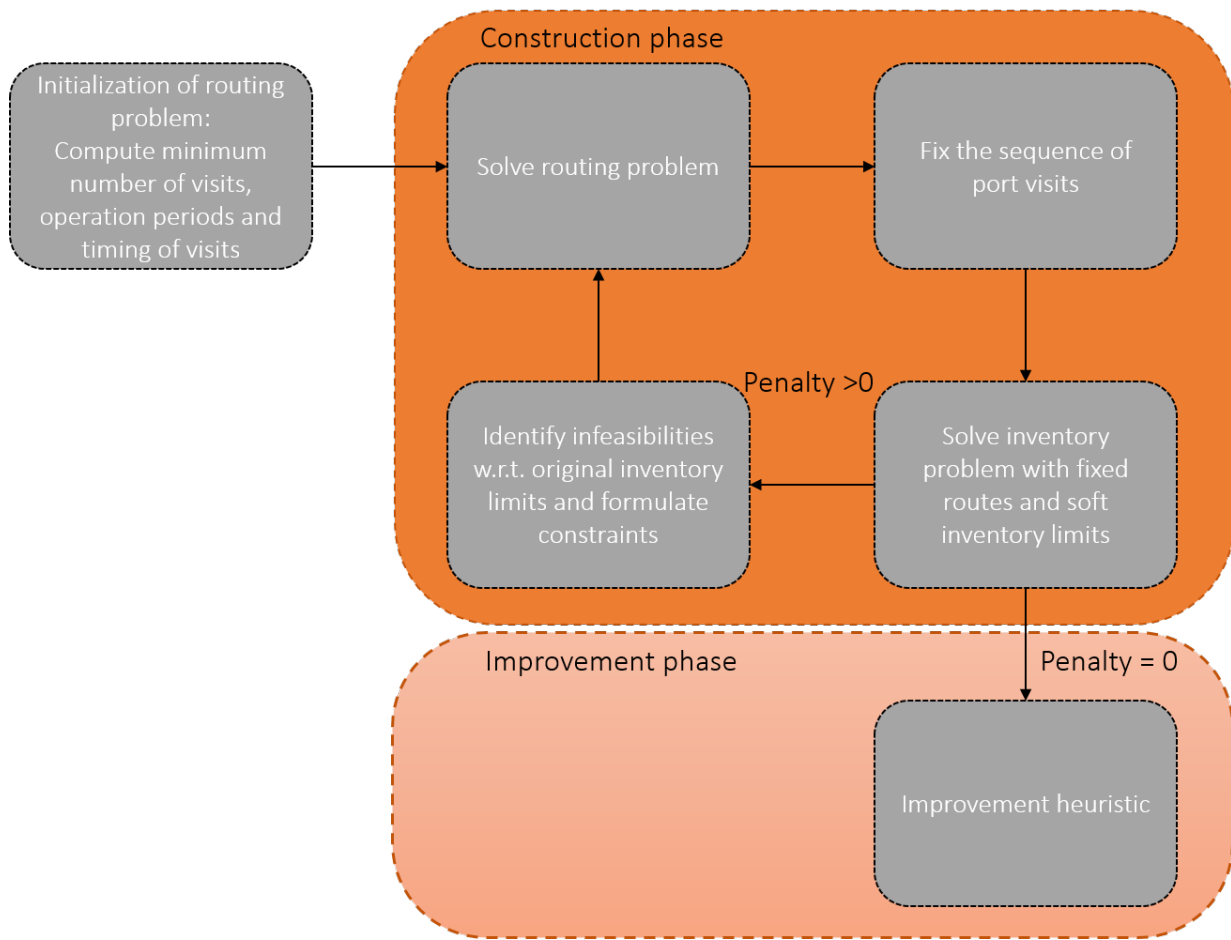


Figure 5.1: Flow chart representing the iterative matheuristic scheme

a set of routes, namely the geography for each ship. Then, the scheduling-and inventory management problem, hereafter called the inventory problem, is solved with the routes found by the routing problem. The scheme is based on the idea of iteratively working towards a feasible solution.

In the first iteration, the routing problem generates a set of initial routes based on a series of valid inequalities presented in Section 4.3. The solution of the routing problem, the set of routes for the ships, serves as input to the inventory problem. Note that it is only the routes, and not the schedules of the ships, that are used by the inventory problem. Since there is no guarantee that the routes generated by the routing problem are optimal in the original model, the inventory problem must be adjusted to be able to handle an infeasible set of routes. The inventory problem is almost equivalent to the original problem, however soft inventory limits are added to be able to handle a set of infeasible routes. The inventory problem is fixed to the routes from the routing problem and is solved to optimality with soft inventory limits. If the inventory problem manages to find a solution that does not utilize the soft limits and stay within the original inventory limits, the construction phase has found a feasible solution. However, if the soft inventory limits are utilized at any point in time, the corresponding solution is infeasible in the original problem and a new iteration is needed.

One of the main ideas of this matheuristic scheme is to utilize the information from the infeasibilities of the inventory problem to guide the generation of routes in the next iteration routing problem. This is done by imposing additional constraints in the routing problem based on the infeasibility information. The purpose is to gradually constrain the routing problem in a way that forces the generated routes towards a set of routes that is feasible in the inventory problem. Thus, the infeasibilities identified in the inventory problem function as input to the next iteration of the routing problem. Hence, for each iteration, the routing problem is further constrained to adhere the inventory requirements of the original problem.

Since the construction phase terminates immediately when the first feasible solution is found, an improvement phase is added to the matheuristic in order to be able to further improve the solution. There is no guarantee that the constraints that are created in the routing problem to handle the violations does not remove the optimal solution. The improvement heuristic uses the solution found in the construction phase as input and explores the neighborhood of this solution to see if a better solution can be found. The improvement phase iteratively relaxes parts of the problem to explore parts of the feasible area that were possibly removed in the construction phase.

In addition to the flow chart presented in Figure 5.1, Algorithm 1 roughly describes the presented matheuristic in a pseudo code framework.

5.2 Construction phase: The routing problem

One of the essential ideas of the proposed matheuristic is the extraction of the routing problem from the inventory and load management part of the problem. The main purpose of solving the routing problem separately is to constrain the remaining part of the problem to a set of fixed routes and thus reduce the size of the problem to a manageable size. The solution of the routing problem is a set of ship routes and schedules. A route of a ship is the sequence of port visits, and as noted earlier, it is only the routes and not the schedules that are used as input to the inventory problem.

The routing problem is used to decide which ports each ship visits during the planning horizon. A goal of the design of the matheuristic is efficiency, and so it is important to keep the problem size of the routing problem as small as possible. In the original formulation, the routing-and scheduling component of the formulation included three variables, namely the routing variable, operation variable and waiting variable. In the routing problem of the matheuristic we, however, only need two types of variables to describe the geography of the solution. The routing variable x_{ijvt} presented in Section 4.2 is essential. In the original formulation, if a ship is routed to a port, at least one operation period is required in that port before the ship is allowed to leave. This should be accounted for in the routing, so that the utilization of the ships over the planning horizon more correctly corresponds to the utilization needed in the original problem. With this in mind, operation variables o_{ivt} are included in the routing problem, and operation periods are assigned to ships according to

Algorithm 1 Overview of Matheuristic

```
Calculate minimum number of visits ( $p_i$ ), compartment visits ( $p_{ik}$ ), operation periods ( $\sigma_i$ ) and time periods for mandatory visits ( $z_{jk}^1, z_{jk}^2, z_{jk}^3$ )  
initialRoutes  $\leftarrow$  RouteGenerator( $p_i, p_{ik}, \sigma_i, z_{jk}^1, z_{jk}^2, z_{jk}^3$ )  
fix sequence of initialRoutes  $\rightarrow$  Sequence  
continue := true  
while continue do  
  CurrentSolution = InventoryProblem(Sequence)  
  Save inventory infeasibilities from InventoryProblem(Sequence)  $\rightarrow$   
inventoryViolations  
  if inventoryViolations is _empty then  
    continue := false  
    return CurrentSolution  
  else  
    Routes = RouteGenerator(inventoryViolations)  
    Fixed sequence of Routes  $\rightarrow$  Sequence  
  end if  
end while  
Run Improvement heuristic with CurrentSolution  
return CurrentSolution
```

the requirements known from the problem definition and the valid inequalities. The original waiting variables are not included. Since the routing problem is not directly subject to inventory limits, it is never necessary to wait prior to operation in the routing problem.

In order to generate routes as close to the optimal routes as possible, it is essential to utilize as much information as possible from the inventory management part of the original formulation. A series of types of valid inequalities that employ this information was proposed in Section 4.3, all of which are included in the routing problem formulation. The goal is to force the routing problem to adhere to the original inventory limits through these valid inequalities. The idea of imposing restrictions on the timing of visits is essential. Even though it is only the routing of the ships that is kept by the inventory problem, imposing a limit on the timing of visits can impact the sequence of the port visits.

By incorporating the introduced valid inequalities and the routing constraints from the original formulation, we have the routing problem as it is in the first iteration. In Section 5.4, the constraints added to the routing problem as a result of infeasibilities in the inventory problem are presented.

Objective function

$$\text{minimize} \quad \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{N} \cup o(v)} \sum_{j \in \mathcal{N} \cup d(v)} \sum_{t \in \mathcal{T}} C_{ijv}^T x_{ijvt} + \sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} C_{iv}^O o_{ivt} \quad (5.1)$$

Constraints

$$\sum_{j \in \mathcal{N} \cup d(v)} x_{o(v)jv1} + o_{o(v)v1} = 1 \quad v \in \mathcal{V} \quad (5.2)$$

$$\sum_{i \in \mathcal{N} \cup o(v)} \sum_{t \in \mathcal{T}} x_{id(v)vt} = 1 \quad v \in \mathcal{V} \quad (5.3)$$

$$\sum_{j \in \mathcal{N} \cup o(v)} x_{jiv(t-T_{jiv})} + o_{iv(t-1)} = \sum_{j \in \mathcal{N} \cup d(v)} x_{ijvt} + o_{ivt} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (5.4)$$

$$o_{iv(t-1)} \leq \sum_{j \in \mathcal{N} \cup d(v)} x_{ijvt} + o_{ivt} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (5.5)$$

$$o_{iv(t-1)} \geq \sum_{j \in \mathcal{N} \cup d(v)} x_{ijvt} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (5.6)$$

$$\sum_{v \in \mathcal{V}} o_{ivt} \leq B_{it} \quad i \in \mathcal{N}, t \in \mathcal{T} \quad (5.7)$$

$$\sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}'} x_{jivt} \geq p_i \quad i \in \mathcal{N} \quad (5.8)$$

$$\sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}_k} \sum_{t \in \mathcal{T}'} N_v^c x_{jivt} \geq p_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K} \quad (5.9)$$

$$\sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} o_{ivt} \geq \sigma_i \quad i \in \mathcal{N} \quad (5.10)$$

$$\sum_{v \in \mathcal{V}_k} \sum_{i \in \mathcal{N} \cup o(v)} \sum_{\substack{t \in \mathcal{T} \\ t < Z_{jk}^1}} x_{ijvt-T_{ijv}} \geq F_{jk} \quad j \in \mathcal{N}, k \in \mathcal{K} \quad (5.11)$$

$$\sum_{v \in \mathcal{V}_k} \sum_{i \in \mathcal{N} \cup o(v)} \sum_{\substack{t \in \mathcal{T} \\ Z_{jk}^1 < t < Z_{jk}^2}} x_{ijvt-T_{ijv}} \geq A_{jk} \quad j \in \mathcal{N}, k \in \mathcal{K} \quad (5.12)$$

$$\sum_{v \in \mathcal{V}_k} \sum_{\substack{t \in \mathcal{T} \\ t \geq Z_{jk}^3}} o_{jvt} \geq L_{jk} \quad j \in \mathcal{N}, k \in \mathcal{K} \quad (5.13)$$

$$x_{ijvt} \in \{0, 1\} \quad i \in \mathcal{N}, j \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (5.14)$$

$$o_{ivt} \in \{0, 1\} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (5.15)$$

Constraints (5.2) and (5.3) are the start and end conditions, respectively, while Constraints (5.4) give the flow balance. Constraints (5.5) and (5.6) handle the timing of operation and require operation in a port prior to sailing to the next port. Constraints (5.7) are the berth capacity constraints. Constraints (5.8) enforce a minimum number of visits to a port as presented in Section 4.3.1, while Constraints (5.9) enforce a minimum number of compartments necessary for each port-product combination and are presented in Section 4.3.2. Constraints (5.10) require a minimum number of operation periods in a port and are presented in Section 4.3.3. Constraints (5.11)- (5.13) are the first, second and last visit constraints which are presented in Section 4.3.4. Finally, Constraints (5.14) and (5.15) are the binary restrictions. The objective function is given in Equation 5.1, and only the relevant components of the original model formulation are included.

5.3 Construction phase: The inventory problem

The matheuristic iteratively tries to find a feasible solution to the problem. The purpose of the inventory problem is to check the feasibility of the routes generated by the routing problem as well as solving the scheduling, allocation and inventory management problem. The inventory problem essentially corresponds to the original MIRP that is formulated in Chapter 4, however with the set of routes fixed. This greatly reduces the size of the problem and likewise the solution time. The routes are included in the inventory problem as parameters originating from the routing problem. For each iteration, new routes are generated and the parameters are updated correspondingly.

Since the route of each ship is fixed, the decisions to be made by the inventory problem is the scheduling of each ship, the allocation of products to compartments and the inventory management in each port. There is no guarantee that the routes generated by the routing problem is feasible in the inventory problem. The only consequence of an infeasible route in the inventory problem is that at least one product exceeds its inventory limits in a port at some or several points during the planning horizon. The only adaptation that must be made to account for possible route infeasibilities is the addition of soft inventory limits.

With soft inventory limits, the inventories of each product in the ports are allowed to exceed their inventory limits at a high penalty cost. Consequently, the inventory problem is considered feasible when the soft inventory limits are utilized. However, the need to utilize the soft inventory is equivalent with an infeasible set of routes in the original problem. The feasible solution of the inventory problem includes information about the port, product,

time combination where the soft inventory limits are utilized. With this information, it is possible to understand what makes the routes from the routing problem infeasible. Hence, this is the information returned to the routing problem from the inventory problem. When the solution of the inventory problem does not use the soft inventory limits, a feasible solution to the original problem is found and the construction phase is terminated. Figure 5.2 illustrates an example of an inventory problem solution with a set of infeasible routes forcing the inventory problem to utilize the soft inventory. In the figure, the usage of the soft inventory is marked with red in port 2.

In the remainder of this section, the discussed inventory problem is presented. The inventory problem represents the result of adapting the original problem formulation presented in Section 4.2 to the above specifications.

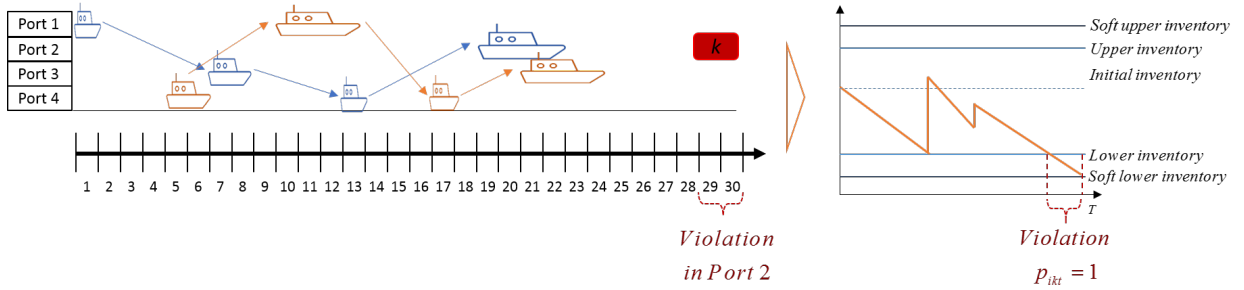


Figure 5.2: Illustration of the functionality of the inventory problem

Sets and indices

\mathcal{H}_v Set of all hops h made by ship v To keep the routes received from the routing

problem fixed in the inventory problem, a hop formulation is used. To be able to index each hop that a ship makes, i.e. keep account of every arc that is traversed by ship v and in which order the arcs are traversed, set H_v is introduced. In H_v , each element h is the h 'th arc traversed in the route of ship v .

Parameters

All of the parameters from the original formulation are kept in the inventory problem. In addition, the following parameters are added.

R_{vml}	Port visited as number m in the route of ship v in <i>Iteration</i> l of the routing problem
\bar{A}_{ijvl}	Number of times ship v traverses the arc between port i and port j during the planning horizon in iteration l
C_t^P	Penalty cost for extending the inventory limits for a product in a port in time period t
S^S	Additional inventory if soft inventory limits are evoked
\bar{H}_v	Number of hops made by ship v

For practical reasons, we refer to R_{vl} as the route of ship v in *Iteration* l , where $R_{vl} = \{R_{vml} | m = 1, \dots, M\}$.

Variables

The original routing variables x_{ijvt} are replaced by r_{ijvht} . The hop index h is added to control which x_{ijvt} variable a routing variable r_{ijvht} in the inventory problem corresponds to. In addition, penalty variables p_{ikt} are added to the problem to penalize the use of soft inventory limits.

$$r_{ijhvt} \begin{cases} 1, & \text{if ship } v \text{ in hop } h \text{ sails from port } i \text{ starting in the beginning of} \\ & \text{time period } t \text{ directly to port } j \\ 0, & \text{otherwise} \end{cases}$$

$$p_{ikt} \begin{cases} 1, & \text{if port } i \text{ extends its inventory limits for product } k \text{ in time period } t \\ 0, & \text{otherwise} \end{cases}$$

R_{vl} contains the route received from the routing problem in terms of the sequence of port visits. The routing variable of the inventory problem, r_{ijhvt} , is limited to the sequence given by R_{vl} . Hence, the inventory problem can only make scheduling decisions for the routes received from the routing problem. The ijv combinations defining the arcs that are travelled by each ship, is given by $i = R_{vml}$ and $j = R_{v(m+1)l}$. The hop index h corresponds to the m index in R_{vml} . For example, the routing problem generates the following route for *Ship* 1: *Port* 1 \rightarrow *Port* 3 \rightarrow *Port* 2 \rightarrow $d(v)$. Then the only r_{ijhvt} variables that are created for *Ship* 1 are r_{1311t} , r_{3221t} , and r_{2051t} for all time periods t . Eliminating all other combinations reduces the size of the inventory problem significantly. The creation of operation variables o_{ivt} and w_{ivt} to ship v is also restricted to the ship route R_{vl} .

p_{ikt} is the binary variable used to introduce the soft inventory limits in the inventory problem. In other words, p_{ikt} gives information about when a port i for product k violate the inventory limits of the original problem. Note that a violation in a port cannot occur before it is possible for a ship to arrive to the port. Hence, the penalty variables p_{ikt} are only created for time periods after the nearest ship that can carry product k is able to reach port i .

Objective function

$$\begin{aligned}
 \text{minimize} \quad & \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{N} \cup o(v)} \sum_{j \in \mathcal{N} \cup d(v)} \sum_{h \in \mathcal{H}_v} \sum_{t \in \mathcal{T}} C_{ijv}^T r_{ijhvt} + \sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} C_v^W w_{ivt} + \\
 & \sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} C_{iv}^O o_{ivt} + \sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}_v} \sum_{k \in \mathcal{K}_v} \sum_{t \in \mathcal{T}} C_{ivk}^Q (q_{ivckt}^L + q_{ivckt}^U) + (5.16) \\
 & + \sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{K}} \sum_{t \in \mathcal{T}} C^P p_{ikt}
 \end{aligned}$$

The objective function used in the inventory problem is presented in (5.16). Note that with the new hop index introduced in the routing variables, we must also sum over all hops to find the transportation costs. In addition to the original costs, the penalty cost for enforcing soft inventory limits for a product in a port in a time period is added. When the original inventory limits are adhered, no penalty cost is incurred. Hence, the cost structure of the real problem described in Chapter 2 is not changed.

Constraints

All routing constraints (4.2)-(4.7) and loading constraints (4.8)-(4.12) are kept unchanged, except for the routing variables x_{ijvt} being replaced by r_{ijhvt} . Even though the routing is fixed and defined by R_{vl} , flow conservation must be maintained when scheduling decisions are made. Note that similarly to the objective function, whenever the routing variables are part of the constraints, a summation over all hops must be added to correctly include the routing variables.

Hop constraints

In addition to the original flow conservation constraints, the following constraints are included to force the routes in the inventory problem to be equal to the routes defined in R_{vl} .

$$\sum_{j \in \mathcal{N} \cup o(v)} \sum_{t' \in \mathcal{T} | t' < t} r_{jivt'h} \geq \sum_{j \in \mathcal{N} \cup d(v)} r_{ijvt(h+1)} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T}, h \in \mathcal{H}_v | h < \bar{H}_v \quad (5.17)$$

$$\sum_{i \in \mathcal{N} \cup o(v)} \sum_{j \in \mathcal{N} \cup d(v)} \sum_{t \in \mathcal{T}} r_{ijhvt} = 1 \quad v \in \mathcal{V}, h \in \mathcal{H}_v \quad (5.18)$$

$$\sum_{t \in \mathcal{T}} r_{ijvt} = \bar{A}_{ijv} \quad i \in \mathcal{N} \cup o(v), j \in \mathcal{N} \cup d(v), v \in \mathcal{V} \quad (5.19)$$

Constraints (5.17) make sure that the sequence of port visits of each ship v is maintained in the schedule created by the inventory problem. This means that the routing variable corresponding to the h 'th arc that ship v traverses, cannot be positive until the previous $(h-1)$ 'th arc has been traversed by the same ship. Constraints (5.18) make sure that each

hop made by a ship v is distinct for a (ijt) combination, and must be made exactly once. In addition, Constraints (5.19) state that the number of times ship v travels from port i to port j during the planning horizon, must be equal to the number of times the port combination (i, j) occurs in the ship's route.

In Section 4.3.4, valid inequalities on timing was presented. Since only the route and not the schedule is fixed from the routing problem, this type of valid inequalities is also needed in the inventory problem. Constraints (4.31), (4.33), and (4.35), presented in Section 4.3.4, are included to enforce restrictions on the timing of visits made to a port.

Inventory constraints

Inventory balance constraints (4.13) and initial inventory constraints (4.15) are kept unchanged, while constraints (4.14) are adjusted to include soft inventory limits, presented in constraints (5.20).

$$\underline{S}_{ik} - S^S p_{ikt} \leq s_{ikt} \leq \overline{S}_{ik} + S^S p_{ikt} \quad i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{T} \quad (5.20)$$

Constraints (5.20) define lower and upper inventory limits for each product in every port. As can be seen, the inventory problem introduces the possibility of extending the inventory limits for a product in a port and by that forcing the binary variable p_{ikt} to be equal to one in those time periods.

Binary and non-negativity constraints

$$r_{ijhvt} \in \{0, 1\} \quad i \in \mathcal{N}, j \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (5.21)$$

$$p_{ikt} \in \{0, 1\} \quad i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{T} \quad (5.22)$$

5.4 Construction phase: Handling of infeasibilities

As discussed in Section 5.1, an important component of this matheuristic scheme is the transition from the solution of the inventory problem to the next iteration of the routing problem. If the inventory problem is only able to find a feasible solution by utilizing the soft inventory, a new iteration of the construction phase is executed. The next iteration routing problem is further constrained by utilizing information from the infeasibilities of the inventory problem. The purpose of this section is to develop suitable strategies for handling the infeasibility information in the routing problem.

Essentially, an infeasibility in the inventory problem occurs when a product in a port is not served in time, forcing the inventory problem to utilize the soft inventory. Given that an infeasibility occurs in port i for product k in time period t , the penalty variable p_{ikt} is set equal to one and we call this port, product and time period combination for a violation. Hence, each violation has a violation port, violation product and a violation time. The

violation also includes information about whether it is a violation to a production product or a consumption product in the violation port. The incurred violations are used when constructing constraints for the next iteration routing problem. It is important to note that p_{ikt} is positive for the port, product combination for all time periods that the inventory limits are violated, however it is only the first time period the violation occurs that is used when constructing the constraints for the routing problem.

In this section, we present six different strategies for handling the violations, i.e. ways of creating constraints using the information from the violations. The focus when developing the strategies is two-folded. We both want to restrict the next iteration routing problem enough to avoid getting the same violations in the inventory problem but also avoid removing the optimal route solution. Note that when solving the MIRP-UC with the matheuristic, it might be necessary to use multiple of the following strategies simultaneously to secure a faster convergence towards a feasible set of routes.

5.4.1 Strategy 1: Increasing the minimum number of arcs required to the violation port prior to the violation time

One way to utilize the violation information is to impose a requirement on the number of arcs going into the violation port. In a greedy mindset, we can utilize the information on whether the violation product is produced or consumed in the violation port. In the case of a consumption violation product, i.e. consumption violation, we impose an arc from a production port of the violation product directly to the violation port prior to the violation time. A consumption violation is equivalent to a shortage of the violation product, which can be supplied from a production port of the violation product. Similarly, a direct arc is imposed from a consumption port to the violation port in the case of a production violation product, i.e. a production violation, prior to the violation time. As a production violation is equivalent to excess inventory of the violation product in the violation port, it is not possible to validly justify the direct arc from a consumption port to the violation port. However, by forcing a visit to a consumption port directly prior to the production violation port, the ship is able to free up space in its compartments for the violation product.

Figure 5.3 shows an example of how Strategy 1 handles violations from the inventory problem. This example illustrates a consumption violation. Before the strategy is presented in a pseudo code, the new needed notation is presented.

\mathcal{L}	Set of all iterations
\mathcal{N}_k^P	Set of production ports for product k
\mathcal{N}_k^C	Set of consumption ports for product k
T_{ikt}^L	The first time period the violation (i, k, t) can be resolved
\bar{A}_{ijvtl}	Number of times ship v traverse the arc between port i and port j prior to time period t in <i>Iteration</i> l

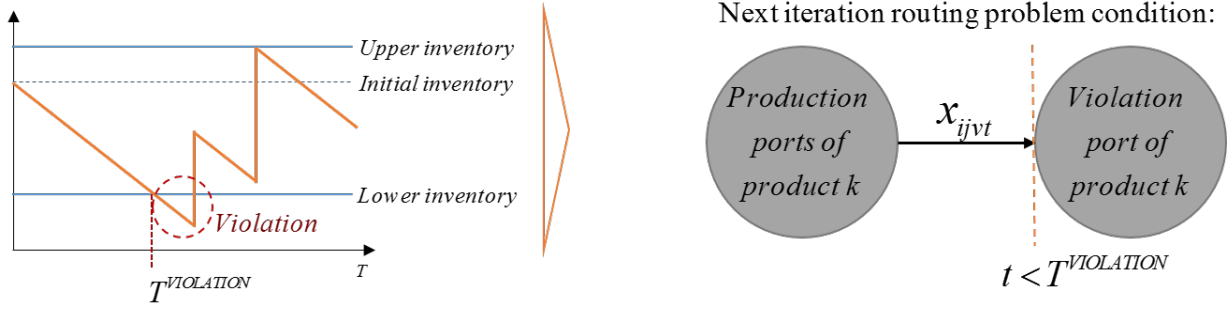


Figure 5.3: Illustration of Strategy 1 functionality for a consumption port violation

Algorithm 2 Strategy 1: Increasing the minimum number of arcs required to the violation port prior to the violation time

for (l in Iteration) **do**

for each (Violation of l) **do**

 Violation port = i , Violation product = k , Violation time = t

if (Port i is a consumption port for Product k) **then**

$$\sum_{j \in \mathcal{N}_k^P} \sum_{v \in \mathcal{V}_k} \sum_{t' \in \mathcal{T} | t' \leq t - T_{jiv} - 1} x_{jivt} \geq \sum_{j \in \mathcal{N}_k^P} \sum_{v \in \mathcal{V}_k} \sum_{\substack{t' \in \mathcal{T} \\ t' \leq t - T_{jiv} - 1 \\ t \geq T_{ikt}^L}} \bar{A}_{ijvlt} + 1 \quad (5.23)$$

else if (Port i is a production port for Product k) **then**

$$\sum_{j \in \mathcal{N}_k^C} \sum_{v \in \mathcal{V}_k} \sum_{t' \in \mathcal{T} | t' \leq t - T_{jiv} - 1} x_{jivt} \geq \sum_{j \in \mathcal{N}_k^C} \sum_{v \in \mathcal{V}_k} \sum_{\substack{t' \in \mathcal{T} \\ t' \leq t - T_{jiv} - 1 \\ t \geq T_{ikt}^L}} \bar{A}_{ijvlt} + 1 \quad (5.24)$$

end if

end for

end for

The procedure of creating new constraints for the next iteration routing problem is presented in Algorithm 2. It is important to note that in *Iteration* l , all violations from $1 \dots l - 1$ are added to the routing problem to avoid returning to a previous violation.

If a violation occurs in the last time period, it is not necessarily possible to resolve this violation in the first time period. By this we mean that routing a ship to a violation port in first time period might not ensure that a the violation in the last time period is accounted for. Hence, the first time period in which the violation can be resolved must be calculated. A violation in time period t is equivalent to saying that the inventory of product k is at or beyond its extreme limit, i.e upper inventory limit for a production product and lower inventory limit for a consumption product. Given the time of the violation, the inventory limit in that time period and the quantity produced or consumed in each time period are needed to calculate the earliest time period in the time horizon where the inventory is at the opposite inventory limit. This time period is denoted T_{ikt}^L . Time interval $[T_{ikt}^L, t]$ is thus the time interval where it is possible to solve the violation with violation time period t .

Compared to the last iteration routing solution, the result of this strategy is not necessarily one additional arc into the violation port. The dimension of time is included in the greedy constraints presented in Algorithm 2. It is only demanded that an additional arc is added prior to the violation time, and it might be that the arc added is in fact only an arc moved to an earlier time. It is, however, essential that the measures taken to resolve the violations result in a change in port visit sequence and not only a scheduling difference, as the time stamps of the visits in the routing problem is not forwarded to the inventory problem. Hence, if there is only a scheduling change and not a routing change, the inventory problem will return the same set of violations in the next iteration, making the previous iteration redundant.

5.4.2 Strategy 2: Increasing the minimum number of operation periods in violation ports

Another way of handling the violations from the inventory problem is to impose restrictions on the operation periods needed in each port. The idea is that the lower bound on the total number of operation periods in a violation port can be improved by demanding additional operation periods to account for the shortage/excess amount of the violation products. p_{ikt} is the binary variable in the inventory problem accounting for the violations to the original inventory limits. From this variable, the first and the last time period for each violation can be derived. This time interval is used to calculate the total amount that violate the inventory limits and thus also the number of operation periods needed to account for the violation. First, the new notation needed is introduced.

\mathcal{K}_i^V	Set of violation products for violation port i
O_{il}	Number of operation periods used in port i in iteration l of the inventory problem
T_{ikl}^1	First time period of violation for violation port i and violation product k in iteration l
T_{ikl}^2	Last time period of violation for violation port i and violation product k in iteration l

In addition to the number of operation periods used in the previous iteration of the inventory problem, the number of operation periods needed to account for all violation products in a port should be added to this bound. In each violation port i in *Iteration* l , the shortage/excess amount of violation product k in the violation time interval $[T_{ikl}^1, T_{ikl}^2]$ can be calculated. The total quantity in which the inventory limits are violated is the sum over all the violation products in the violation port. Thus, the total number of operation time periods needed to account for the violation quantity is the ceil of the total violation quantity divided by the appropriate loading capacity. One constraint for each violation port is added to the next iteration of the routing problem. The corresponding algorithm is presented in Algorithm 3.

Algorithm 3 Strategy 2: Increasing the minimum number of operation periods in violation ports

for (l in Iteration) **do**
 for each (Violation in l) **do**
 Violation port = i, Violation product = k

$$\sum_{k \in \mathcal{N}_k^P} \sum_{v \in \mathcal{V}_k} \sum_{t \in \mathcal{T}} o_{ivt} \geq O_{i(l-1)} + \left[\frac{\sum_{t \in \mathcal{T}} (D_{ikt} + P_{ikt})}{\sum_{k \in \mathcal{K}_i^V} \frac{T_{ik(l-1)}^1 \leq t \leq T_{ik(l-1)}^2}{\max\{\min\{\bar{Q}_i^P, \bar{Q}_v^V\}; v \in \mathcal{V}\}}} \right] \quad (5.25)$$

end for
end for

5.4.3 Strategy 3: Introducing a commodity flow formulation for each violation

The first strategy presented, Strategy 1, is quite restrictive. A way of making Strategy 1 less restrictive is to relax the requirement on the sequence of port visits. With Strategy 3, we only impose that a port visit must be made prior to, and not directly precede, the visit to the violation port. Apart from this relaxation, Strategy 3 is based on the same assumptions as introduced in Strategy 1. Hence, in the case of a consumption violation, a production port of the violation product must be visited prior to the violation port. Equivalently, for a production violation, a consumption port of the violation product must be visited prior to the violation port.

To accommodate this, a commodity flow formulation is proposed. With a commodity flow formulation, it is possible to impose restrictions on the port visit sequence by requiring that a flow of a commodity goes from a source port to a sink port. The sink always corresponds to the violation port, while the sources are always the corresponding consumption or production ports depending on the type of violation. For each violation port and product, a flow network is created where only the edges to transport the commodity to the sink within the violation time are created. Since there might be more than one source, a super source is introduced. From the super source, only the arcs going to the sources are created and with unlimited capacity.

First, we define the different concepts needed to understand a flow formulation. Following is an introduction of the notation used, as well as a presentation of the commodity flow formulation in a pseudocode framework.

- Source nodes - \mathcal{N}_k^P is the set of source nodes for a consumption violation, while the set \mathcal{N}_k^D is the source nodes for a production violation.
- Super source - artificial node connecting multiple source nodes

- Sink node - the violation port
- Transshipment nodes - all nodes that are neither source nor sink nodes for a given violation

Sets and parameters

\mathcal{N}^S	Set of all source ports
\mathcal{N}^{SS}	Set of all source ports including the super source S
Q_{kdl}^C	Violation product k commodity with destination in violation port d in iteration l

Variables

$f_{ijvtkdl}^X$	flow between port i and port j with ship v in time period t of violation product k with destination port d in iteration l
f_{ivtkdl}^O	flow during operation in port i with ship v in time period t of violation product k with destination port d in iteration l

It is assumed that all violations are uniquely defined by violation port, violation product and iteration number. Thus the variables does not have an index for violation number from the inventory problem. In *Iteration l* of the routing problem, an isolated commodity flow network is created for each violation for each of the iterations $1..l - 1$ of the inventory problem of the construction phase. The reason for this is that the node definitions, i.e. which ports are defined as source, sink or transshipment nodes, are specific for each violation. Therefore, the violations must be treated separately in order for the flow balance to be correct. The functionality of solving a violation using Strategy 3 is illustrated in Figure 5.4. Algorithm 4 shows how the commodity flow formulation is added to the routing problem.

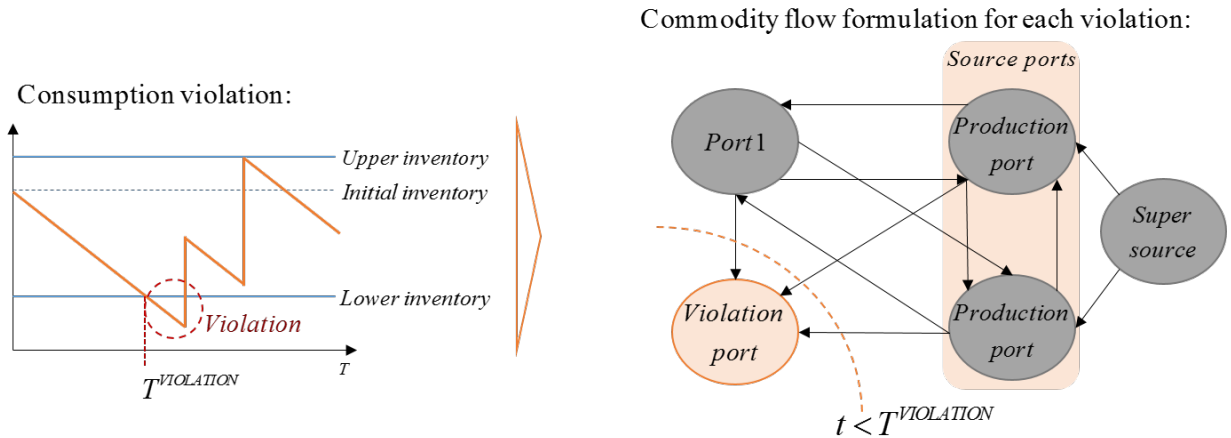


Figure 5.4: Illustration of Strategy 3 functionality for a consumption port violation

Constraints (5.26) and (5.27) defined in Algorithm 4, restrict the flow variables to be positive only when the corresponding routing or operation variables are positive. Hence, the

Algorithm 4 Strategy 3: Introducing a commodity flow formulation for each violation

for (l in Iteration) **do**

for each (Violation in l) **do**

 Violation port = d, Violation product = k, Violation time = t

$$f_{ijvgkdl}^X \leq Q_{kdl}^C x_{ijvg} \quad i \in \mathcal{N}, j \in \mathcal{N}, v \in \mathcal{V}_k, g \in \mathcal{T} | g \leq t - T_{idv} \quad (5.26)$$

$$f_{ivgkdl}^O \leq Q_{kdl}^C o_{ivg} \quad i \in \mathcal{N}, v \in \mathcal{V}_k, g \in \mathcal{T} | g \leq t - T_{idv} \quad (5.27)$$

$$\sum_{i \in \mathcal{N} | i \neq d} \sum_{v \in \mathcal{V}_k} \sum_{t \in \mathcal{T} | g \leq t} f_{idv(g-T_{ijv})kdl}^x = Q_{kdl}^C \quad (5.28)$$

$$\sum_{j \in \mathcal{N}^S} \sum_{v \in \mathcal{V}_k} \sum_{t \in \mathcal{T} | g \leq t - T_{jdv}} f_{Sjvgkdl}^x = Q_{kdl}^C \quad (5.29)$$

$$\sum_{i \in \mathcal{N}^{SS} | i \neq d} f_{ijv(g-T_{ijv})kdl}^x + f_{jvg-1kdl}^o = \sum_{i \in \mathcal{N}} f_{jivgkdl}^x + f_{jvgkdl}^o \quad j \in \mathcal{N} | j \neq d, v \in \mathcal{V}_k, g \in \mathcal{T} | g \leq t - T_{jdv} \quad (5.30)$$

end for

end for

flow can only follow the path of a ship. Constraints (5.30) give the commodity flow balance. A flow from either the super source or any other node that is not the sink node to a port i , must be equal to the flow going out from port i to all other nodes, including the sink node. The variables f_{jvtkdl}^o are included to ensure that the flow is preserved in every time period, also the time periods used for operation. Constraints (5.28) force the commodity to be transported to the sink node, i.e. the violation port d , before violation time t . Finally, Constraints (5.29) ensure that the commodity must travel from the super source to at least one of the sources.

Elimination of flow variables. This strategy impose a significant increase in the number of variables in the routing problem. Hence, it is important to keep the number of variables as low as possible. The right side of Figure 5.4 gives an illustration of which arcs that are created. No arcs leaving the sink node is created. The only arcs created in association to the super source are the arcs going from the super source to the set of source nodes. All other arcs are only created in the part of the time domain that is feasible with respect to the violation time. This also applies for the flow operation variables.

5.4.4 Strategy 4: Alternative commodity flow formulation for each violation

As introduced above, a commodity flow network has a source and a sink. In this thesis problem, there are multiple sources which produce/consume the violation product and a single sink that consumes/produces the violation product. In the above formulation, a super source was introduced to be able to handle multiple potential sources for the commodity. When developing the formulation, an alternative method to handle multiple sources emerged. There are still three distinct categories of nodes in the commodity flow network, namely the sources, the sink and the transshipment nodes. The idea is to formulate the commodity flow formulation with multiple potential sources without utilizing a super source.

With multiple potential sources and the possibility of travelling between those source, it is hard to formulate the flow balance for the source nodes. It is necessary to have one flow balance for the source nodes, aggregated over all source nodes and all time periods. Consequently, discrepancies in the formulation can occur due to the possibility of sailing between source nodes. When the ships can sail between the potential sources, the same flow variables can occur on both sides of the source flow balance, consequently cancelling them out. This is illustrated in Figure 5.5. Figure 5.5 shows a case where both *Port A* and *Port B* are source nodes. All the possible incoming and outgoing edges are also included. As can be seen in the figure, the arc between *Port A* and *Port B* occurs both as an incoming arc to a source and an outgoing arc from the source node, marked in blue. The same applies to the arc between *Port b* and *Port A*, marked in orange. Since we sum over all ports on both sides of the balance, these two sets of arcs will cancel each other out in the flow balance. This leads to an infeasible formulation. Thus, we need to distinguish between the edges going into a node and the edges leaving the node to be able to avoid the use of a super source. We now introduce the flow variables as in the previous subsection, however adjusted for the distinguishing factor introduced above.

Variables

$f_{ijvtkdl}^{X,IN}$	commodity flow from port i IN to port j with ship v in time period t of product k with destination port d in iteration l
$f_{ijvtkdl}^{X,OUT}$	commodity flow OUT of port i to port j with ship v in time period t of product k with destination port d in iteration l
f_{ivtkdl}^O	flow during operation in transshipment port i with ship v in time period t of product k with destination port d in iteration l

The flow operation variables are now only created for the transshipment nodes as these are the only nodes that have a flow balance in each time period. We have two sets of flow routing variables, one set of IN variables and one set of OUT variables. For the educated reader this might seem strange as these two variables are always equal for the same (i, j, v, t) combination. However, this formulation twist is one way of making a commodity flow formulation with multiple sources without utilizing a super source. Again, one commodity flow network is created for each violation in each of the previous iterations and

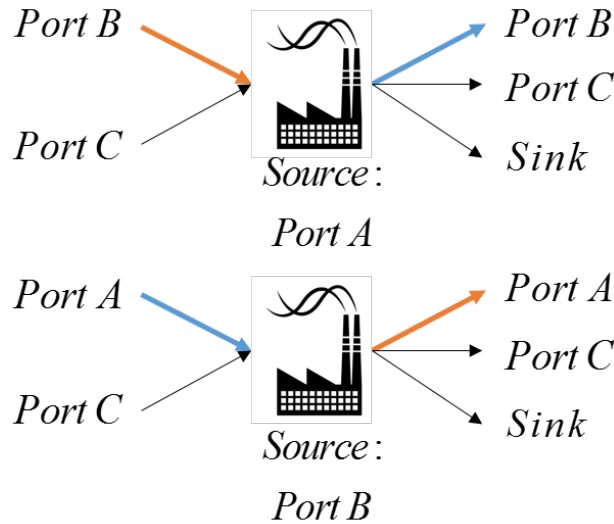


Figure 5.5: Illustration of issue with multiple sources

added to the next iteration routing problem. The algorithm and constraints are presented in Algorithm 5.

Constraint (5.31) is an aggregated flow balance for all the source nodes, forcing a commodity Q_{kdl}^C to leave at least one source. Constraint (5.32) is the flow balance for the sink node, i.e. the violation port, demanding a commodity Q_{kdl}^C to arrive prior to the violation time period t' . Following are Constraints (5.33), which are the flow conservation constraints in all the transshipment nodes. Constraints (5.34)-(5.37) are all included with the purpose of connecting the IN-edge flow variables with the OUT-edge flow variables, which should be the same on all edges between i and j . The only combination not included above is the edge i to j where both i and j are source nodes. In all solutions these are also identical, however this is where the distinction between the IN and OUT edges is needed. The distinction ensures that the flow balance for the sources is feasible. Finally, Constraints (5.38)-(5.40) limit the flow to only be positive when the corresponding routing or operation variable is positive.

5.4.5 Strategy 5: Require a change in routes generated in each iteration

In order to ensure that the matheuristic converges fast towards a feasible solution, it is important that the routing problem is limited from generating the same set of routes in two separate iterations. To elaborate, if a set of routes is generated in *Iteration l* and are proven to not be feasible in the inventory problem, it should not be possible to generate this set of routes in any later iteration. One way of enforcing this restriction is to use a hop formulation by adding a hop index to the routing variable. This approach would be similar to the approach used in the inventory problem. The hop index keeps control of the sequence of port visits in every route. Thus, it can be used to demand the next iteration route solution to be different from all previous solutions. However, this formulation requires a significant increase in routing variables, as each routing variable x_{ijvt} would be created

Algorithm 5 Strategy 4: Alternative commodity flow formulation for each violation

for each (Iteration) **do**
for each (Violation) **do**

 Violation port = d , Violation product = k , Violation time = t' , Iteration = l

$$\sum_{j \in \mathcal{N}} \sum_{i \in \mathcal{N}^S} \sum_{v \in \mathcal{V}_k} \sum_{t \in \mathcal{T} | t < t' - T_{ijv}} f_{jivtkdl}^{X,IN} + Q_{kdl}^C = \sum_{i \in \mathcal{N}^S} \sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}_k} \sum_{t \in \mathcal{T} | t < t' - T_{ijv}} f_{ijvtkdl}^{X,OUT} \quad (5.31)$$

$$\sum_{j \in \mathcal{N} | j \neq d} \sum_{v \in \mathcal{V}_k} \sum_{t \in \mathcal{T} | t < t' - T_{ijv}} f_{jdvtkdl}^{X,IN} = Q_{kdl}^C \quad (5.32)$$

$$\begin{aligned} \sum_{i \in \mathcal{N}} f_{ijv(t-T_{ijv})kdl}^{X,IN} + f_{jv(t-1)kdl}^O \\ = \sum_{i \in \mathcal{N}} f_{ijvtkdl}^{X,OUT} + f_{jvtkdl}^O \end{aligned} \quad j \in \mathcal{N} - \mathcal{N}^S | j \neq d, v \in \mathcal{V}, t \in \mathcal{T} | t \leq t' \quad (5.33)$$

$$f_{ijvtkdl}^{X,OUT} = f_{ijvtkdl}^{X,IN} \quad i \in \mathcal{N}^S, j \in \mathcal{N} - \mathcal{N}^S | j \neq d, v \in \mathcal{V}_k, t \in \mathcal{T} | t < t' \quad (5.34)$$

$$f_{jdvtkdl}^{X,OUT} = f_{jdvtkdl}^{X,IN} \quad j \in \mathcal{N} - \mathcal{N}^S | j \neq d, v \in \mathcal{V}_k, t \in \mathcal{T} | t < t' \quad (5.35)$$

$$f_{idvtkdl}^{X,OUT} = f_{idvtkdl}^{X,IN} \quad i \in \mathcal{N}^S, v \in \mathcal{V}_k, t \in \mathcal{T} | t < t' \quad (5.36)$$

$$f_{jzvtkdl}^{X,OUT} = f_{jzvtkdl}^{X,IN} \quad j, z \in \mathcal{N} - \mathcal{N}^S | j, z \neq d, v \in \mathcal{V}_k, t \in \mathcal{T} | t < t' \quad (5.37)$$

$$f_{ijvtkdl}^{X,IN} \leq Q_{kdl}^C x_{ijvt} \quad i \in \mathcal{N}, j \in \mathcal{N}, v \in \mathcal{V}_k, t \in \mathcal{T} | t \leq t' - T_{idv} \quad (5.38)$$

$$f_{ijvtkdl}^{X,OUT} \leq Q_{kdl}^C x_{ijvt} \quad i \in \mathcal{N}, j \in \mathcal{N}, v \in \mathcal{V}_k, t \in \mathcal{T} | t \leq t' - T_{idv} \quad (5.39)$$

$$f_{ivtkdl}^O \leq Q_{kdl}^C o_{ivt} \quad i \in \mathcal{N}, v \in \mathcal{V}_k, t \in \mathcal{T} | t \leq t' - T_{idv} \quad (5.40)$$

end for
end for

for each hop. The size of the hop set depends on the length of the routes. Considering that the MIRP-UC we are solving is already very large, an alternative formulation is developed.

To restrict the routing problem from creating the same set of routes, a hop formulation

is introduced through an additional variable, z_{ijvh} . Hence, we can avoid the additional index on the routing variable and limit the increase in variables. The variable is referred to as a hop variable, because it is a variable keeping count of the visit number on each arc travelled by the ship.

$$z_{ijvh} \begin{cases} 1, & \text{if ship } v \text{ sails arc } i, j \text{ as the } h\text{'th arc} \\ 0, & \text{otherwise} \end{cases}$$

Let H be the set of all possible hops, i.e. the visit numbers of travelled arcs. Constraints (5.41) - (5.44) are needed for merging the hop variable into the routing problem.

$$\sum_{i \in \mathcal{N} \cup o(v)} \sum_{j \in \mathcal{N} \cup d(v)} z_{ijvh} \leq 1 \quad v \in \mathcal{V}, h \in \mathcal{H} \quad (5.41)$$

$$\sum_{j \in \mathcal{N} \cup d(v)} z_{ijvh} - \sum_{j \in \mathcal{N} \cup o(v)} z_{jiv(h-1)} \leq 0 \quad i \in \mathcal{N}, v \in \mathcal{V}, h \in \mathcal{H} | h \geq 1 \quad (5.42)$$

$$\sum_{h \in \mathcal{H}} z_{ijvh} - \sum_{t \in \mathcal{T}} x_{ijvt} = 0 \quad v \in \mathcal{V}, i \in \mathcal{N} \cup o(v), j \in \mathcal{N} \cup d(v) \quad (5.43)$$

$$\sum_{j \in \mathcal{N} \cup d(v)} z_{o(v)jv1} = 1 \quad v \in \mathcal{V} \quad (5.44)$$

The hop constraints are used to make sure that the hop variable z_{ijvh} corresponds to the behavior of the routing variable x_{ijvt} . Constraints (5.41) ensure that only one arc is associated with one hop number, while Constraints (5.42) require that hop h can only exist if hop $h - 1$ exists. Constraints (5.43) ensure a connection between the hop variable and the routing variable, while Constraints (5.44) are the start conditions.

By creating an interdependence between z_{ijvh} and x_{ijvt} , it is possible to impose restrictions on x_{ijvt} through z_{ijvh} . With the information about the hop number of each arc, it is possible to state that the hop numbers can not be identical in any succeeding iteration. However, before presenting the constraint that require change in each iteration, one important remark must be made. Constraints (5.43) are the only set of constraints in the hop formulation that connects the x_{ijvt} and z_{ijvh} variables. Thus, the only connection imposed is that the total number of hops on arc (i, j) is equal to the total number of times the ship traverses the arc (i, j) . This implies that there is no restrictions imposed ensuring that the z_{ijvh} is awarded the correct visit number other than Constraints (5.42). Hence, in theory, it is possible to rearrange the arcs into a new sequence, such that the z_{ijvh} does not get the correct sequence numbers and does not directly correspond to x_{ijvt} . As a result, when imposing that the set of solutions of z_{ijvh} must change in the next iteration, it does not necessarily impact x_{ijvt} . Hence, a change is not enforced in the next iteration. It is important to note that in the majority of cases, it is not possible to rearrange the arcs as previously described. The only cases where this is possible is when the same arc is tra-

versed multiple times. It is not possible to remove this discrepancy in the hop formulation with linear constraints given the assumptions of this thesis. However, we consider it better to have this formulation with a small degree of malfunction in some rare occasions rather than a hop index on the routing variable. The solution for these exceptions is presented after the strategy constraint is introduced.

In order to restrict the routing problem from generating the same set of routes more than once, the restriction use information about the routing solutions from earlier iterations. The following constraints demand that at least one of the binary hop variables that in previous iterations was 1 must be 0 or vice versa, at least one of the variables that was 1 must be 0. By enforcing this, the routing problem ensures that the new routes generated must be changed with at least one hop from earlier solutions. The constraints are presented in (5.45). New notation is used; $z_{ijvh}^*(l-1)$ denotes the optimal solution of z_{ijvh} in the previous iteration, $l-1$.

$$\sum_{\substack{i \in \mathcal{N} \cup o(v), \\ j \in \mathcal{N} \cup d(v), \\ v \in \mathcal{V}, \\ h \in \mathcal{H}_v | \\ z_{ijvh}^*(l-1)=0}} z_{ijvh} + \sum_{\substack{i \in \mathcal{N} \cup o(v), \\ j \in \mathcal{N} \cup d(v), \\ v \in \mathcal{V}, \\ h \in \mathcal{H}_v | \\ z_{ijvh}^*(l-1)=1}} z_{ijvh} \geq 1 \quad l \in \mathcal{L} \quad (5.45)$$

When the routing problem is not subject to the presented exception, Constraints (5.45) are binding. Hence, the routing problem is forced to create a new set of routes. In the exceptions described above, when the route of the hop variable can differ from the routing variable, the restriction fails to be binding and thus *Iteration* l of the routing problem can generate the same routes as in *Iteration* $l-1$. If the hop variable manages to rearrange the edges and still fulfill all the constraints of the hop formulation, it is important that the construction phase does not have to run a complete iteration with an identical set of routes as an earlier iteration. In order to limit this, the construction phase is designed in such a way that the route generated by the routing problem, x_{ijvt} , is checked before the solution is forwarded to the inventory problem. If the route is identical to the previously generated routes, the matheuristic return to a new iteration of the routing problem with information about the new solution of z_{ijvh} . Hence, the discrepancy in the hop formulation is solved without incurring a high time cost. This functionality is presented in a flow chart in Figure 5.6.

5.4.6 Strategy 6: Safety inventory

In the construction phase, it is essential that the routing problem manages to find representative routes for the overall inventory routing problem. It is hard to create realistic conditions in the routing problem corresponding to the conditions of the inventory problem. Since there are no inventory limits in the routing problem and the routes in the first iteration must be created solely based on valid inequalities, there is a good chance that the routes are not able to capture all the port activity. One way of mitigating this effect is to introduce a safety inventory in the routing problem, meaning that the inventory window to be respected by the routing problem is smaller than the original window. Imposing

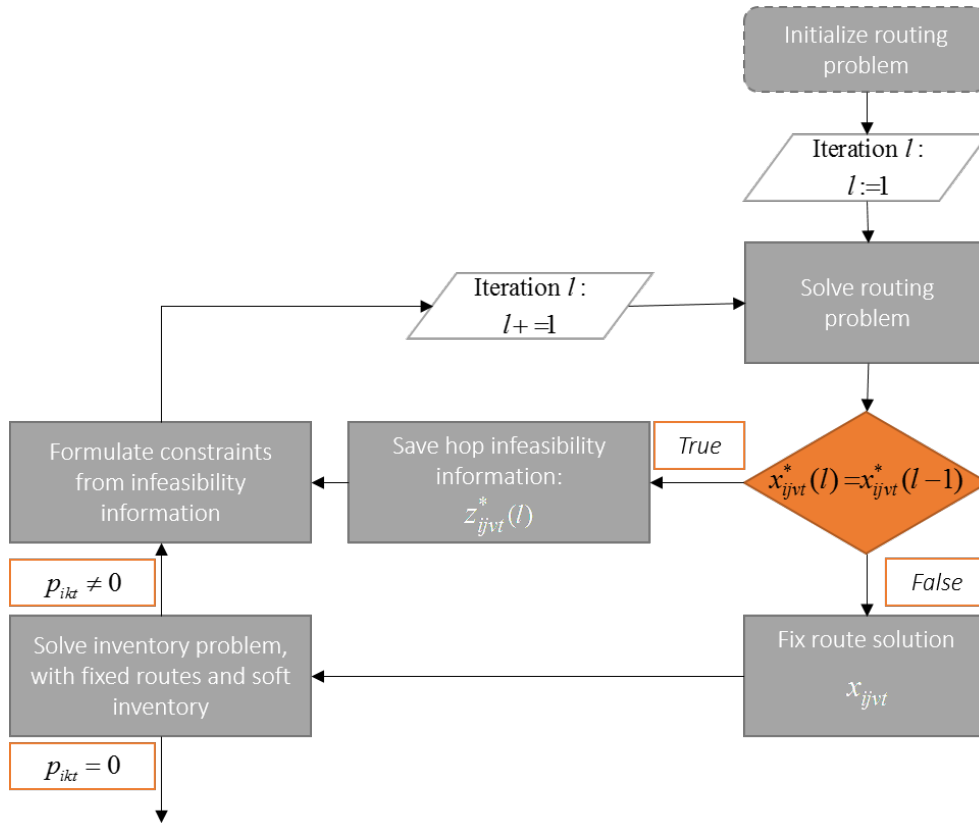


Figure 5.6: Flow chart presenting the matheuristic when Strategy 5 is included

safety inventory increases the activity to be respected in each port and the probability of violating the original inventory limits is possibly reduced. The concept of safety inventory can be applied to all iterations and is illustrated in Figure 5.7.

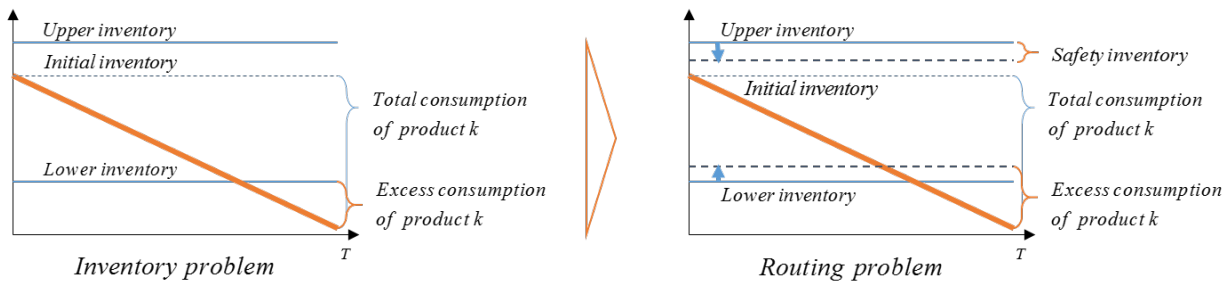


Figure 5.7: Safety inventory of routing problem

5.4.7 Additional techniques and features

In addition to the strategies presented above, we also propose some additional techniques and features to improve the chances of convergence of the construction phase. First, it is possible to impose restrictions on the run time of each restriction. Three main restrictions are proposed. In order to limit the total time of each iteration, the time used by each iteration is limited. Second, the global search is stopped when a MIP solution has been found and the Optimizer can guarantee that it is within $x\%$ of the optimal solution. Finally,

to increase the efficiency of the overall matheuristic, a third time constraint is imposed in the Optimizer. If the time since the last discovered MIP solution is greater than a defined threshold, the current iteration search is terminated. The goal of imposing these restrictions on running time is to achieve efficiency in the search for a feasible solution. The setting of these parameters is further elaborated on in Section 6.2.2.

Further, actions have been taken to increase the efficiency of the inventory problem. The idea of the following feature is to ensure that the violations are likely to occur in the same time span of the planning horizon. First, it is very hard to adjust for many violations occurring in the earliest time periods of the planning horizon. In all strategies of the routing problem, we require that all violations that occurred in all previous inventory problems are accounted for. Thus, if there are enough violations in the different ports early in the planning horizon, it might be infeasible for the succeeding routing problem to construct routes with respect to these violations. Violations occurring the very last time periods of the planning horizon can also be hard to resolve. This is because in all the strategies presented above, some conditions are imposed to the routing problem for each violation. However, all the conditions only ensure that some action happen prior to the violation time. In other words, there is no guarantee that the action taken to resolve the violation happen close enough to the violation. Thus it might not resolve the violation and the violation returns in the next iteration. To account for this, we impose a time dependent violation cost. The middle time periods are less expensive since violations occurring in those time periods are easier to resolve. This measure does not prevent violations from occurring the early and late time periods, however, it comes at a cost and the occurrence of violations in these time periods is reduced. By differentiating the time periods the violation occurs, we also manage to reduce the symmetry in the violation dimension.

5.5 Improvement phase

The construction phase terminates once it finds a feasible solution to the original problem. With such a scheme, there is no guarantee on the quality of the solution. Due to this uncertainty, an improvement phase is added to the matheuristic.

The idea is to explore the neighborhood around the feasible solution. In general, it is difficult to only make small adjustments to the feasible solution of a MIRP and still adhere the inventory limits of all products in all ports. Consequently, the neighborhood is always quite large and the difference between neighbors can be significant. We propose three different operators to create the neighborhood to be explored.

The improvement heuristic always starts with a feasible solution found by the construction phase of the matheuristic. In order to be able to find improved solutions, some parts of the given feasible solution must be opened up. When designing the different operators for the neighborhood search, the essential decisions to be made are precisely which parts of the solution to improve and which parts to keep fixed to the current solution. It is important to keep in mind the balance between the degree of freedom introduced and the corresponding

complexity. The goal is to improve the feasible solutions within an acceptable amount of time.

We propose an iterative improvement scheme that is designed in a first improvement fashion. In each iteration, certain components of the solution are open for improvement and the remaining parts are fixed. An iteration is terminated a given number of seconds after the first improving solution is found. The reason for including this time buffer is to capture the possibly better solutions found in close proximity to the first discovered solution. When an iteration is terminated due to the discovery of a better solution, the newly discovered solution is saved, both in terms of objective value and route solution. This newly saved solution is used as input for the next iteration of the improvement heuristic.

With this functionality in mind, the three operators are presented.

5.5.1 Intra ship route improvement

The first operator, Intra, focus on optimizing the routes of each ship separately. In each iteration, all but one ship is fixed to the currently best solution. The ship that is open for improvement is fixed to the port visits of the route, but neither the sequence nor the number of visits to each port is fixed. The purpose of this operator is thus to optimize the utilization of each ship given the port visits of the original solution. This operator never gives a solution where the ships visits other ports than the ports visited in the input solution.

The number of ships in the fleet, defined as \bar{V} , is used as the minimum of number of iterations executed. This means that if the test case being used has a fleet of 4 ships, a minimum of four iterations is executed. After \bar{V} iterations, the improvement heuristic has iterated over each of the ships. The Intra operator continue to iterate until a predefined termination criteria. The termination criteria for the Intra operator is discussed and defined in the Section 6.2.5. Figure 5.8 illustrates the functionality of the Intra operator. In *Iteration 1*, *Ship 1* is subject for improvement, marked in orange. Equivalently, the route of *Ship 2* can be improved in *Iteration 2*, marked in blue.

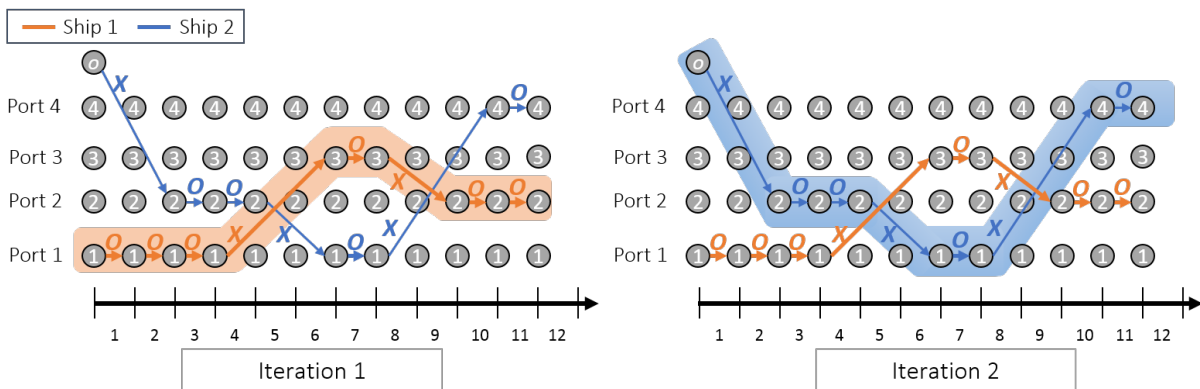


Figure 5.8: Illustration of the Intra ship improvement operator

5.5.2 Time-constrained inter ship route improvement

While the first improvement operator only improves the schedule of one ship at a time, the Inter operator allows for improvements across ship schedules. By employing time intervals to open parts of the solution in each iteration, the Inter operator iteratively search for improvements in highly constrained versions of the original problem.

In each iteration of the Inter execution, a time interval is defined where the routing-, operating-, and waiting-variables are free for all ships. This means that all variables for all port, ship, and time period combinations are created. Hence, inside the range of this interval, the model functions similarly as the original model and improvements can be made with respect to the ship routing and scheduling, inventory management in ports and the allocation of products.

For all time periods outside the time interval of the current iteration, all variables are created based on the current best solution. The ship-port combinations from the current best solution are kept fixed for all variables outside the time interval. However, a slack is introduced on which time periods the combinations can occur. The time slack is introduced to be able to adjust to the changes across ship schedules that are possible within the time interval. The time slack parameter is defined as T^{Slack} . For example, if $T^{Slack} = 1$ and *Ship 2* operates in *Port 4* in time period six in the current best solution, the following operation variables are created: o_{425} , o_{426} and o_{427} .

Apart from the adjustments that can be made due to the time slack, the solution is kept fixed outside of the time interval. However, it is always optional for a ship to utilize the arc going to $d(v)$ according to the fixed route. This does not mean that a ship does not have to end its schedule. It only facilitates a potential adjustment within the time interval that involves a change in the last port visit. Hence, a new arc to $d(v)$ is needed within the time interval. In that case, the original arc to $d(v)$ in the fixed solution does not have to be used. The procedure for the Inter ship route improvement operator is illustrated in Figure 5.9. The arcs colored in orange are the only arcs that can be changed and that are subject for improvement.

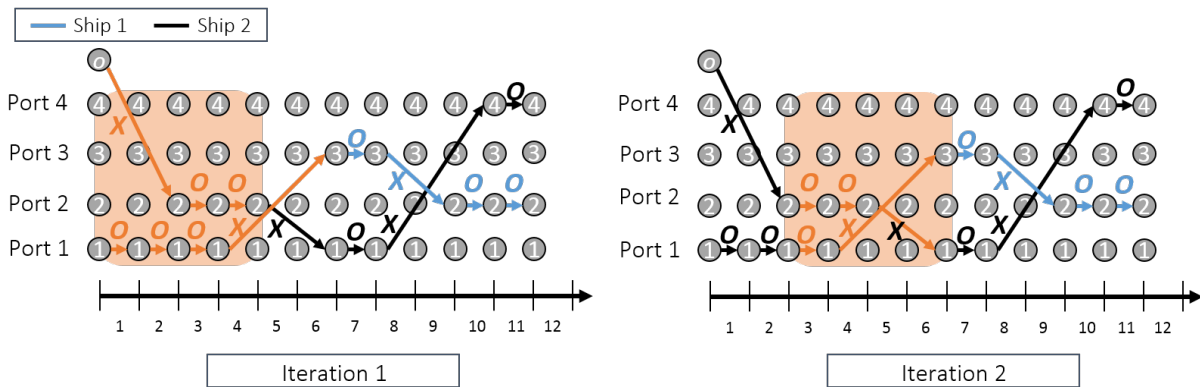


Figure 5.9: Illustration of the Inter ship improvement operator

One iteration of the Inter operator corresponds to running the improvement heuristic for one specific time interval. To test all improvement possibilities, the improvement heuristic changes the time interval in each iteration. For each iteration, the current best solution is saved and used as the basis for further improvements.

The start of the interval shifts with a predefined number of time periods in each iteration. However, the interval shift is smaller than the length of the time interval and so the intervals overlap. For example, if the first time interval is $[1,10]$ and the interval shift is 5 time periods, then the second time interval is $[6,15]$. This overlap between consecutive time intervals is constant during an execution of the heuristic.

For the Inter operator, the number of intervals needed to cover the planning horizon, \bar{I} , is used as the minimum number of iterations to be executed. The number of intervals, \bar{I} , depends on the length of the time interval and how much the interval shifts for each iteration. Equivalently as for the Intra operator the termination criteria is defined in Section 6.2.5.

5.5.3 A combination of the Intra and Inter operators

The Combination operator incorporates the Intra operator and Inter operator in one execution of the improvement heuristic. Intra optimizes each ship route by rearranging each ship's port visits or shortening the ship route, while Inter can change which ships that serve which port-product combinations. When combining these operators, if either operator finds a new solution, the neighborhood of this solution is reachable for both. This opens up parts of the feasible area to the operators not attainable through executing the operators independently. By combining these two in one improvement scheme, even further improvements to the solution can be found.

When employing the Combination operator, the Intra and Inter operator is executed in that order. Hence, the heuristic first systematically iterates through all the ships with the Intra operator, and then all possible time intervals using the Inter operator. The heuristic continues to execute the operators sequentially in that order until the termination criteria is met. As already mentioned, termination criteria is defined and further discussed in Chapter 6.

Chapter 6

Computational Study

The purpose of this chapter is to present an overview of the results from the testing of the MIRP-UC with both an exact solution method and the matheuristic solution method presented in Chapter 5. When using the term exact solution method, we refer to solving the MIRP-UC with the solver in Mosel Xpress-MP. For testing the model with the exact solution method and the valid inequalities proposed in Section 4.3, three test cases of different sizes are utilized. When testing the matheuristic method, additional test cases are used to achieve a comprehensive evaluation of the alternative solution method. This chapter is organized in three main sections. First, in Section 6.1, we present the results from the exact solution method and the effectiveness of the proposed valid inequalities. Following is Section 6.2, presenting the results of the matheuristic solution method. Lastly, Section 6.3 discusses practical implications of the formulation and the different solution methods.

All test instances of our mathematical programming models are solved using Mosel Xpress-MP. Mosel Xpress-MP is run on a Hewlett Packard 64-bit Windows 7 Enterprise PC with Intel(R) Core(TM) i7-3770 3.40 GHz processor and 16,0 GB (15.9 GB usable) RAM.

6.1 Exact solution method

First, the test instances and parameter settings are presented, along with the corresponding problem sizes. Following is a presentation of the test results of the MIRP-UC model and the proposed valid inequalities. For each test case, we have tested the valid inequalities independently, as well as in combination with each other.

6.1.1 Test instances and notation

The name of each test instance is built up of two components; test case and valid inequalities included. An overview of all abbreviations used in Section 6.1 is presented in Table 6.1. UC refers to the original model solved exactly, without any valid inequalities. MV, MCP, MO, F, and L refer to the valid inequalities presented in Section 4.3. For example, the test instance M_MO refers to the model with the medium-sized test case and the valid inequality on minimum number of operation periods in each port added.

Table 6.1: Notation for exact solution method test instances and valid inequalities

Models	Notation
Undedicated compartments	UC
Test cases	Notation
Small test case: 2 ships 4 ports 3 products	S
Medium test case: 3 ships 6 ports 4 products	M
Large test case: 4 ships 8 ports 4 products	L
Valid Inequalities	
No cuts	UC
Minimum number of visits per port with ship sequence [4.25]	MVs
Minimum number of compartments per product with ship sequence [4.27]	MCP
Minimum number of operation periods per port [4.29]	MO
Timing of first visit to a port [4.30]	F
Timing of last visit to a port [4.34]	L
All cuts remaining at the time of use	A

The only valid inequality presented in Section 4.3 not included in Table 6.1 is the timing of the second visit constraint. With the current test cases, there is not enough activity in any port to be able to determine the time of a second visit validly. Hence, the performance results of this valid inequality is not included in the following sections.

For the smallest test case, we present the optimal integer solution, time to optimality or gap if optimal solution is not found for the test instance. For the medium and large test cases, we present the best integer solution found, best bound and optimality gap. We also present the LP bounds for all of the test instances. The LP bounds, best solutions and best bounds are given in absolute value, the optimality gap is given in %, and time to optimality in the small test case is given in seconds. A bold font is used to identify the best solution and best lower bounds in the tables of results. Other important notation is the dash, '-', which means that no integer solution is found within the predetermined running time and 'N/A' for not applicable results.

6.1.2 Parameter settings

Due to time and computational power resource constraints during the work on the thesis, it is important to ensure that the resources are used optimally. Prior to any empirical results, we suggested 10 000 seconds to be a reasonable running time to both get sufficient

results and to use the resources efficiently. However, when running instance L_UC , we saw that the last integer solution was found prior to 3 000 seconds running time. This is illustrated in Figure 6.1. Hence, we can limit the maximum running time of our models substantially compared to a maximum running time of 10 000 seconds. This observation also applies to the other test instances. Due to the large number of test instances to run, the maximum running time is set to 5 000 seconds for all test instances.

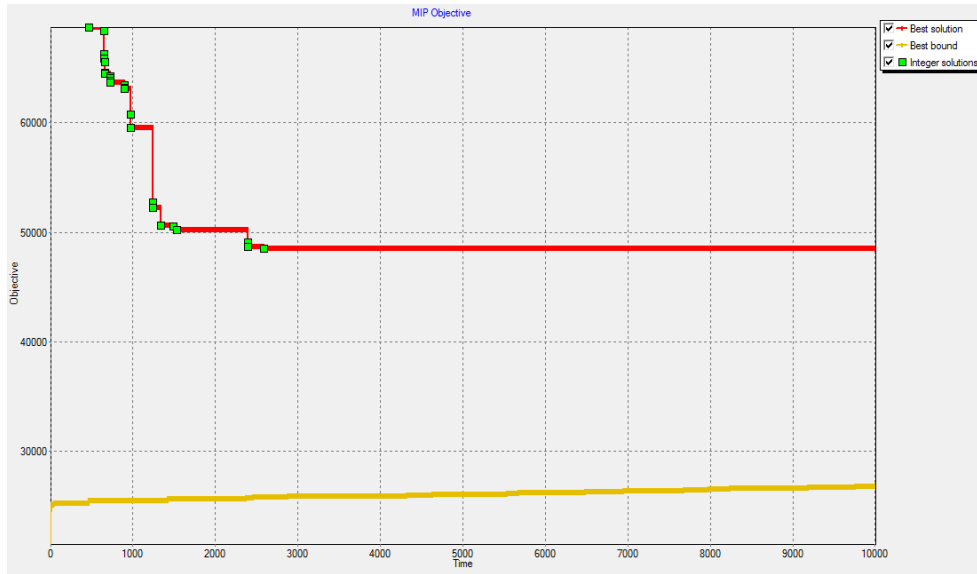


Figure 6.1: Discovery of feasible solutions in Mosel Xpress-MP

Several of the valid inequalities presented in Section 4.3 apply the use of a time interval when deciding the binding capacity of a ship or a compartment in the ship/compartment capacity sequence. Preliminary testing showed that using the entire length of the planning horizon as the length of the time interval gives the tightest formulation and thus the best results. This is also evident in earlier studies by Andersson et al. (2015). Consequently, the length of the planning horizon is used as the length of the time interval in all succeeding tests.

Another decision to be made when implementing the valid inequalities employing time interval is the starting period of the time interval. In theory, there could be multiple time intervals starting at different times. However, the results presented by Andersson et al. (2015) suggest that having the time interval start in the first time period is most beneficial. We have limited the testing to a single time interval and defined it to start in the first time period of the planning horizon, i.e. $\underline{T}' = 1$. Consequently, the incoming inventory level $s_{ik}(\underline{T}'-1)$ of the time interval is always equal to the initial inventory level of the product, S_{ik}^0 , and not the variable inventory level, i.e. s_{ikt} . Hence, of the alternative formulations of the valid inequalities employing a ship/compartment capacity sequence defined in Section 4.3, only valid inequalities 4.25 and 4.27 are applicable.

6.1.3 Problem Sizes

Table 6.2 shows the problem size of each test instance. The problem sizes are given before and after Presolve. Presolve is a function integrated in Xpress Optimizer where one of the purposes is to reduce the number of redundant variables and constraints, and thus reduce the complexity of the problem and improve running time. From the table, we see that the Presolve function is successful in eliminating both constraints and variables redundant to the problem when solving the formulation with Presolve. The fact that Presolve have managed to eliminate a large number of variables suggests that we should have used more time and focused more on eliminating unnecessary variables in the process of implementation. That could have been beneficial for the efficiency of the solution process.

Table 6.2: Problem size of exact solution method instances

	Presolve	S	M	L
Total number of variables	Before	4427	11401	21992
	After	3702	9633	18685
Number of constraints	Before	3895	8078	12114
	After	2252	4745	7888

The small test case consists of two ships, four ports. When looking at the size of the medium test case, which consists of three ships and six ports, the number of variable of the problem more than doubles compared with the small-sized test case. A similar pattern can be seen for the large test case, compared with the medium test case.

6.1.4 Exact solution method results

The performance results of the test instances are presented in Table 6.3. As justified in Section 6.1.1, all instances have a maximum running time of 5 000 seconds. Hence, a test instance is either run to the maximal running time or to optimality. All test instances are run with the Presolve function integrated in the Xpress Optimizer. One part of the functionality of Presolve is to generate cuts and thus improve the bounds of the problem prior to the branch-and-bound search. When disabling Presolve, the tree to search through becomes much larger and the bounds weaker. Running the models without Presolve, and thus without non-specified improvements, would have shown the true effect of the valid inequalities. However, the difference in performance before and after Presolve is not a focus area in this thesis.

Table 6.3: Results of small-, medium- and large-sized test cases with different valid inequalities. Running time 5000 seconds

Test case	Info	UC	MV	MCP	MO	F	L
S	LP bounds	16 257	18 975	25 199	16 510	16 257	16 257
	Best solution	29 883	29 883	29 883	29 883	29 883	30 010
	Best bound	29 883	29 883	29 883	29 883	29 883	28 772
	Gap	-	-	-	-	-	4,12%
	Time to optimality	2 066 s	1 749 s	1 032 s	1 325 s	848 s	N/A
M	LP bounds	19 016	21 589	25 128	19 463	19 059	19 016
	Best solution	35 633	42 708	36 673	34 948	37 446	37 593
	Best bound	26 018	25 731	27 544	25 946	26 689	25 035
	Gap	26,9%	39,8%	24,9%	25,8%	28,7%	33,4%
L	LP bounds	24 473	28 293	34 391	24 527	24 688	24 473
	Best solution	64 930	57 251	61 703	62 360	61 717	62 733
	Best bound	30 902	30 629	34 933	30 886	32 684	31 139
	Gap	52,4%	46,5%	43,4%	50,5%	47,0%	50,4%

LP bounds

In Table 6.3, we see that all the valid inequalities but L can be categorized as a cut. The LP bound performance of all the valid inequalities follow approximately the same trend in all test instances. *MCP* has the largest impact on the LP bound and is thus the tightest cut.

The addition of a valid inequality is always a restriction of the feasible region. Hence, an LP bound can never be reduced when adding a valid inequality to a model, because a reduction in the LP bound would be equivalent to a relaxation of the problem. However, a reduction in LP bound after the inclusion of a valid inequality can be seen in Table 6.3 for multiple test instances of the medium-sized test case. The only possible reason for this is our use of the Presolve function in Xpress. With Presolve, Xpress always tries to optimize which constraints are included in the running of a model. Hence, in some case, it can end up choosing a set of constraints actually relaxing the problem resulting in a lower LP bound.

We strive for a tighter formulation because it enables a smaller search space. The result of a smaller solution space is a smaller branch-and-bound tree and possibly a shorter solving time. However, a tighter formulation often comes at the expense of an increase in the number of variables. As a result, even though the size of the whole branch-and-bound

tree decreases, the solution time in each branch-and-bound node increases. We now look at the running time performance to see how the complexity added with the valid inequalities affects the running time.

Minimum number of visits with ship capacity sequence (MV)

The purpose of this valid inequality is to impose a lower bound on the number of visits to each port by employing a ship capacity sequence when determining maximum capacity of a ship. When imposing this lower bound, the optimal LP solution is improved, making the model tighter. The bound in the root node of the branch-and-bound tree is thus higher, resulting in a reduced search space and a more efficient branch-and-bound procedure. In S_MV this is evident as the time to optimality is improved from S_UC . L_MV returns a smaller gap than L_UC due to the discovery of a better feasible solution.

Minimum number of compartments per product with ship capacity sequence (MCP)

MCP gives the strongest LP bound of all the valid inequalities proposed. MCP reduces the gap between the LP bound and the optimal integer solution of the small test case from 45,6% to 15,9%. It is reasonable to believe that this reduction can explain the great efficiency of the MCP addition. S_MCP gives a significantly better time to optimality and is the cut with the second best performance on the small test case. M_MCP has the best performance in terms of the highest bound and the lowest gap. For the large test case, MCP continue its good performance giving the strongest lower bound and correspondingly the smallest gap.

Minimum number of operation periods (MO)

This valid inequality is imposed with the intention of creating a lower bound on the minimum number of operation periods needed by each port. MO can be characterized as a cut since it removes the optimal LP solution making the formulation tighter. MO significantly improves the time to optimality in S_MO . However, for the larger test cases it does not manage to contribute as notably to the efficiency.

Timing of first visit in port (F)

F is included with the intention of limiting the number of options in the time dimension by forcing a first visit prior to a specific time period. It does not manage to tighten the formulation by improving the LP bound in the small test case, and it only tightens the LP bound by a small fraction for the larger test cases. However, F performed best in the smallest test case and offered the shortest time to optimality. In the smallest test case, it manages to contribute to efficiency, however, the contribution from F in the larger test cases is not equally essential.

Timing of last visit in port (L)

L is added with the same intention as F. It does not improve the LP bound of the test cases, and it is the only valid inequality preventing the small test case from finding the optimal solution within 5000 seconds. For the largest test case, L contributes to finding a higher best bound.

6.1.5 Results on combination of valid inequalities

Until now, the valid inequalities have only been tested independently. However, it is at least equally interesting to test different combinations of the valid inequalities and see how it affect the efficiency of the exact solution method. First, we test all valid inequalities combined, followed by this combination excluding one of the valid inequalities in turn. S_A denotes all valid inequalities are included in the small test case, while S_A_MV denotes all valid inequalities except for MV are in the combination of the small-sized test case. The performance results of the MIRP-UC with these combinations can be found in Table 6.4.

Table 6.4: Results on combination of valid inequalities for all test cases. Running time 5000 seconds

Test case	Info	A	A_MV	A_MCP	A_MO	A_F	A_L
S	LP bounds	25 711	25 711	19 443	25 472	25 435	25 711
	Best solution	29 883	29 883	29 883	29 883	29 883	29 883
	Best bound	29 883	29 883	29 883	29 883	29 883	29 883
	Time to optimality	516 s	516 s	262 s	1 357 s	444 s	367 s
M	LP bounds	25 467	25 467	22 044	25 128	25 465	25 467
	Best solution	35 461	36 867	37 010	38 903	36 918	35 937
	Best bound	28 441	28 402	26 042	27 135	27 879	27 752
	Gap	19,8%	23,0%	29,6%	30,3%	24,5%	22,8%
L	LP bounds	34 396	34 396	28 445	34 395	34 391	34 395
	Best solution	65 253	-	65 533	62 040	-	-
	Best bound	35 039	35 231	31 919	35 106	34 973	34 952
	Gap	46,3%	-	51,2%	43,4%	-	-

When testing the valid inequalities independently, MCP had the tightest LP bound in all test cases. The LP bounds reported in Table 6.4 show how the LP bound is improved in all

test instances where MCP is included. The combination where MCP is left out, A_MCP consistently return the weakest LP bound. It is clear that MCP has the largest impact on the LP bound. This is in line with the increase of the LP bound when only including MCP.

For the small test case, the running times to optimality are significantly improved in all the combination test instances. $S - A_MCP$ gives the shortest running time to optimality, which is somewhat surprising given the independent high quality performance of MCP. This deviating result is most likely due to the inclusion of the Presolve function in Xpress. As expected, the exclusion of L in $S - A_L$ has a positive impact on the efficiency.

$M - A$ is the only test instance in the medium test case able to find a better solution than M_UC , however only by a small margin. None of the combination instances are able to find a better solution than M_MO . Several of the instances with different combinations of valid inequalities achieves higher bounds than the best reported bound in Table 6.3. On average, the combination test instances have an improved performance compared to the independent testing of the valid inequalities for the medium-sized test case.

In the large-sized test case, the performance of the combination test instances are not enhanced compared to the independent testing. $L - A$, $L - A_MCP$, and $L - A_MO$ are the only test instances able to find a feasible solution within the 5000 seconds of running time. Of these, only $L - A_MO$ discovers a better solution than L_UC . However, all combination test instances present a notable improvement of the lower bound. It is hard to explain why removing MV, F or L disables the model to find a feasible solution within the 5000 seconds when several feasible solutions are found when all valid inequalities are included. $L - A$ should be the test instance with the highest degree of complexity. It is also hard to explain why removing MCP and MO does not disable the model from finding a feasible solution, when especially MCP is essential for tightening the model. Again, it is likely that one explanation for these diverging results is the inclusion of the Presolve function in Xpress. Anyway, it is clear that adding multiple valid inequalities when solving the large-sized test case increases the complexity significantly. For the large-size test case, the increase in complexity outweigh the tightening of the formulation.

6.1.6 Concluding the exact solution results

As evident in Table 6.3, the valid inequalities MV, MCP, and MO all enhance the performance of the model for all test cases. In the small-sized test case, all the valid inequalities manage to improve the time to optimality independently. The effect of the valid inequalities on the LP bound in the medium-sized test case is somewhat unclear due to the case of Presolve relaxing the problem. However, for the large-sized test case, all the valid inequalities have an exclusively positive effect on the optimality gap. It is apparent from the performance results in Table 6.3 that MCP is the overall best performing valid inequality.

Table 6.4 shows an overall enhancement in performance for both the small- and the

medium-sized test case when using different combinations of valid inequalities. However, for the large-sized test case, the increase in complexity from the inclusion of multiple valid inequalities prevent the model from improving its performance compared to using the valid inequalities independently.

Even though MCP stands out as the best performing valid inequality, on average, the search for the optimal solution is most efficient when including all the presented valid inequalities. The performance of A is superior to the performance of MCP for both the small- and the medium-sized test cases. MCP has a slightly better performance than A on the large test case, however, this is outweighed in the overall performance. Hence, the standard setting for the model is the original MIRP-UC formulation in combination with all the presented valid inequalities. In all further tests, the model is solved with this setting.

6.2 Matheuristic solution method

In Chapter 5, a matheuristic solution method was introduced as an alternative to the exact solution method. The purpose of this section is to test and analyze the potential of the matheuristic. First, the strategies presented in Section 5.4 are further explained before relevant combinations are tested. Along with the explanations of the functionality presented in Chapter 5, a presentation of parameter settings is given. Section 6.2.3 presents the performance of the construction phase of the matheuristic with the different strategy settings. The performance of the matheuristic is evaluated in relation to the performance of the exact solution method. Finally, the improvement phase of the matheuristic is tested and the results are presented in Section 6.2.5.

6.2.1 Construction phase: Defining the strategy settings

As previously explained, a decomposition of the MIRP into a routing problem and an inventory problem lies at the core of the matheuristic. A solution to the MIRP-UC is found by iterating between solving these two problems, converging towards a feasible solution. Multiple techniques for guiding the iterations have been proposed in terms of strategies in the previous chapter. The purpose of this subsection is to define relevant and interesting strategy settings to be tested. The term strategy setting is used to denote a specific combination of strategies.

Before introducing the strategy settings tested, some final notes on the strategies presented in Chapter 5 have to be made. The following solution strategies are proposed:

1. Strategy 1: Increasing the number of arcs required to the violation port prior to the violation time
2. Strategy 2: Increasing the minimum number of operation periods in violation ports
3. Strategy 3: Introducing a commodity flow formulation for each violation

4. Strategy 4: Alternative commodity flow formulation for each violation
5. Strategy 5: Require a change in routes generated in each iteration
6. Strategy 6: Safety inventory

When forcing an arc directly between a production/consumption port and the consumption/production violation port, all feasible solutions where the origin port only precede and not directly precede the violation port are removed. Thus, there is no guarantee that the optimal solution remains in the feasible region. Strategy 1 is a greedy strategy where the greedy choice is to impose a direct arc to the violation port.

The aim of Strategy 2 is to impose additional operation periods needed to account for the violation amount in the violation port. Only the sequence of port visits is saved between the routing problem and the inventory problem. Hence, a strategy only impact the solution if it requires change in the route and not only the schedule. During preliminary testing of Strategy 2, it was evident that imposing additional operation periods was not always restrictive enough to alter the sequence of port visits. As a result, the convergence of the matheuristic was slow. However, due to the nature of the strategy as a valid inequality, it can be included in combination with other strategies.

Strategy 3 can be viewed as a more valid version of Strategy 1. The constraint imposed is that a commodity source port must be visited prior to the violation port but not directly prior to the violation port. Some uncertainty is still present on whether the integer feasible area is reduced. However, compared to Strategy 1, Strategy 3 is more consistent with the use of valid inequalities. Strategy 4 has the same goal as Strategy 3, however it makes use of an alternative and experimental formulation of the commodity flow.

Strategy 5 imposes an obvious wish for this matheuristic, namely that the routes generated in each iteration is never generated more than once. Independently, this strategy only require one arbitrary change in the set of routes in each iteration, making the convergence of the construction phase slow. However, this strategy can be essential in combination with other strategies. It can prevent a situation where no new constraints are imposed in each iteration as a result of returning violations and that the set of routes generated are identical.

As mentioned, all strategies can be combined into different strategy settings. The purpose of the strategies is to handle the violations caused by the current routes in the inventory problem. In the case of returning violations, different actions can be taken. One idea is to allow greedy choices to be made. A specific example could be that Strategy 3 is used to handle the violations and to ensure convergence, but in the case of a returning violation, the violation is handled using Strategy 1.

We now have all the fundamental components for understanding how the violations from the inventory problem are handled in the routing problem. Hence, we can introduce the different strategy settings to be tested and the following notation. The set of strategy

settings reflects the combinations of strategies we find interesting and most promising.

Strategy settings. The different strategy settings that are chosen for testing are presented in Table 6.5. Each strategy setting is named and presented with corresponding solution strategies from Section 5.4.

Table 6.5: Notation for matheuristic solution method strategy settings

Construction approach	Solution strategies	Notation
Greedy strategy	Strategy 1	G
Valid approach 1	Strategy 2,3,5	V1
Valid approach 1 w/ safety inv.	Strategy 2,3,5,6	V1s
Valid approach 2	Strategy 3,5	V2
Valid approach 2 w/ safety inv.	Strategy 3,5,6	V2s
Hybrid approach	Strategy 2,3 & Strategy 1 on returning vio.	H
Hybrid approach w/ safety inv.	Strategy 2,3,6 & Strategy 1 on returning vio.	Hs
Valid alternative	Strategy 2,4,5	VA
Valid alternative w/ safety inv.	Strategy 2,4,5,6	VAs

The name of each test instance used in all preceding tests is built up of two components. The first component is the strategy setting used, and the second component is the test case. For example, G_S is small test case solution of the construction phase employing the greedy strategy setting.

6.2.2 Construction phase: Parameter setting

One purpose of implementing the proposed matheuristic is to increase the run time efficiency compared to the exact solution method. Along with the basic functionality of the matheuristic, several parameters are needed to facilitate a fast running construction phase. In this section, we elaborate on the already presented parameters and introduce new relevant parameters. The purpose of this section is to motivate the decisions made on the relevant parameters, both through testing and theoretical reasoning. It is worth mentioning that extensive testing has been done, however, only the most relevant results are presented. Even though a great deal of testing has been done, it is nowhere near complete. Ideally, the parameter settings should have been tested on a larger number of test instances. In that way the settings could have been determined with statistical methods. Due to both time and computational power constraints, the number of tests are limited. Also, we have chosen to test the dimensions that we consider most interesting and relevant for the thesis.

Time limits

As presented in Section 5.4, introducing time constraints and limits can help increase the efficiency of the construction phase. As a result, we have chosen to impose a max time for each iteration to limit unnecessary time used on each iteration. During preliminary testing we observed that a time limit of 500 seconds for each iteration of either the routing problem or the inventory problem is suitable. If the optimal solution of the iteration has not been found within 500 seconds, it is likely more beneficial to go to the next iteration where new information is supplied rather than to continue further. However, with a shorter time limit for each iteration, the iteration might be terminated prematurely before currently available information is utilized to its fullest.

The next relevant parameter is a time limit on the amount of time since the discovery of last feasible solution. It is necessary to limit situations where no new feasible solutions are found in a long period of time. During these situations, the execution of the iteration is inefficient and little progress is made. After testing the construction phase using different time limits since the last discovered feasible solution, we concluded with 200 seconds as a reasonable time limit. Lastly, each iteration is terminated when the optimality gap is at 1%. This is because it is not necessary to prove optimality in each iteration. The difference between a solution with a 1% gap and the optimal solution is insignificant in this scheme.

First iteration parameters

Another dimension which is important to consider is the handling of the first iteration. The first iteration routing problem is the initialization of the construction phase. It is unlikely that the first iteration routing problem is constrained such that it can find a set of feasible routes immediately. Hence, whether the first iteration routing problem should be run to optimality or be terminated earlier is a relevant question. In later iterations, the quality of the routes are likely higher due to the addition of the violation handling constraints. Hence, the parameter settings controlling the first iteration routing problem and the subsequent inventory problem must be evaluated.

Before we continue to elaborate on the parameter settings of the first iteration routing problem and inventory problem, an important phenomenon of this matheuristic scheme must be explained and clarified. When separating the routing problem and the inventory problem, it is impossible for the routing problem to adopt all inventory considerations made by the inventory problem. The routing problem can, in the first iteration, generate respectable routes as a result of a series of valid inequalities. However, not to the same degree as the inventory problem. This implies that it cannot be known which of the feasible solutions found by the routing problem in the search for the optimal solution of the routing problem is the optimal solution in the inventory problem. The discovery of feasible route solutions in the routing problem in one iteration is not necessarily strictly improving in relation to the inventory problem. In one iteration, the best set of routes for the inventory problem might be the third best route solution found by the routing problem. In an other iteration

it might be the optimal route solution of the routing problem that indeed is the optimal route solution in the inventory problem. Since the routing problem presents no guarantee that the next route solution discovered is improved from the currently best solution with respect to the original problem, it is difficult to both test and decide the parameter settings.

The greedy approach. In Section 5.4, we presented several strategies for handling violations from the inventory problem in the routing problem. The parameter setting for the first iteration of the routing and inventory problem depends on the strategy used. This can be explained by the fact that the strategies depend differently on the quality of the route solution of the first iteration. A greedy strategy is likely more dependent on a good quality in the first iteration route solution than more valid strategies. The greedy approach, only impose strict requirements through adding direct arcs. Such a greedy scheme does not facilitate much flexibility to change none-violation components of the initial route solution. Hence, for the greedy strategy, it is important to find good initial route solutions. Since the measures taken by the next iteration routing problem are strict, it is also likely that the specifics of the violations in the inventory problem should be close to optimal. The probability of a good second iteration route solution should thus increase when having the "correct" violations from the inventory problem. With this theoretical reasoning in mind, we look at the test results for the greedy strategy with different combinations of parameters for the first iteration. The termination of the routing problem is tested for four different settings, differing on percentage gap and the number of feasible solutions demanded. The inventory problem is only tested with either no settings or a termination on 5% gap. The parameter test results can be found in Table 6.6.

From the test results presented in Table 6.6, it is evident that terminating the first iteration routing problem at the first feasible solution is the most efficient parameter setting. However, the quality of the solution found is affected by the choice of the first feasible solution. This is along the lines of the theoretical reasoning presented above. The performance of the greedy strategy is improved when having a better first iteration route solution. In the medium test case, terminating the first iteration inventory problem early positively affects the running time while ensuring the same quality on the returned solutions. This shows that ensuring that the inventory problem returns the "correct" violations does not necessarily have an effect on the quality of the final solution. From the test results available, the parameter setting facilitating the best overall performance of the greedy approach is breaking the first iteration routing problem at a 10% gap or 10 feasible solutions and breaking the first iteration inventory problem at 5% gap. We use this parameter setting in all further tests with the greedy strategy.

The remaining approaches. For the remaining strategies introduced, there is a larger degree of freedom in the measures imposed in later iterations of the routing problem to overcome the violations from the inventory problem. Hence, it is reasonable to assume that these strategies are not as dependent on the quality of the first iteration routing problem solution as the greedy strategy. Consequently, it could be beneficial for the efficiency of these instances to break the first iteration routing problem as early as possible. Table

Table 6.6: Parameter testing: First iteration, Greedy strategy

Test size	Param - route	Param - inv.	Sol.	Nr. iter.	Total Time	Time-1st route	Time - 1st inv.	Tot. time - route	Tot. time - inv.
M	None	None	38 650	3	292 s	9 s	147 s	12 s	279 s
	First feas.	None	39 553	4	86 s	1 s	25 s	16 s	70 s
	20% gap / 5 feas. sol.	None	38 830	3	281 s	4 s	200 s	12 s	269 s
	10% gap / 10 feas. sol.	None	38 210	3	238 s	8 s	200 s	17 s	223 s
	5% gap / 15 feas. sol.	None	39 503	4	359 s	9 s	200 s	20 s	339 s
	First feas.	5% gap	39 543	4	58 s	1 s	10 s	18 s	40 s
	20% gap / 5 feas. sol.	5% gap	38 830	3	239 s	4 s	157 s	15 s	149 s
	10% gap / 10 feas. sol.	5% gap	38 210	3	164 s	8 s	125 s	12 s	227 s
	5% gap / 15 feas. sol.	5% gap	39 503	4	357 s	9 s	200 s	19 s	339 s
L	None	None	50 586	2	215 s	179 s	6 s	199 s	16 s
	First feas.	None	50 340	3	171 s	17 s	22 s	94 s	76 s
	20% gap / 5 feas. sol.	None	51 776	3	498 s	29 s	58 s	308 s	191 s
	10% gap / 10 feas. sol.	None	50 586	2	116 s	80 s	6 s	100 s	16 s
	5% gap / 15 feas. sol.	None	50 586	2	215 s	179 s	6 s	199 s	16 s
	First feas.	5% gap	51 360	3	136 s	16 s	6 s	86 s	49 s
	20% gap / 5 feas. sol.	5% gap	49 076	2	366 s	27 s	17 s	96 s	13 s
	10% gap / 10 feas. sol.	5% gap	50 586	2	110 s	81 s	3 s	136 s	536 s
	5% gap / 15 feas.sol.	5% gap	50 586	2	208 s	179 s	4 s	195 s	13 s

6.7 shows the equivalent parameter testing as for the greedy strategy in Table 6.6. For these tests we have chosen to use strategy setting Valid 1 introduced in the previous section.

Table 6.7 portrays no clear conclusion on which parameter setting is optimal for the remaining approaches. As for the greedy approach, there is a trade off between the quality of the solution returned and the time used to return the solution. On average, the fastest

Table 6.7: Parameter testing: First iteration, Valid strategy

Test size	Param - route	Param - inv.	Sol.	Nr. iter.	Total Time	Time- 1st route	Time - 1st inv.	Tot. time - route	Tot. time - inv.
M	None	None	36 413	3	1 126 s	10 s	160 s	592 s	535 s
	First feas.	None	36 283	3	670 s	1 s	25 s	596 s	167 s
	20% gap / 5 feas. sol.	None	35 480	4	771 s	4 s	200 s	326 s	444 s
	10% gap / 10 feas. sol.	None	35 913	8	1 998 s	8 s	200 s	1 387 s	61 s
	5% gap / 15 feas. sol.	None	36 413	3	1 116 s	10 s	160 s	580 s	536 s
	First feas.	5% gap	36 283	10	4 387 s	1 s	10 s	3 352 s	1 035 s
	20% gap / 5 feas. sol.	5% gap	35 480	4	779 s	4 s	204 s	330 s	449 s
	10% gap / 10 feas. sol.	5% gap	35 913	8	1 922 s	8 s	136 s	1 389 s	532 s
	5% gap / 15 feas. sol.	5% gap	36 973	20	12 412 s	9 s	209 s	6 451 s	5 961 s
	L	None	None	51 073	4	1 806 s	180 s	6 s	1 225 s
First feas.		None	46 790	4	1 864 s	16 s	22 s	1 311 s	553 s
20% gap / 5 feas. sol.		None	55 983	2	374 s	36 s	53 s	260 s	114 s
10% gap / 10 feas. sol.		None	51 393	4	1 704 s	81 s	6 s	1 048 s	655 s
5% gap / 15 feas. sol.		None	51 393	4	1 806 s	181 s	6 s	1 152 s	654 s
First feas.		5% gap	55 936	3	1 438 s	16 s	6 s	518 s	920 s
20% gap / 5 feas. sol.		5% gap	55 586	2	592 s	28 s	17 s	276 s	316 s
10% gap / 10 feas. sol.		5% gap	47 500	2	530 s	81 s	4 s	506 s	24 s
5% gap / 15 feas. sol.		5% gap	47 500	2	629 s	181 s	3 s	606 s	23 s

parameter setting is terminating the first iteration routing problem on 20% or 5 feasible solutions. However, the quality of the corresponding solutions are not equally good. The goal of the construction phase is to find the best possible feasible solution at the shortest amount of time. Given the test results in Table 6.7, the parameter setting resulting in the best balance between time and solution quality is to terminate the first iteration rout-

ing problem at the first feasible solution and not terminate the inventory problem. This is consistent with the logical reasoning presented earlier. Thus, this is the parameter setting that is used in all further tests on the strategy settings except for the greedy approach.

Safety inventory

In Section 5.4, the concept of safety inventory in the routing problem was introduced as a strategy. In this thesis we express the safety inventory as a percentage of the original inventory windows. Preliminary testing was executed to find the best level of safety inventory. In all later test instances where safety inventory is included, we use 5% of the original inventory window as safety inventory. During parameter testing, it was evident that a too low level of safety inventory lacks influence on the resulting routes. Equivalently, a too high level of safety inventory influenced the routes too much and the end solution lacked quality.

6.2.3 Construction phase: Results

In Section 6.1, the performance of the exact solution method was analyzed using three different sized test cases. In this section, the performance results of the construction phase of the matheuristic is presented. However, when testing the construction phase, multiple test cases of each size are needed. With a greater number of test instances, it is possible to ensure that the construction phase is independent of test case and not designed to fit a specific test case. Hence, in order to truly test the performance of the matheuristic, five small-sized test cases and four medium- and large-sized test cases are used.

In order to assess the performance of the construction phase, the exact method performance results are needed as benchmark. Table 6.8 shows the performance results of 13 test cases. The three original test cases, corresponding to the test cases used to test the valid inequalities in the previous section are marked as S1, M1 and L1 respectively. The table shows the best solution found within a 5000 second time limit, along with the best bound and corresponding gap. For the test cases that reach optimality within 5000 seconds, the time to optimality is provided instead of optimality gap. All the exact solutions presented in Table 6.8 are the result of the exact solution method with the valid inequality settings recommended in Section 6.1, namely the inclusion of all tested valid inequalities. In Table 6.8, it can be seen that all small test cases manage to find the optimal solution. None of the test cases of medium- and large size are able to find the optimal solution within the 5000 second time limit.

The remaining part of this section is organized as follows. First, the performance of the small-sized test cases are analyzed, followed by an analysis of both the medium- and the large-sized test cases. To conclude, we look at the overall average to identify the best performing strategy settings. The final part of this section is an analysis of the scalability of the construction phase.

Table 6.8: Benchmark from exact solution method results

	Best solution	Best bound	Gap	Time to optimality
S1	29 883	29 883	N/A	516 s
S2	27 170	27 170	N/A	938 s
S3	26 590	26 590	N/A	2 084 s
S4	32 404	32 404	N/A	119 s
S5	33 113	33 113	N/A	512 s
M1	35 462	28 420	19,9%	N/A
M2	25 262	20 396	19,3%	N/A
M3	28 036	20 476	27,0%	N/A
M4	21 682	16 851	22,2%	N/A
L1	65 253	35 039	46,3%	N/A
L2	62 550	34 789	44,4%	N/A
L3	34 190	25 134	26,5%	N/A
L4	36 313	28 897	20,4%	N/A

Small-sized test cases

All small-sized test cases have two ships, four ships and three products. The performance results of the five small-sized test cases are presented in Table 6.9. For each test case, the solution found in the construction phase, along with the time and number of iterations used are presented. To be able to judge the quality of the performance of the construction phase, the exact solution method performance is given in the first results column, denoted E for exact. For all strategy settings, the gap between the solution found by the construction phase of the matheuristic and the best bound identified by the exact solution method in Table 6.8 is reported. Hence a gap of 0% is equivalent to the optimal solution, while a gap lower than the gap presented in the E column is equivalent to an improvement in quality of the solution found. The bottom row of Table 6.9 present the overall average for the small-sized test cases. The overall average results are color marked in order to highlight the difference in quality of the solutions found and the time used, in relation to the exact results. The colors are on a scale from green to red, where green denotes the highest quality and red denotes lowest quality.

Since all of the small-sized test cases manage to reach optimality within 5 000 seconds with the exact solution approach, it is possible to identify whether the construction phase is able to find the optimal solution. For test case *S4*, all strategy settings but the greedy approach manage to find the optimal solution in significantly shorter time than the exact solution method. On average, all strategy settings manage to find a feasible solution in a

Table 6.9: Test results of construction phase test instances for the small test cases

Test case		E	G	V1	V1s	V2	V2s	H	Hs	VA	VAs
S1	Solution	29 883	30 303	29 943	30 020	29 943	30 303	29 943	30 020	29 943	30 020
	Time [s]	516	20,0	20,7	25,6	20,2	32,0	19,6	24,5	20,6	25,7
	Iterations	N/A	2	2	3	2	3	2	3	2	3
	Gap [%]	0,00	1,39	0,21	0,46	0,21	1,39	0,21	0,46	0,21	0,46
S2	Solution	27 170	27 690	29 700	29 423	29 700	29 423	29 700	29 423	29 700	29 423
	Time [s]	938	11,1	17,3	13,4	17,5	13,6	17,5	13,0	17,6	13,7
	Iterations	N/A	2	1	1	1	1	1	1	1	1
	Gap [%]	0,00	1,88	8,52	7,66	8,52	7,69	8,52	7,66	8,52	7,66
S3	Solution	26 590	32 203	29 823	28 380	28 823	28 380	30 426	32 810	29 006	28 380
	Time [s]	2 084	62,5	415	103	781	104	504	274	241	188
	Iterations	N/A	4	7	2	10	2	5	5	10	3
	Gap [%]	0,00	20,5	14,2	9,81	11,2	9,81	15,9	22,0	11,8	9,81
S4	Solution	32 404	33 548	32 404	32 404	32 404	32 404	32 404	32 404	32 404	32 404
	Time [s]	119	24,0	9,0	9,0	8,8	8,7	8,7	8,7	8,9	8,8
	Iterations	N/A	3	2	2	2	2	2	2	2	2
	Gap [%]	0,00	3,41	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
S5	Solution	33 113	35 153	33 113	35 510	35 510	35 510	35 153	33 113	35 510	33 113
	Time [s]	512	28,6	49,6	17,7	47,0	15,5	54,7	12,3	35,5	12,0
	Iterations	N/A	3	6	2	5	2	6	2	4	2
	Gap [%]	0,00	5,80	0,00	6,75	6,75	6,75	5,80	0,00	6,75	0,00
Avg.	Avg. sol.	29 832	31 779	30 997	31 147	31 276	31 206	31 525	31 554	31 317	30 668
	Avg. time [s]	834	29,2	102	33,7	175	34,8	121	66,5	64,7	49,6
	Gap [%]	0,00	6,60	4,58	4,94	5,33	5,13	6,08	6,02	5,45	3,59

significantly shorter time than the exact solution method, however, often at the cost of the quality on the feasible solution found.

The greedy strategy setting manages to find a feasible solution fastest out of the all the strategy settings. This is evident in the average results presented in Table 6.9. This, along with the fact that it performs lowest on quality of the feasible solution found, is in line with the expectations for the greedy strategy. Equivalently, the valid strategy setting is able to find high quality solutions at the expense of slower convergence to the feasible solution. The hybrid strategy settings, are in the middle ground between the greedy strategy setting and the valid strategy settings. Overall, the performance of the construction phase on the small-sized test cases corresponds with the expectations. On average, none of the

construction phase solutions manage to outperform the exact solution on solution quality. However, the run time efficiency performance is superior to the average exact solution time to optimality.

The strategy settings that include safety inventory use a shorter time to find a feasible solution compared to the corresponding strategy settings without safety inventory. The time is reduced with the inclusion of safety inventory because the number of iterations is reduced. When looking at the results of the majority of the small-sized test instances with safety inventory, significantly fewer iterations are needed. However, as the inclusion of the safety inventory cannot be called a valid addition, the quality of the solutions found with safety inventory is affected. In general, the quality of the feasible solutions found suffers and is lower than the corresponding strategy setting without safety inventory. The inclusion of safety inventory is a positive contribution with respect to the solution time of the small-sized test instances. However, it is important to note that the safety inventory increases the complexity of the routing problem, and this increase in complexity has likely a greater impact on larger test cases.

The only difference between strategy setting $V1$ and $V2$ is the inclusion of Strategy 2 in $V1$. Strategy 2 imposes additional operation periods in the violation port. When looking at the test results of the small-sized test cases, the majority of the test instances show almost no difference in performance between strategy setting $V1$ and $V2$. When designing this strategy, the focus was to retain the validity of the routing problem and not knowingly remove the optimal solution. As a result, the majority of the small test cases is not affected by the addition of the valid inequalities of Strategy 2.

When analyzing the functionality and effect of the proposed valid inequalities of Strategy 2, it became evident that the way the valid inequalities are designed, they will not necessarily force a change on the routes of the routing problem. To be able to understand why, it is important to consider the interaction between the routing problem and the inventory problem. The only component of the routing problem solution that is saved between the routing problem and the inventory problem is the set of ship routes, and the operation variable solution of the routing problem is not utilized by the inventory problem. The purpose of the operation variable in the routing problem is to affect the sequence of port visits through valid inequalities on the operation variable. Hence, there is not necessarily a connection between the number of operation periods in the routing problem and the number of operation periods in the inventory problem. Since the routing problem is not directly constrained by the inventory limits like the inventory problem, the distribution of operation periods across the ports and ships in the routing problem may differ notably from the distribution in the inventory problem. The impact of this difference in distribution of operation periods is most easily explained through an example. Lets say that in the first iteration of the routing problem, the fleet operate for seven time periods in *Port 1*. However, when returning the route solution to the inventory problem, the inventory only operate four time periods in *Port 1* and provoke a three period violation in *Port 1*. With Strategy 2, a new constraint is added to the next iteration of the routing problem using the

number of operation periods in the inventory problem as reference. The lower bound on operation periods now become four time periods plus the periods needed to account for the violation. In this example two time periods are needed. Thus, the valid inequality added to the routing problem for *Port 1* is a requirement of at least six time periods of operation, a lower requirement than what was used in the first iteration of the routing problem. The lower requirement is a result of a wrongful distribution of the operation periods in the routing problem in the previous iteration. Hence, no changes to the routes are imposed solely based on Strategy 2 in these situations. However, it is important to note that even though the strategy does not necessarily enforce a change in the route in this iteration, it serves as a lower bound for all later iterations and contributes to limit the feasible region of the problem. Test instance *V1_S3* is a good counter example, where Strategy 2 has contributed to a significant reduction in time before a feasible solution was found.

In order to be sure of a change in the number of operation periods in each iteration of the routing problem, a more greedy approach is needed. An alternative way to formulate this strategy is to always use the number of operation periods from last iteration of the routing problem as the basis of the lower bound. However, since it not necessarily the number of operation periods, but the distribution across both ports and ships that is the challenge, there is no guarantee that this approach is well functioning.

Medium-sized test cases

The medium-sized test cases are designed with three ships, six ports and four products. We now present and discuss the results of the medium-sized test cases with the strategy settings. Table 6.10 presents the performance results of the four medium-sized test cases. As for the small-sized test case results, all test instance results are presented with the feasible solution found by the construction phase, along with the running time, number of iterations and gap between the solution found and best bound from Table 6.8.

In Table 6.10, '-' is used to denote that no feasible solution was found given the parameter setting chosen. Not all of the relevant test instances managed to find a feasible solution in the iteration reported within the time limit of 500 seconds for each iteration. Hence, the search was terminated. This impacts the calculation of the average solution and average time since there is no solution or time to take into consideration. However, it should count negatively for a strategy setting that it did not manage to find a feasible solution. As a result, we have chosen to calculate the solution and the time for these test instances as the worst solution and time found by any other strategy setting of that test case with an addition of 20%. The penalty is added to account for the fact that it is worse for a strategy setting to not find a feasible solution than to find a solution of low quality.

When looking at the average performance results of the medium-sized test instances in Table 6.10, there are some strategy settings that stand out. As was evident in the results of the small-sized test instances, the greedy strategy has the fastest rate of convergence to

Table 6.10: Test results of construction phase test instances for the medium test cases

Test case		E	G	V1	V1s	V2	V2s	H	Hs	VA	VAs
M1	Solution	35 462	38 210	36 683	36 683	35 753	36 747	37 323	38 093	34 813	36 683
	Time [s]	5 000	166	2 253	737	348	339	1 579	849	399	705
	Iterations	N/A	3	5	3	2	2	5	4	2	3
	Gap [%]	19,9	25,6	22,5	22,5	20,5	22,7	23,9	25,4	18,4	22,5
M2	Solution	25 262	25 513	23 730	33 444	-	33 444	27 111	33 444	24 571	33 444
	Time [s]	5 000	107	5 699	37,0	-	36,9	1 206	37,2	3 329	39,4
	Iterations	N/A	2	13	1	17	1	7	1	9	1
	Gap [%]	19,3	20,1	14,0	39,0	-	39,0	24,8	39,0	17,0	39,0
M3	Solution	28 036	32 136	27 960	-	27 960	-	28 060	34 837	27 960	27 960
	Time [s]	5 000	369	1 216	-	848	-	432	1 659	1 041	1 009
	Iterations	N/A	7	9	20	6	20	4	7	9	6
	Gap [%]	27,0	36,3	26,8	-	26,8	-	27,0	41,2	26,8	26,8
M4	Solution	21 682	21 283	20 172	-	24 966	-	21 805	23 344	23 156	21 897
	Time [s]	5 000	228	3 321	-	5 460	-	3 044	3 805	8 464	3 727
	Iterations	N/A	4	10	13	13	9	8	7	13	7
	Gap [%]	22,2	22,2	16,5	-	32,5	-	22,7	27,8	22,3	23,0
Avg.	Avg. sol.	27 610	29 286	27 136	35 473	32 203	35 489	28 575	32 430	27 625	29 996
	Avg. time [s]	5000	218	3 122	3 120	3 374	3 131	1 565	1 588	3 308	1 370
	Gap [%]	22,1	25,7	20,0	39,1	32,2	39,1	24,6	33,4	22,3	27,8

a feasible solution. Compared to all other strategy settings, the time to a feasible solution is significantly lower. The difference in time to a feasible solution between the different strategies is even greater for the medium-sized test cases than for the small ones. This is likely due to the greater complexity of the instances and corresponding longer average time per iteration for the none-greedy approaches. However, all strategies are able to find a feasible solution within the 5 000 second time limit. In terms of average quality on the solutions that are found, only the Valid 1 approach manage to compete with the exact solution method.

The average performance of strategy settings V1s and V2s along with V2 on the medium-sized test case is dramatically affected by the penalty from the lack of ability to find a feasible solution with the given parameter settings. The inclusion of safety inventory increases the complexity greatly for test instance *V1s_M3* compared to *V1_M3*. The same applies to test instances *V2s_M3*, *V1s_M4* and *V2s_M4*. As already discussed, imposing safety inventory implies a potential for an increase in activity for the ships in the routing problem. With stricter requirements to the number of port visits, compartments

needed and operations periods in each port, the complexity increases. As a result, these strategy settings fail to find a feasible routing solution in the routing problem in the given iteration.

Large-sized test cases

The large-sized test cases are composed of four ships, eight ports and four products. The performance of the construction phase of the matheuristic with the large test cases are presented in Table 6.11. All test instances are presented with solution found, time to a feasible solution, the number of iterations utilized and gap. '-' is used to denote the cases where no feasible solution was found given the parameter settings, and the implications are identical to that of the medium-sized test cases.

Table 6.11: Test results of construction phase instances for the large test cases

Test case		E	G	V1	V1s	V2	V2s	H	Hs	VA	VAs
L1	Solution	61 310	50 587	46 790	49 030	53 323	56 600	48 726	47 843	47 160	57 013
	Time [s]	5 000	109	1 844	1 180	1 887	2 446	1 645	926	1 679	3 420
	Iterations	N/A	2	4	3	4	5	5	3	6	6
	Gap [%]	43,0	28,1	41,1	33,2	19,8	11,0	34,2	37,3	39,7	10,0
L2	Solution	62 550	47 010	42 786	45 416	45 400	46 427	42 787	47 200	47 063	55 756
	Time [s]	5 000	480	1 895	4 441	2 604	4 234	2 252	2 904	2 337	4 092
	Iterations	N/A	2	3	6	4	6	4	5	4	7
	Gap [%]	44,4	18,5	41,8	26,8	26,9	21,5	41,8	17,6	18,3	-18,5
L3	Solution	34 190	32 146	34 888	-	33 782	35 528	36 656	36 142	35 050	-
	Time [s]	5 000	1 110	2 064	-	8 926	2 555	3 912	2 976	4 355	-
	Iterations	N/A	3	4	9	13	5	7	4	7	9
	Gap [%]	26,5	29,3	9,50	-	17,1	5,32	-1,70	1,44	8,42	-
L4	Solution	36 313	36 653	35 067	34 257	33 953	34 290	36 440	38 550	33 413	34 257
	Time [s]	5 000	537	1 599	5 874	8 228	386	1 164	1 553	1 222	1 002
	Iterations	N/A	5	3	11	14	2	6	7	3	3
	Gap [%]	20,4	11,6	25,6	33,2	36,2	32,9	13,5	-3,50	41,6	33,2
Avg.	Avg. sol.	49 577	41 599	39 883	43 173	41 645	43 211	41 152	42 434	40 672	47 753
	Avg. time [s]	5 000	559	1 851	5 552	5 411	2 405	2 243	2 090	2 393	4 806
	Gap [%]	34,4	24,9	22,3	27,6	24,5	27,0	24,7	27,1	23,4	33,7

The construction phase is not able to compete on quality with the exact solution method for the small-sized test cases and not notably on the medium-sized test cases either. How-

ever, there is an increase in performance on quality from the small-sized test cases to the medium-sized test cases. In Table 6.11, it is evident that the construction phase is superior to the exact solution method on both time and quality in the large sized test cases. It is apparent that the value of the construction phase increases with the complexity and size of the test cases. The more complex the test cases are, the harder it is for the exact solution method to efficiently find good solutions.

On average, all strategy settings return a lower optimality gap than the exact method. The Valid 1 approach, which is one of the settings striving to make valid choices and return high quality solutions, imposes a 35% reduction of the exact gap. Unlike in the smaller test cases, setting V1 does not sacrifice the efficiency of the search in the same degree to produce high quality solutions. As expected and in line with earlier results, the greedy strategy is highly efficient. However, on average, the greedy strategy also manage to find a fairly good feasible solution. The feasible solutions found by the construction phase are subject to a subsequent improvement phase. With the possibility of improving the quality of the solution, the superior efficiency of the greedy strategy setting should be noted.

Strategy setting V1 outperforms V2 both in the medium- and large-sized test instances. This implies that the contribution from Strategy 2 is both significant and important. Even though Strategy 2 might not always impose restrictions directly forcing a change on the routes single-handedly, the valid inequalities function as lower bounds essential for the efficiency in all later iterations.

Two of the strategy settings with the best overall average performance on the large-sized test cases are V1 and VA. The only difference in the design of these two strategy settings is the use of Strategy 3 in V1 and Strategy 4 in VA. Since the purpose of Strategy 3 and 4 is identical, V1 and VA should to a great extent have similar performance on the same test cases. The only difference between Strategy 3 and 4 is the method of formulating the commodity flow. The commodity flow in Strategy 3 is formulated with a super source to solve the issue of possibly multiple sources for the commodity. This is considered common practice and is the method used to solve problems with multiple sources. Strategy 4 is a new way of solving the occurrence of multiple sources, however, it can not be considered equally elegant as the use of a super source. Still, it is interesting to see how the difference in formulation impacts the solution efficiency and quality.

The first iteration routing and inventory solution is identical for V1 and VA. We use test case L1 as an illustrating example of the impact of the difference in formulation of the commodity flow in Strategy 3 and Strategy 4. The first iteration routing problem for instances *V1_L1* as well as for *VA_L1* is built up of 10 110 variables and 2 739 constraints after the presolve procedure in Xpress. The most important difference between the formulation in Strategy 3 and Strategy 4 is the number of variables and constraints added in each iteration. The second iteration routing problem of test instance *V1_L1* has 24 200 variables and 14 207 constraint, hence an 139 % increase in the number of variables from the first to the second iteration. On the other hand, the second iteration routing problem of test

instance VA_L1 has 26 917 variables and 18 134 constraints, which is equivalent to a 166 % increase in the number of variables. This increase in size of the routing problem results in higher complexity of the routing problems of strategy setting VA. As a result, the routing problems of the VA instances have a harder time reaching optimality within the 500 second time limit and the 200 second time limit since the last discovered feasible solution. In the specific case of test case VA_L1 , the routing problem is terminated often due to the fact that the time since last feasible solution found in the search exceeds 200 seconds. Consequently, the time in each iteration is reduced at the cost of more iterations compared to $V1_L1$. Instance $V1_L1$ manage to a great extent to reach the optimal solution within the given time limits and thus use noteworthy more time in each iteration. However, since the routing problems in VA_L1 are terminated early, the routes returned to the inventory problem are not as good as those returned from a routing problem of $V1_L1$ with closed gap. As a result, the feasible solution returned by VA_L1 is of lower quality than the solution returned by $V1_L1$. The performance of the strategy setting V1 and VA is on average positively correlated, however, the average performance of V1 is superior to VA due to the lower complexity of the routing problem as a result of a more elegant formulation.

Construction phase performance

The performance of the construction phase has until now been presented and commented independently for the three test case sizes. However, in order to conclude on which strategy settings that facilitate for the best construction phase performance, it is essential to take the overall results into account. Hence, we now look at the overall average performance results of the construction phase for each strategy setting. Table 6.12 presents the average solution found by the construction phase, along with the average time to a feasible solution. The third row of Table 6.12 reports the average gap given the best bounds discovered by the exact solution approach in Table 6.8.

Table 6.12: Overall average construction phase results for the different strategy settings

	E	G	V1	V1s	V2	V2s	H	Hs	VA	VAs
Avg. sol.	35 673	34 034	32 543	36 178	4 742	36 218	33 580	35 171	33 058	35 718
Avg. time [s]	3 611	250	1 569	2 715	2 770	1 717	1 218	1 157	1 779	1 920
Avg. gap [%]	18,8	14,8	12,5	18,2	17,2	18,1	15,2	17,5	13,8	18,0

At this point, it is possible to draw some conclusions on which strategy settings that are well-functioning and which should be discarded. There are especially three strategy settings that stand out as interesting for continued testing and as input to the improvement phase of the matheuristic. First, the greedy strategy setting has consistently been proved as the most efficient setting. On average, it has a significantly shorter time to a feasible solution compared to the other strategy settings and the exact solution method. As can be

seen in Table 6.12, the greedy strategy setting manages on average to be efficient while also returning fairly good solutions. Further, it has become evident through all test results that the Valid 1 approach manages to return the highest quality feasible solutions. The superior performance on returning high quality solutions comes at the expense of the efficiency of the construction phase using V1 as strategy setting. The last strategy setting that stands out is the hybrid strategy setting. The hybrid approach is neither best on efficiency nor quality, however, it shows overall good performance in both dimensions. Due to these conclusions, we continue with strategy settings G, V1, and H when testing the improvement phase of the heuristic. Before analyzing the performance of these three strategy settings further, some last attention is dedicated to the remaining strategy settings not chosen for further analysis.

The use of safety inventory has had a tendency to positively affect the efficiency of the construction phase on the smaller test instances. However, this is not equally clear in the overall average of the safety inventory strategy as it is highly affected by the fact that multiple instances in both the medium- and large-sized test cases failed to find a feasible solution. The positive contribution on the efficiency does not outweigh the reduction in quality on the solutions returned. Hence, the testing of the strategy settings with safety inventory is not further extended.

The contribution of Strategy 2 in strategy setting V1 is significant and essential for both the efficiency and the quality of the solution. As this is the major difference between strategy settings V1 and V2, it is more interesting to precede testing with strategy setting V1 than V2.

The last remaining unaddressed strategy setting is VA. As already explained, the purpose of VA is identical to V1, and there is only a technical difference in the formulation of the components of VA and V1. Since V1 has consistently, on average, marginally outperformed VA, we have chosen to continue testing with V1 rather than VA.

The three promising strategy settings: G, V1, and H

We can now evaluate the most promising strategy settings. From now on, G is referred to as the greedy strategy, V1 as the valid strategy (V), and H as the hybrid strategy. Table 6.13 shows the development of the number of variables and constraints in the routing problem for each strategy setting for example test case L1. There is a significant difference in both the number of variables and constraints between the three strategies. The number of variables in the greedy strategy setting remains unchanged in each iteration. In the greedy strategy, only one constraint is added for each violation in the previous inventory problem. Hence, there is an increase in number of constraints in each iteration. Compared to the complexity of the hybrid and valid approach, the complexity of the greedy strategy is significantly lower which enables the efficiency of the greedy approach. The valid approach has a higher variable count than the hybrid approach due to the inclusion of the hop formulation. The number of constraints is highly dependent on the number of violations in the inventory problem. Hence, the numbers in Table 6.13 imply that the hybrid approach

experienced a higher number of violations than the valid approach, since the difference in the number of constraints is supposed to be fairly equal.

Table 6.13: Development of the size of the routing problem

Iter.	Variables			Constraints		
	G	V	H	G	V	H
1	12 076	10 110	10 100	4 424	2 739	2 739
2	12 076	24 200	23 855	4 432	14 207	27 817
3		34 700	28 885		25 103	39 473
4		43 246	31 201		34 406	35 793
5			37 805			43 040

The search for a feasible solution is an iterative search, where the routing problem supplies information to the inventory problem which return additional information to the next iteration routing problem. Figure 6.2 shows the development of the objective value in the routing problem and in the inventory problem for the three chosen strategy settings for example test case L1. As can be seen, the objective value of the two problems converge towards each other. The objective value of the routing problem is strictly increasing after the first iteration. In the first iteration, the routing problem is terminated early resulting in an artificially high objective value. However, in all later iterations, new restrictions are imposed and it becomes impossible to reduce the routing objective value. Due to the unpredictable nature of the violation penalty, the inventory problem is not strictly decreasing. In earlier iterations when the quality of the route solution is low, high penalty costs are incurred and the objective value of the inventory problem is artificially high. As the quality of the route solution improves, the inventory objective value decreases. However, the objective value might increase in later iterations in situations where the cost of resolving the violation is higher than the actual penalty cost of the violation.

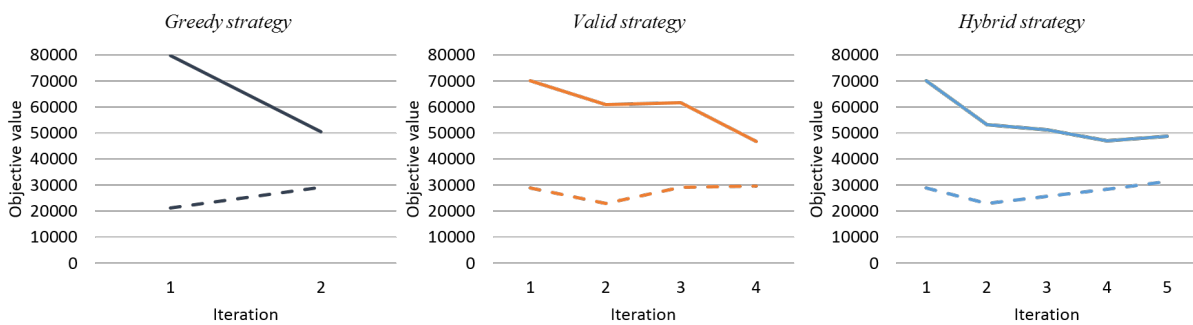


Figure 6.2: Development of objective value in the routing and inventory problem in each iteration. Dotted/solid line represents the routing/inventory problem objective value.

6.2.4 Construction phase: Scalability study

To be able to investigate the scalability of the construction phase, it is necessary to test the construction phase on time horizons of different lengths. Originally, we have used 14 days as standard planning horizon with 3 time periods each day. Hence, the used time horizon is equivalent to 42 time periods. When testing the scalability of the construction phase, both 18 days and 21 days are tested as length of the planning horizon. This is equivalent to a length of 54 time periods and 63 time periods, respectively. To extend the time horizon, it was necessary to add port production and consumption levels for the added time periods. The scalability results of the greedy strategy is presented in Table 6.14. Scalability is measured in terms of both time to find a feasible solution and the total number of nodes processed by both the routing problem and the inventory problem in all iterations. Overall, a significant increase in both time and the number of nodes processed can be seen for the longer time horizons. The notation '-' means that the construction phase failed to find a feasible solution given the parameter settings used.

Table 6.14: Scalability results of the greedy strategy

	G					
	14d		18d		21d	
	Time [s]	Nodes	Time [s]	Nodes	Time [s]	Nodes
S1	15,8	21 710	14,4	5 838	106	45 459
S3	17,2	10 735	531	315 436	1 464	454 5143
M1	182	86 795	388	202 687	-	-
M3	369	148 015	1 970	320 847	1 390	131 964
L1	583	266 942	8 373	571 094	-	-
L4	537	150 430	2 169	161 131	7 424	368 281

The reason for only presenting the scalability results of the greedy strategy is that the tests with longer time horizons on both the hybrid and the valid approaches failed to return a feasible solution within the time limits given in the parameter settings. Both of these strategies impose a large increase in complexity to the problem compared to the greedy strategy with the original time horizon. Hence, neither the hybrid nor the valid strategy can be considered scalable strategies.

6.2.5 Improvement phase: Parameter setting

In this section, we present the parameter settings for the improvement phase of the matheuristic. The purpose of the improvement phase is simply to improve the feasible solution from the construction phase. In addition to the operators designed and presented in Section 5.5, several parameters are needed to facilitate this improvement. First, we

present the necessary time limit constraints applicable for all improvement operators. Second, we present the parameter settings relevant for the three operators independently.

Time limits

As for the construction phase, time constraints and limits are necessary to secure an efficient improvement phase. First, the time limit of each iteration must be decided. For the Inter operator, preliminary testing showed that very few iterations needed more than 100 seconds to find an improvement of the currently best solution. Equivalently, when employing the Intra operator, the majority of the iterations of the search managed to reach a sufficiently small gap within the 100 second time limit. Hence, the time limit for each iteration of the search is 100 seconds.

In Section 5.5, the improvement phase of the matheuristic was presented as a first improvement scheme, where the search is terminated in relation to the timing of the first improvement made to the currently best solution. The concept of a time buffer was introduced to not limit the search from reaching solutions in close proximity. The size of this time buffer is determined to be 20 seconds on the basis of preliminary testing. It is important to keep this time buffer relatively small in size to ensure the efficiency of the improvement search.

The last technical efficiency measure added to the improvement phase is the termination of the search at 1% optimality gap. Naturally at 1% optimality gap, we can ensure that the current solution is only 1% away from optimum, and using time on proving optimality is not adding any value to the improvement search.

Parameter setting: Intra operator

The Intra operator iteratively opens up for improvement of the route of each ship. With this in mind, the only relevant parameter is the termination criteria. With termination criteria, we mean at which criteria should the improvement heuristic stop the iterative search for better solutions. As presented in Section 5.5, the minimum number of iterations for the Intra operator is equivalent to the number of ships in the fleet, \bar{V} . There are multiple options of defining the maximum number of iterations, for example \bar{V} iterations, $2x\bar{V}$ iterations, or run until no further improvements have been found in the last \bar{V} iterations. During preliminary testing, it was observed that the latter was the best option when considering both quality of the final solution and the efficiency. Hence, for all testing of the Intra operator, the termination criteria is the following: Stop if there has not been any new improvements in the last \bar{V} iterations.

Parameter setting: Inter operator

For the Inter improvement operator, there are four main parameters to be determined in addition to the time constraints defined above. The relevant parameters are: termina-

tion criteria, length of time interval, fraction overlap between time intervals and time slack.

The lower bound on the number of iterations of the improvement phase when using the Inter operator is equivalent to the total number of intervals, \bar{I} . Two termination criteria are chosen to be tested for the Inter operator. First, the improvement heuristic is run exactly \bar{I} iterations before it is terminated. With this criteria, the improvement heuristic is able to run through all the periods of the planning horizon exactly once. The second criteria is equivalent to that used by the Intra operator, namely to run the improvement heuristic until no new improvements have been found in the last \bar{I} iterations. Hence, the heuristic has the opportunity to make further improvements after the first round of improvements. The first criteria is denoted \bar{I} and the second criteria is denoted \bar{I}^+ .

Since the number of iterations needed to cover all time periods of the planning horizon is a function of the length of each time interval and the degree of overlap, these are important parameters for the Inter operator. There are many considerations that need to be taken when determining these two parameters. First, the degree of overlap must be decided. A large degree of overlap results in a large number of intervals covering close to all time period combinations in the route. With such an overlap, very few improvements are removed from the list of possible improvements. However, the efficiency cost of a large degree of overlap is high. With a small degree of overlap, the consequence is fewer intervals and thus higher efficiency as well a reduction in possible improvements. Whether it is beneficial with a large degree of overlap or a small degree of overlap is highly dependent on the structure of each specific input route. As a result of the lack of consistency in the preliminary results, we have fixed the degree of overlap to half the length of the time interval. It can be noted that a measure to minimize the repercussions of this specific parameter setting is to arbitrarily change the starting points of the time intervals when starting the iterations again at the beginning of the planning horizon. Hence, for each time you iterate over the planning horizon, new combinations of time intervals are used, resulting in more possibilities for improvements of the routes. However, preliminary testing showed that only marginal benefits could be gained from this. This supports the decision to keep this parameter constant without further testing.

For the remaining two parameters, the interval length and time slack, further testing is needed. Table 6.15 presents the test results for the three dimensions we want to test, namely termination criteria, length of time interval and the number of time periods of time slack. The lengths of time intervals tested are 8, 10, or 12 time periods. The number of time periods of time slack tested is 0,1,2 and 3. A maximum of three time periods of slack is tested as preliminary tests showed that a greater number of time periods of slack resulted in too much flexibility outside the free time interval. No significantly better solutions were found to make up for the increase in running time. For all test results, the objective value, running time and the number of improvements are reported. With number of improvements we refer to how many iterations there has been an improvement to the best current solution.

Table 6.15: Parameter testing: Inter operator

Test case	Interval length	Time slack	Objective \bar{I} iter.	Time \bar{I} iter.	Nr. of impr.	Objective \bar{I}^+ iter.	Time \bar{I}^+ iter.	Nr. of impr.
M	8	0	40 173	8 s	2	39 890	15 s	4
	8	1	39 730	17 s	5	38 650	40 s	7
	8	2	37 020	29 s	4	37 020	49 s	4
	8	3	36 027	33 s	4	36 027	54 s	4
	10	0	40 783	10 s	2	39 890	34 s	4
	10	1	39 930	21 s	4	36 033	48 s	10
	10	2	35 530	36 s	7	35 463	86 s	8
	10	3	36 450	56 s	5	35 463	171 s	7
	12	0	40 313	17 s	3	39 460	44 s	7
	12	1	35 530	31 s	5	35 463	66 s	7
	12	2	35 530	57 s	5	34 633	152 s	9
	12	3	35 530	98 s	4	34 633	171 s	11
L	8	0	50 000	13 s	1	50 000	17 s	1
	8	1	47 773	35 s	5	47 450	76 s	8
	8	2	48 333	66 s	2	48 333	114 s	2
	8	3	47 143	115 s	3	47 143	186 s	3
	10	0	50 340	27 s	0	50 340	27 s	0
	10	1	47 993	56 s	4	47 417	194 s	7
	10	2	47 993	109 s	2	47 993	254 s	2
	10	3	48 289	189 s	3	48 289	329 s	3
	12	0	50 000	60 s	1	50 000	172 s	1
	12	1	47 120	128 s	5	47 120	252 s	5
	12	2	50 000	161 s	1	50 000	339 s	1
	12	3	48 993	209 s	2	48 037	470 s	3

As can be seen in Table 6.15, for each length of time interval, the running time increases as the time slack increases. For the medium-sized test case, the objective value also improves with the increase in number of time periods of time slack. The running time increase with both the length of the time interval and the number of time periods slack. This results in a larger neighborhood and explains both the improvement in objective value and the increase in running time. For the large-sized test instance, the objective value does not consistently improve with the length of time interval and number of time periods slack. This can be explained by the added size and flexibility. The result can be the need for

more time to discover the same solutions and a termination of each iteration caused by the time limit rather than the discovery of an improved solution.

When deciding the parameter setting, the trade-off between adding enough flexibility to find good solutions and not increasing the problem to an unmanageable size must be considered. Without test results from a larger pool of test instances, it is hard to determine the best suiting parameter setting for this improvement operator. However, we must conclude on the parameter setting with the currently available information. Based on the results of Table 6.15 and the desire to achieve a flexible problem without increasing the complexity significantly, we conclude that the use of 10 as time interval length and 2 as the number of time periods slack on variables outside the free time interval are suitable parameters. This parameter setting seems able to achieve good solutions without using unreasonably long time, independent of termination criteria. Consequently, the number of time periods each time interval shifts is 5.

The final parameter that needs to be decided on is the termination criteria for the Inter operator. Given the parameters already decided, the choice between the two termination criteria comes down to the weighting of quality of solution contra the weighting of the run time. For the medium-sized test case, the use of \bar{T}^+ returns a better solution than \bar{T} at a cost of longer run time. For the large-sized test case, the solutions returned are equal, but \bar{T}^+ use longer time. The use of \bar{T}^+ implies an increase in run time, however it also increases the potential for finding better solutions. Since the difference in run time between the two termination criteria is insignificant in relation to the time needed by the exact solution method, we want to use \bar{T}^+ as termination criteria to facilitate a search for high quality solutions.

Parameter setting: Combination operator

All the parameter settings applicable for the Intra and Inter operator separately are also applicable for the Combination operator. The Combination operator always utilizes the termination criteria of the improvement operator that is currently being executed. Hence, no additional parameters need to be defined.

6.2.6 Improvement phase: Results

In this subsection, we present the results from testing the improvement heuristic on solutions from the construction phase of the matheuristic. In Section 6.2.3, we concluded that strategy settings G, V1 and H were the best performing settings for the construction phase. Consequently, we only test the improvement phase on solutions returned by the construction phase with either of these three settings. Due to time and computational resource constraints, only two test cases from each test case size from the construction phase is chosen as input for the improvement phase. The input test cases were chosen at random, and S3, S5, M1, M3, L1, and L4 are the test cases we use to test the improvement phase. In the remaining part of this section, the results from running the improvement heuristic with the three operators defined are presented and discussed. The Intra, Inter and

Combination operators are all tested with the three construction phase strategy settings.

The results from running the improvement heuristic on the specified test instances is reported in Table 6.16. First, the improvement solution, i.e. the final solution achieved by the matheuristic, is reported. To be able to assess the performance of the improvement phase, the percentage improvement from the construction phase solution to the improvement phase solution is included. The construction phase solutions used to calculate the percentage improvement can be found in Section 6.2.3. The improvement time, which refers to the running time of the improvement phase is reported along with the number of improvements. In the case where the optimal solution was found in the construction phase, the improvement phase is not needed and results are thus reported as not applicable.

Overall, the improvement heuristic is able to improve the majority of the test instances. At least one of the operators are able to find a solution that improve the construction phase solution. In fact, it is only two test instances, *V1_M1* and *V1_M3*, out of the 18 test instances where the improvement phase fails to find an improvement from the construction phase solution. In general, all the three improvement operators are efficient. This can be seen by the average performance in Table 6.16. On average, all operators manage to find an improvement for all the three strategy settings. However, Table 6.16 indicate that it is the solutions from the construction phase run with the greedy approach that is most responsive. This is along the lines of the expectation, and we continue this discussion later. First, to get a deeper understanding of the functionality of the improvement phase, we discuss the results of the different improvement operators in greater detail.

The Intra and Inter operators

As described in Section 5.5, the Intra operator improves the route of one ship at a time, while the Inter operator can improve the solution by making changes across ship routes. When analyzing the results in Table 6.16, it is clear that there is no consistency in which of the Intra and Inter operator that is superior in performance. This can easily be explained by the fact that both of Intra and Inter is highly dependent on the structure of the input solution received from the construction phase. Hence, given one solution structure, Intra is superior to Inter and vice versa for other structures. The solutions returned from the construction phase are sometimes constructed in a way that enables improvements within one ship route, while for other solutions the possibility of doing cross ship changes is essential for improving the solution.

The Intra operator is able to find solutions that are not reachable through the use of the Inter operator. This happens when the changes in a ship schedule with the Intra operator occur beyond the limits of all the time intervals used in Inter. An example of this occur in test instance *V1_S3*. In the input solution from the construction phase, *Ship 1* travels the following route (departure time): [5(1) - 4(7) - 1(14) - 3(23) - 4(27) - 0]. After employing Intra, the solution is improved and *Ship 1* has the following route: [5 - 4 - 3 - 1 - 0]. *Port 4* is no longer visited twice, and *Port 3* is visited before *Port 1*. In the first solution, *Port 1* was originally visited in time period 14, while *Port 3* was visited in time period 23. When

Table 6.16: Results from testing the Intra, Inter and Combination improvement operators

Test case	Info	G			V1			H		
		Intra	Inter	Comb.	Intra	Inter	Comb.	Intra	Inter	Comb.
S3	Imp. sol.	27 813	29 793	26 763	26 763	29 793	26 763	27 713	30 377	26 763
	% Imp.	13,6	7,48	17,2	10,3	0,10	10,3	8,91	0,15	12,0
	Imp. time [s]	147	36	71	24	31	35	188	17	145
	Nr. of impr.	2	4	4	2	0	2	3	1	4
S5	Imp. sol.	33 380	34 113	33 113	N/A	N/A	N/A	33 280	34 113	33 113
	% Imp.	5,04	2,96	5,80	N/A	N/A	N/A	5,33	2,96	5,80
	Imp. time [s]	71	26	83	N/A	N/A	N/A	69	25	83
	Nr. of impr.	3	2	3	N/A	N/A	N/A	3	2	3
M1	Imp. sol.	36 900	35 217	35 217	36 683	36 683	36 683	37 323	36 683	36 283
	% Imp.	3,43	7,83	7,83	0,00	0,00	0,00	0,00	1,71	2,79
	Imp. time [s]	124	51	200	114	31	139	128	65	498
	Nr. of impr.	2	3	4	0	0	0	0	2	4
M3	Imp. sol.	26 11	27 693	26 110	27 960	27 960	27 960	27 960	27 960	27 960
	% Imp.	18,8	13,8	18,8	0,00	0,00	0,00	0,40	0,40	0,40
	Imp. time [s]	33	88,3	50	103	41,5	262	208	53	344
	Nr. of impr.	2	6	2	0	0	0	1	1	1
L1	Imp. sol.	47 850	48 743	46 637	46 540	46 637	46 540	46 637	48 643	46 637
	% Imp.	5,41	3,65	7,81	0,53	0,33	0,53	4,29	0,17	4,29
	Imp. time [s]	300	404	691	210	149	415	298	40	458
	Nr. of impr.	3	4	6	1	1	1	2	2	2
L4	Imp. sol.	34 373	33 953	33 313	34 173	33 313	33 313	33 480	33 313	33 313
	% Imp.	6,22	7,37	9,11	2,55	5,00	5,00	8,12	8,58	8,58
	Imp. time [s]	482	126	255	157	154	286	140	102	178
	Nr. of impr.	2	6	5	1	4	2	1	5	2
Average	Avg. % Imp.	8,75	7,19	11,1	2,22	0,90	2,63	4,50	2,32	5,64
	Avg. time [s]	193	122	225	101	68	189	172	50	284

employing the Inter operator with the defined parameter settings, no time interval is created where both time period 14 and 23 are included. As a result, the Inter operator is not able to reach the same solution as Intra in this case. This is clearly a drawback of the Inter operator. If the improvement heuristic is to be developed further, it can be beneficial to consider an alternative way of creating time intervals to cover the planning horizon more exhaustively.

Naturally, the Inter operator can also find solutions not reachable by the Intra operator. Test instance G_L4 is an example of this. When using the Inter operator to improve the construction phase solution, an additional port visit is added to the route of *Ship 2*. This type of change in the ship routes is not possible with the Intra operator, as the operator can only change the sequence of port visits, and/or remove a port visit from the sequence.

In the majority of the test instances, the Intra and Inter operator causes the improvement phase to return different solutions. However, there is no clear trend on which operator returns the highest quality solution. The major difference between the performance of the two operators are found in the running time. As can be seen in Table 6.18, the Inter operator is superior to the Intra operator when it comes to the average run time. Since the difference in run time is low on the exact solution time scale, it is not enough to discard the Intra operator. The next interesting step is to see how these two operators function together. With the Combination operator, the improvement heuristic might be able to reach solutions that are not reachable by either Intra or Inter independently.

The combination operator

As can be seen in Table 6.16, the Combination operator is consistently able to find at least an equally good solution as was found by either the Intra or Inter operator. It is worth noting that the run time cost of the Combination operator is also consistently higher than the Intra and Inter operator. To run both Intra and Inter operators sequentially in the same improvement scheme requires an increase in the number of iteration. The result is an increase in running time. However, in approximately half of the instances tested, the Combination operator is able to find a solution superior to the solutions found both by Intra and Inter. This improvement is made possible because the Combination operator alternate between the Intra and the Inter operator. Whenever one of the operators find an improvement to the currently best solution, a new neighborhood that was unreachable for the other operator is made available. The drawback of the additional running time when employing the Combination operator must be seen in relation to the time used in the Construction phase. Initially, the increase in run time as a result of the Combination operator instead of the Intra or Inter operator is not substantial.

Even though it was not apparent in our test results, there is no guarantee that the Combination operator is able to find an equally good solution as was found by either the Intra or the Inter operator. During the execution of the improvement heuristic with the Combination operator, each ship is iterated through, employing the Intra operator and keeping one ship free in each iteration. During the execution of Intra, changes to the route of one or more ships might occur. This consequently changes the input solution for the subsequent Inter operator. The change in the solution results in a change in the neighborhood of the Inter operator. High quality solutions that was originally reachable can now have been removed from the neighborhood. The result is a lower quality solution. However, it is important to keep in mind that it is equally likely that high quality solutions are added to the neighborhood of Inter.

Improvement phase performance

The improvement phase of the matheuristic is essential for finding high quality solutions. It is a substantial difference on the effectiveness of the improvement phase between the different strategy settings used by the construction phase. As can be seen by the results in Table 6.17, it is the greedy strategy which is most responsive to the improvement heuristic. Overall, the improvement phase manages to improve the greedy solutions with 9,0%. The solutions from the construction phase with the greedy strategy often have typical greedy characteristics. With greedy characteristics, we for instance mean visits added to the routes to greedily resolve a violation that could have been solved through a reordering of the current visits of the route. As a result, there are multiple possibilities for improvement for our improvement operators. The improvement heuristic use the longest time on the improvement of the greedy construction phase solutions as a large number of improvements are made in total. It is important to note that the difference of average time in the improvement heuristic is not notable in the context of both the construction phase time and exact solution method time.

Table 6.17: Average improvement phase results of the three chosen strategy settings

	Greedy	Valid	Hybrid
% Improvement [%]	9,0	1,9	4,2
Improvement time [s]	180	143	169
Number of improvements	3,5	0,93	2,2

It is also evident from Table 6.17 that the improvement heuristic has a hard time improving the construction phase solution constructed with the valid approach. As the goal of the valid approach is to construct high quality solutions, the improvement heuristic results are in line with the expectation. The improvement phase performance on the hybrid approach is also as expected, the results lie in between the greedy and valid average performance.

The performance of the improvement heuristic on greedy construction phase solutions is remarkable, hence the overall performance with the greedy setting can compete with the valid and hybrid settings. We use test case S5 to illustrate this. In the construction phase, $V1_S5$ finds the optimal solution in 49,6 seconds. We see from Table 6.16, that by employing the improvement heuristic with the Combination operator, both G_S5 and H_S5 are also able to find the optimal solution in 83 seconds, corresponding to a total of 115 and 141 seconds, respectively. This confirms that the matheuristic is able to find an optimal solution, both as the first feasible solution found in the construction phase, and after searching through the neighborhood of the initial solution in the improvement phase. In this specific example, only using the construction phase with the valid approach is faster than finding the optimal solution through both the construction and the improvement phase. In the following section we look at the overall performance of the matheuristic and determine which combination of strategies and operators that is beneficial on average.

Lastly, we must evaluate the overall performance of each of the three proposed improvement operators. Table 6.18 reports the average results of each of the improvement operators. It is equivalent to the average of the data presented in Table 6.16. On average, Inter has the shortest running time of 84 seconds followed by 162 seconds on average with the Intra operator. The slowest, but still very efficient on the exact solution method time scale, is the Combination operator with 245 seconds as average run time. All improvement operators have the ability to improve the construction phase solutions. From Table 6.18, it is apparent that the Combination operator on average facilitates for the greatest percentage improvement of the construction phase solutions.

Table 6.18: Average improvement phase results of the different improvement operators

	Inter	Intra	Combination
% Improvement [%]	5,2	3,5	6,5
Improvement time [s]	162	84	245

The purpose of adding an improvement phase to the matheuristic is to find high quality solutions. If that was the only consideration, the Combination operator would be the obvious choice as the most effective operator. However, there is a second component to the purpose of the matheuristic, namely time efficiency. Compared to the other operators, the Combination operator is not equally time efficient. As discussed, the difference in running time isolated is quite large, but not in relation to the time needed by the construction phase. Hence, the combination operator is in our judgement the best suiting operator and it is the operator used in all further testing.

6.2.7 Concluding the matheuristic test results

The proposed matheuristic is a sequential combination of a construction phase and an improvement phase. First the construction phase constructs a feasible solution in an iterative scheme. The construction phase is followed by the improvement phase where the feasible solution found by the construction phase is further improved. Until now, the construction phase and improvement phase performance have been presented and discussed independently. The purpose of this section is to summarize these results and evaluate the overall performance of our matheuristic.

Table 6.19 presents the average performance results of the overall matheuristic. The average measure reported for each size is the average of the two test cases used for the testing of the improvement phase. The first results column is the results of the exact solution method. These results are included as a benchmark. For each test case size, the average solution achieved after executing both the construction phase and the improvement phase is reported. Following is the gap. As before, the gap is the average gap between the solution found by the improvement phase and the best bound discovered by the exact solution

method. Total time of both the construction phase and the improvement phase is also reported. A final note is that all performance results of the matheuristic reported in Table 6.19 has been achieved with the use of the Combination operator in the improvement phase.

Table 6.19: Average matheuristic test results

		Exact	Greedy	Valid	Hybrid
Small	Solution	29 852	29 893	29 938	29 938
	Gap [%]	0,00	0,16	0,32	0,32
	Time [s]	1 298	123	250	393
Medium	Solution	31 749	30 664	32 322	32 112
	Gap [%]	23,4	20,4	24,6	24,2
	Time [s]	5 000	393	1 935	1 427
Large	Solution	50 783	39 975	39 927	39 975
	Gap [%]	33,4	19,8	19,7	19,8
	Time [s]	5 000	796	2 259	1 723
Total	Solution	37 461	33 511	34 062	34 012
	Gap [%]	18,9	13,2	14,7	14,5
	Time [s]	3 766	437	1 481	1 181

The results presented in Table 6.19 clearly speak in favor of the matheuristic as opposed to the exact solution method. Independent of strategy setting, the performance of the matheuristic is on average superior to the performance of the exact solution method. The matheuristic manages to return a notably smaller gap in a significantly shorter time. It is evident that the matheuristic has the greatest impact on the larger test cases. For the small-sized test cases, the exact solution method manages to find the optimal solution. Consequently, it is hard for the matheuristic to compete with the exact solution method on solution quality with the small test cases. However, the feasible solutions given by the matheuristic is of high quality and the running time is significantly shorter than the exact solution method. For the larger test cases, the exact solution method struggles to reduce the gap. The matheuristic is not affected by the increase in complexity to the same degree as the exact solution method. Hence, the matheuristic is able to close the gap of the large-sized test cases with 40% while using significantly shorter time than the exact solution method.

It is apparent in Table 6.19 that there is a difference in performance between the different strategy settings. When concluding the construction phase of the matheuristic, the greedy strategy was marked as the efficient strategy where the quality of the solution was sacrificed. The valid strategy was the strategy serving the highest quality solution, while

the hybrid strategy was the middle ground between the greedy and valid strategies. When including the improvement phase, Table 6.19 suggests a different conclusion. When also taking the improvement phase of the matheuristic into consideration, it turns out that the greedy strategy is the strategy with both the highest quality solutions and the greatest efficiency. The improvement phase did not have the same impact on the high quality input solutions from the valid strategy and the strategy is no longer superior on quality. The hybrid strategy is still on average good on both efficiency and quality of solution. Due to the greedy component of the solutions, the hybrid strategy solutions are likely more responsive to the improvement phase than the valid strategy solutions. As a result, also the hybrid solution surpass the valid strategy solution on quality.

In conclusion, the overall matheuristic is performing at its best with the greedy strategy setting and Combination as improvement operator. When only looking at the construction phase, the valid strategy is superior on quality and the greedy strategy is superior on efficiency. The inclusion of the improvement phase makes the greedy strategy the most beneficial strategy setting. For the valid strategy, the additional running time required to return high quality solutions from the construction phase does not increase the solution quality enough to outperform the greedy strategy setting.

6.3 Practical implications

In this section, practical implications of both the exact solution method and the matheuristic solution method is presented. Section 6.3.1 is included to give a deeper understanding of the functionality of the MIRP-UC. Further, we investigate the stability of the solution. Following, Section 6.3.2 also use a specific solution as test case to illustrate the functionality of the matheuristic scheme developed in this thesis.

6.3.1 Exact solution method

Figure 6.3 presents the optimal solution of the small-sized test case, *S1*. It illustrates the routing of the two ships as well as the inventory of each product in each port with a planning horizon of 14 days.

Ship 1 travels the following route: [*Sea* – 3 – 4 – 1 – 2], while *Ship 2* travels: [1 – 3 – 4 – 2]. This implies that all ports are visited twice during the planning horizon. The timing of the visits can be seen both by the times given in the figure, as well as in the development of the inventory in the ports. When the ships visit a port, the inventories spike either up or down, depending on the nature of the port. In *Port 1*, for example, it is evident that *Ship 2* handle both *Product 2* and *Product 3* in the beginning of the planning horizon. Later in the planning horizon, *Ship 1* also visits *Port 1*. However, from the inventory development, it is evident that it is only consumption *Product 2* which is unloaded from *Ship 1* in *Port 1*.

When analyzing the solution depicted in Figure 6.3 in detail, it is evident that the neither

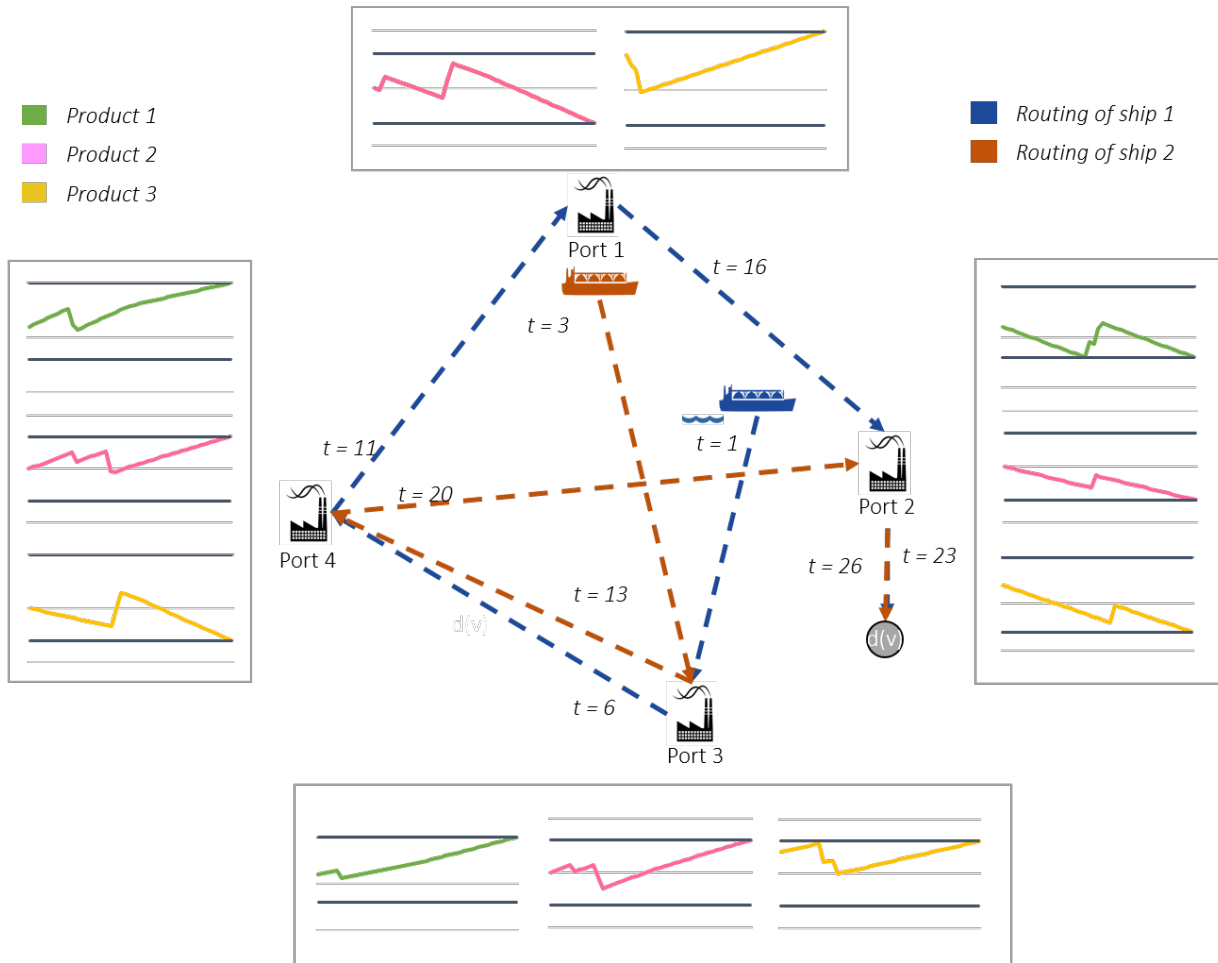


Figure 6.3: Illustration of small test case solution

of the ships utilize the option to switch product loaded in either compartments. However, this functionality is illustrated in Figure 6.4, where the route of *Ship 4* from the best found solution of the large-sized test case, *L1*, is used as example. The figure present both the route, as well as the load development of each compartment. The load visualized in each port is the load of the compartment at the end of the port visit. As the initial allocation of products to compartments is not optimal, the ship chose to empty *Compartment 3* by unloading *Product 2*. The compartment is now free to be used by any product. During the visit to *Port 3*, the empty compartment is utilized when the ship loads *Product 1* as *Product 1* is needed by *Port 8*. Consequently, *Product 1* is partially unloaded during the subsequent visit to *Port 8*. The example illustrates the functionality of undedicated compartments and how this opens up for the possibility of improving the initial allocation of products to compartments. If the compartments had not been undedicated, the ship could not have transported *Product 1* and another ship would have had to serve *Port 8* with *Product 1*.

When further evaluating the solution presented in Figure 6.3, it is notable how all inventories are at their extreme upper or lower bound in the end of the planning horizon. As MIRPs are usually considered continuous problems without a predefined end of the planning horizon, it is relevant to consider the concept of re-planning. In the case of re-

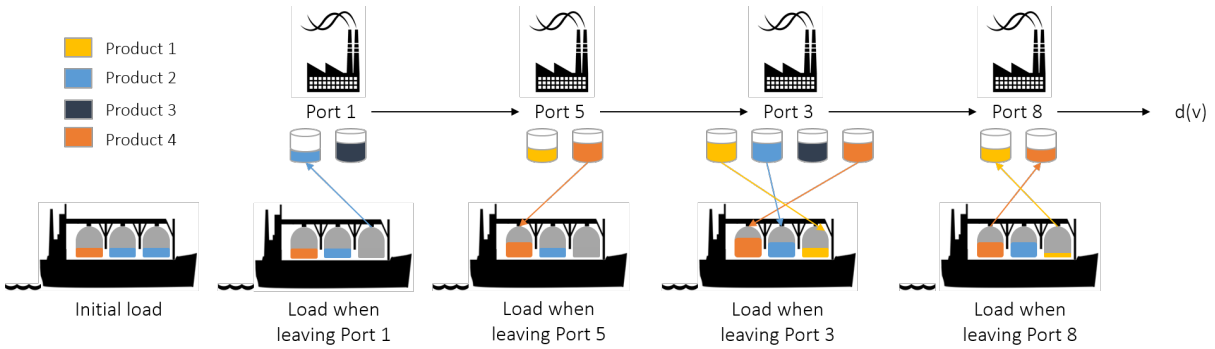


Figure 6.4: Illustration of routing solution for *Ship 4* in the large test case

optimizing this model after the last time period, the ship would not manage to reach all the ports before the inventory limits are violated. Having all inventories at their extreme values at the end of the planning horizon is equivalent to having the extreme values as the initial conditions in the model that is re-optimized. Consequently, the re-optimized model starts with infeasible conditions. One way of preventing this from happening is by introducing end conditions to secure that not all ports have inventory levels at their extreme values at the end of the planning horizon. However, continuous problems like MIRPs are usually handled using a rolling horizon scheme where end conditions are not necessarily needed. As a rolling horizon scheme is likely to only use a first fraction of the solution, the ending state of the model is not equally important. In order to be able to use a rolling horizon scheme, the stability and robustness of the solution over different lengths of planning horizon must be investigated.

Stability of solution with respect to time horizon

In this subsection, we want to examine how the solution changes when the length of the planning horizon changes.

The inventory routing problem is considered to be an ongoing planning problem, i.e. there is usually no predetermined ending of the planning horizon. A continuing planning problem is hard to solve, both due to its size and complexity. Further, for long planning horizons, it is unlikely that all long term information is known at the point of planning. For a short term planning problem, it might be fair to argue that demand, price and cost levels can be assumed deterministic. This is however not a reasonable assumption if the planning horizon is a year or so. Kimms (1997), among others, presents a rolling horizon scheme to solve this problem. You start with a plan for the $1, \dots, T$ periods while only the plan for the first fraction of the planning horizon $\Delta T \geq 1$ is implemented. Subsequently, a new plan is generated for the remaining planning periods, $T - \Delta T$, and a rescheduling occurs (Kimms, 1997). The basic principle of such a rolling horizon scheme is to repeatedly solve the MIRP for shorter sub-horizons of the planning period (Rakke et al., 2011). For problems that do not have any predetermined ending of the planning horizon, it is normal to utilize a rolling horizon scheme.

One important factor for the use of such a scheme is the re-planning point, i.e. at which point in the planning horizon the plan is re-optimized. One property that is desired is thus stability of the solution with respect to the time periods prior to the re-planning point. For example, given a problem with a two week planning horizon and a wanted re-planning point at one week, the solution of the first week should remain unchanged if we extend the overall planning horizon with one week. If a solution exhibits this stability property, we know that the solution (prior to the wanted re-planning point) is not directly dependent on the length of the planning horizon.

Given a wanted re-planning point we want to know how long the planning horizon must be to be able to guarantee a stable solution prior to the re-planning point. Since the size of the problem increases with the length of the planning horizon, it is desirable to not use a longer planning horizon than necessary. As the problem will be re-optimized multiple times with the full length planning horizon, efficiency is essential. For the remaining parts of this section, we use a re-planning point of one week as a starting point to evaluate the stability. By testing the model with planning horizons of different lengths, we can investigate if the model gives a stable solution during the first week of the planning horizon. We do not go into further details on rolling horizon and its implementation, the only focus is the stability of the solution.

To evaluate the stability, we have tested the model with three different lengths of the planning horizon, namely 14 days, 18 days and 21 days, corresponding to 42, 54 and 63 time periods. The results from using a planning horizon of 14 days corresponds to the original solution as presented in Figure 6.3. The resulting schedules of the three instances are presented in Figure 6.5. The schedules of the ships with a planning horizon of 18 and 21 days are only presented up until the time period where the original schedule ends (when both ships have sailed to the destination node). Since the purpose of this discussion is to investigate the stability of the model, we only need the part of the schedules corresponding to the time span of the original solution.

From the figure, we can clearly see that the solution of *Ship 2* is quite stable with respect to the length of the planning period. Out of the 42 time periods of the original solution, the routing of *Ship 2*, can be judged stable the first half of the time horizon. When looking at the schedules of *Ship 1*, we however see a greater variance in the schedules. None of the three schedules follow the same route. Hence, the solution corresponding to the schedule of *Ship 1* can hardly be judged stable.

It is clear that a planning horizon of 21 days is not long enough to conclude that the first week of the solution is stable. Therefore, we cannot guarantee a stable solution for the wanted re-planning point of one week. Preferably, we would like to test the model with even longer planning horizons to investigate further, but the model could not find any feasible solutions for problems of that size, at least not in a reasonable amount of time. We can however conclude that with a wanted re-planning point of one week, a longer planning horizon than 21 days is needed to achieve a stable solution. It should be noted that it is

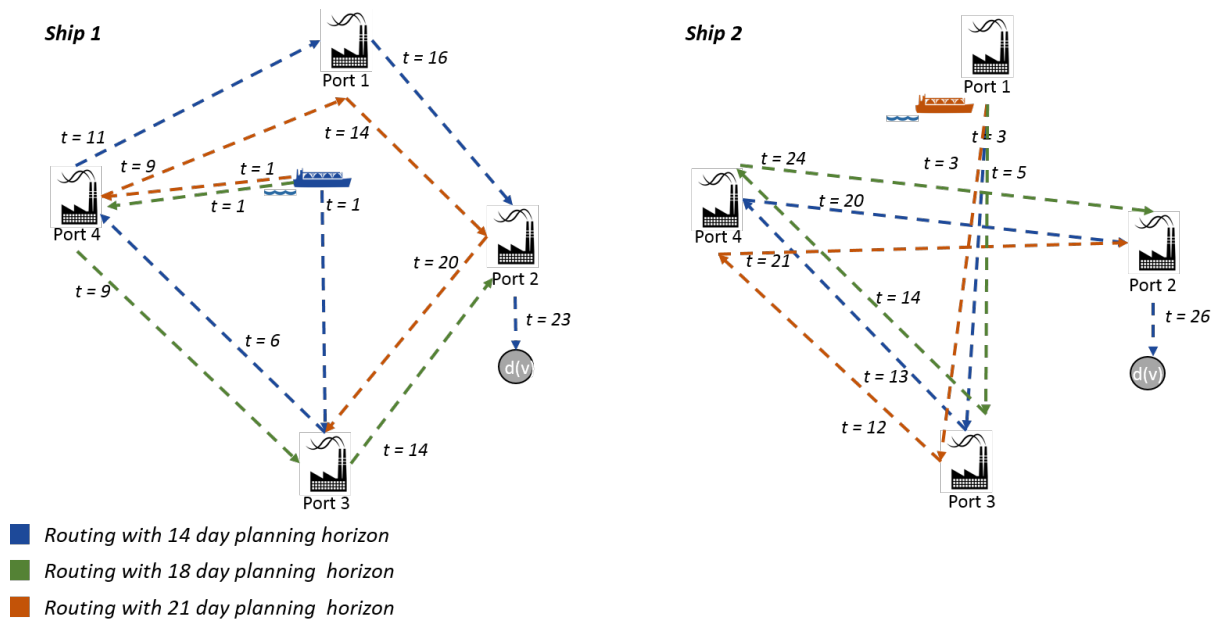


Figure 6.5: Comparison of routing solution when different lengths of planning horizons are applied

possible to shorten the planning horizon needed to achieve a stable solution in the first week by introducing some simple adjustments to the model, e.g. end conditions. We do not elaborate further on this in this thesis, but it is an interesting topic for further research.

6.3.2 Matheuristic solution method

In earlier sections of this chapter, the technical performance of the matheuristic has been thoroughly presented, discussed and analyzed. In order to get a deeper understanding of the procedure undertaken to produce a feasible solution, we take a look at a specific example.

The first component of the matheuristic is the construction phase. The construction phase is an iterative method converging towards a feasible solution. How the search is guided in each iteration depends on the strategy setting used. We present both an example of the greedy strategy as well as the valid strategy. The goal is to both understand the overall procedure of the construction phase and to understand the difference between the two strategy settings. Test case *S3* is chosen as the illustrative example.

The greedy strategy

Figure 6.6 is an illustration of the iterative search for a feasible solution using the greedy strategy to handle violations. The solution presented in each iteration is the inventory problem solution with the corresponding violations. The figure illustrates when each port is visited, and consequently when the violations to each port occur. The length of the orange and blue taps, the port visits from *Ship 1* and *Ship 2* respectively, corresponds to the number of time periods the ships operate in that port.

In *S3*, the greedy strategy uses 4 iteration to discover a feasible solution. The initialization of the routing problem results in two violations in the first iteration of the inventory problem, one large violation in *Port 1* of *Product 1* and one smaller in *Port 4*, also of *Product 1*. In *Iteration 2*, the violation in *Port 1* is adjusted for by routing *Ship 1* directly from production *Port 4* to the consumption violation *Port 1*. Both violations from the first iterations are solved, however at the expense of the feasibility in *Port 2* and *Port 3*. The greedy strategy handles the violations in *Port 3* by routing *Ship 1* directly from production *Port 1* of *Product 1* to consumption violation *Port 3*. In addition, the visit to *Port 2* is extended. As a result, only one small violation is left to be handled in the third iteration. The violation of *Product 2* in *Port 2* at the end of the planning horizon is resolved by adding an additional short visit by *Ship 1* later in the planning horizon. Since *Port 4* is a production port of *Product 2*, a visit to *Port 4* is also greedily forced into the route as it was needed prior to the second visit to *Port 2* by *Ship 1*.

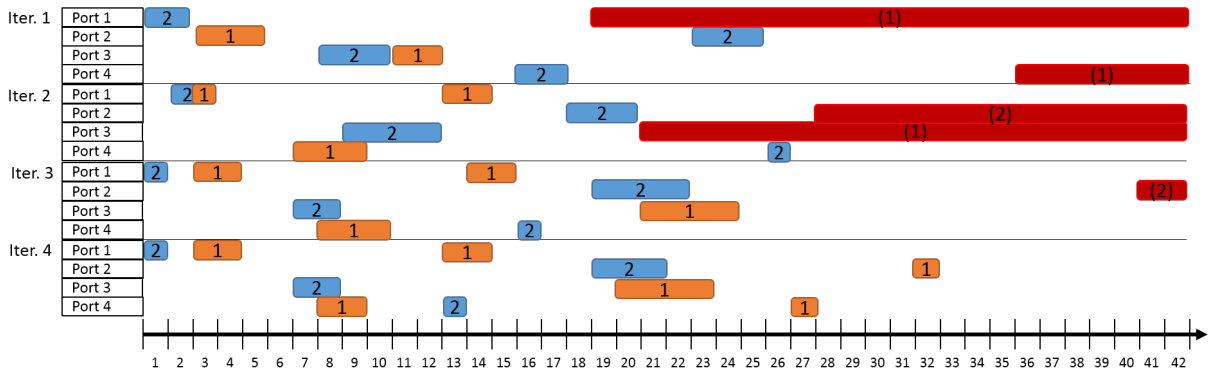


Figure 6.6: Illustration of construction phase solution with the greedy strategy. *Ship 1* marked in orange and *Ship 2* marked in blue. The red tab illustrates a violation with the violation product given inside the parenthesis

The valid strategy

The convergence of the greedy strategy is fast. This can be explained by the fact that greedy choices that actually resolve each violation are made in each iteration. As a result, the greedy strategy does not have to struggle with the same set of violations in more than one iteration. The constraints added by the valid strategy is not as restrictive as the constraints of the greedy strategy. Consequently, the valid strategy on average need more iterations to find a feasible solution than the greedy strategy. The search for a feasible solution performed with the valid strategy on *S3* is illustrated in Figure 6.7.

There is a clear difference between the development of the valid strategy search and the greedy strategy search. In the two first iterations, the two violations incurred are the same as for the greedy strategy. What is evident from Figure 6.7 is that the valid strategy does not have the same restrictive impact as the greedy strategy. In *Iteration 5*, the same violation as was present in *Iteration 4* reoccurs. Even though valid adjustments are made to the route solution of *Iteration 4*, the violation still reappear in *Iteration 5*. Due to the

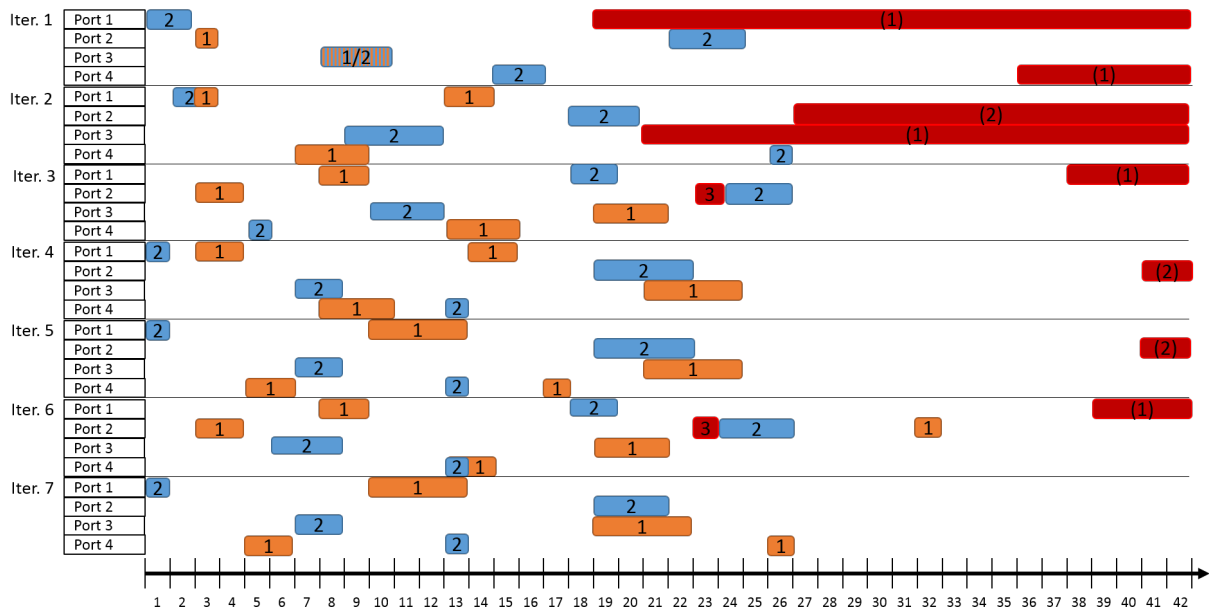


Figure 6.7: Illustration of construction phase solution with the valid strategy. *Ship 1* marked in orange and *Ship 2* marked in blue. The red tab illustrates a violation with the violation product given inside the parenthesis

multiple strategies of the valid strategy, it is harder to pinpoint which of the restrictions imposed are binding. However, the resulting route is better than the resulting route of the greedy solution. In the valid strategy solution, *Ship 1* only have four port visits as opposed to five in the greedy strategy solution. An interesting note is that in the case of the hybrid strategy, the violation in *Iteration 5* would be considered a returning violation and a greedy restriction would have been imposed to solve it.

The improvement phase

The differentiating factor between the quality of the greedy strategy solution and the valid strategy solution is evident in the routes illustrated in Figure 6.6 and Figure 6.7. It is now interesting to further illustrate the functionality of the improvement phase. We continue to use *S3* as the illustrative example. Since we concluded that the greedy strategy was most responsive to the improvement operators, we only use the solution from the construction phase using the greedy strategy to illustrate the improvement phase. Further, we also only use the Combination operator as it was concluded as the most promising operator. Since the Combination operator is the sequential execution of the Intra and Inter operator, both Intra and Inter operators are illustrated. First, we study how the routes from the construction phase have improved after the Intra operator has been fully executed. This is illustrated in Figure 6.8. The figure shows the original route of *Ship 1* and *Ship 2*, marked in blue, and the resulting routes after the Intra improvement, marked in orange.

As noted earlier, the second visit made by *Ship 1* to *Port 4* was added in the construction phase solely to resolve the last violation. Due to the greedy nature of the creation of the solution, an additional visit is added to the route instead of reordering the port visits in the route to make the route feasible. In the first iteration of the improvement phase, *Ship 1*

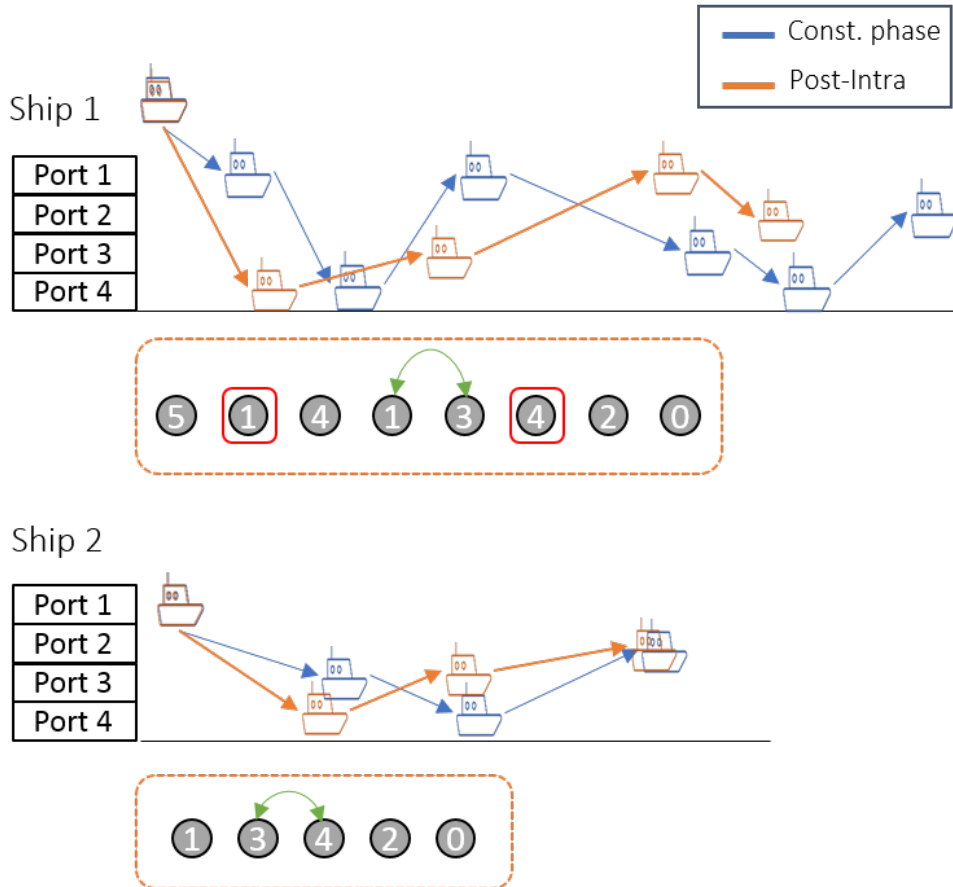


Figure 6.8: Comparison of construction phase input route and the resulting route from executing the Intra operator

is subject for improvement. The ship is fixed to the set of port visits already in the route, however not to the sequence or the frequency of visits. As a result, the Intra operator is able to reorder the port visits by switching the second visit to *Port 1* with the visit to *Port 3*. Consequently, the the second visit to *Port 1* is moved earlier in the planning horizon making the first visit to *Port 1* redundant. Further, with the new sequence of port visits, the additional visit to *Port 4* added by the greedy strategy is also identified as redundant. The two redundant port visits are removed from the sequence by the Intra operator. This illustrates how the Intra improvement operator can be used to undo some of the greedy choices made in the construction phase. Only one improvement is made for *Ship 2* with the Intra operator, switching the port visit to *Port 3* with the visit to *Port 4*.

After two iterations, where the route of *Ship 1* and *Ship 2* is improved respectfully, the Combination operator continues with the execution of the Inter operator. For the example test case *S3*, the sixth time interval stretching from time period 16 to time period 25 is the only time interval in which changes to the routes occurred. Hence, the resulting routes after improvement in that time interval are illustrated in Figure 6.9. As in the previous improvement illustration, the input route from the Intra operator is marked in blue. The improved route from employing the Inter operator is marked in orange.

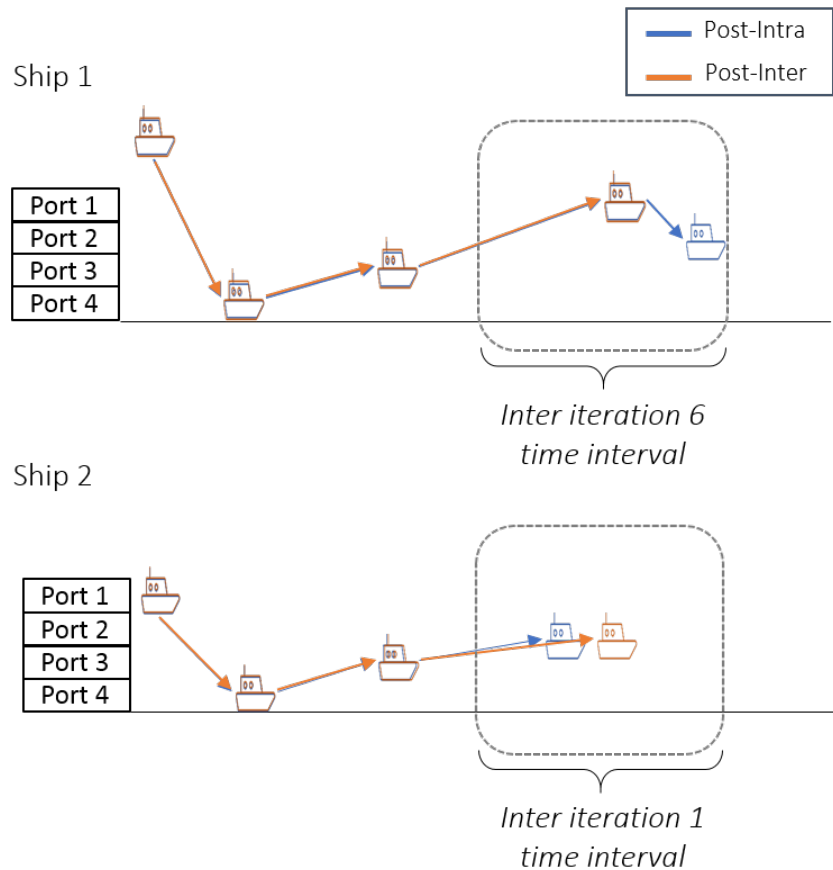


Figure 6.9: Comparison of Intra operator input route and the resulting route from executing the Inter operator

In line with the definition of the Inter operator, changes across the ship routes of *Ship 1* and *Ship 2* can only be made for the port visits occurring within the current time interval. The port visits subject to improvement are embedded in the dotted rectangle in Figure 6.9. The only changes feasible outside the current time interval are scheduling changes, i.e. adjustments in the timing of each port visit. The adjustment cannot exceed the number of time periods specified by the time slack parameter. One exception is the adjustment of the departure to $d(v)$. This allows a ship to change the last port visit of the route. It is evident from the figure that the Inter operator improves the route of *Ship 1* by removing the visit to *Port 2* at the end of the route. The action of removing the visit to *Port 2* was not possible when employing the Intra operator. The removal of the visit to *Port 2* was made possible because it is adjusted for in the schedule of *Ship 2*. In the post-Inter solution, *Ship 2* operates longer in *Port 2* than in the schedules where *Ship 1* also visit *Port 2*. Clearly, this improvement to the solution is only possible due to the fact that the Inter operator takes all the ships into account when improving the solution.

Chapter 7

Concluding Remarks and Further Research

The attention on MIRPs in OR has increased the last decade as an effect of the increase in inter supply chain competition. Coordinating efforts between the inventory management and the transportation planning can potentially provide benefits for the supply chain manager, namely flexibility in services, economical benefits and improved benefits among others. The mainstay of MIRPs are liquid bulk products. Even though most of the liquid bulk products handled by operators are unmixable, most of the currently existing research disregard this underlying characteristic. The majority does not take the allocation of products to compartments on board the ships into consideration, due to the noteworthy increase in complexity this assumption imposes.

The problem studied in this thesis is a case independent multi-product MIRP which considers the dynamic allocation of products to undedicated compartments. To our knowledge, this is the first contribution on MIRPs using undedicated compartments without the assumption that all compartments must be empty prior to the return to a production port. The objective of the presented model is to minimize costs while routing the ships to manage the multi-product inventories in the ports.

Not surprisingly, the exact solution method was only able to solve small test instances to optimality. When the size of the instances grew, the effectiveness of the exact solution method decreased. MIRPs are considered to be highly complex problems, and with the addition of multiple products and undedicated compartments, the complexity increases even further. Consequently, it was necessary to develop an alternative solution method designed specialized for our problem. We proposed a new optimization based algorithm to solve the multi-product MIRP. The algorithm is considered to be a matheuristic and utilizes an interaction between the exact solution methods and heuristic approaches.

The matheuristic consists of two phases, a construction phase and an improvement phase. The construction phase utilizes the structure of the problem and extract the solving of the routing component of the problem from the MIRP. The routing problem and the remaining part of the problem, denoted the inventory problem, is solved iteratively. First, the routing

problem is solved with a series of valid inequalities to find the sequence of port visits for each ship. This sequence is fixed in the subsequent solving of the inventory problem. If the routes are feasible, the construction phase is terminated. If the routes are infeasible, information about the infeasibilities are sent to the next iteration routing problem where it is handled using the different proposed strategies. When the construction phase has converged to a feasible solution, an improvement phase is executed with one of the proposed improvement operators to further improve the feasible solution found.

7.1 Conclusion

The model formulation was tested on three different sized test cases with the exact solution method. The model was only able to find the optimal solution on the smallest of the three test cases, even though none of the test cases can be considered real-sized. The model was tested with a number of valid inequalities. All but one of the proposed valid inequalities removed the original LP solution in at least one test case and were consequently defined as cuts. MCP gave consistently better results than the other valid inequalities and is, on average, the best performing valid inequality. The gap between the LP bound and optimal solution was reduced with 66% for the smallest test case. As a result, the time to optimal solution was reduced with 36%. However, for the smallest test case, valid inequality combination A_MCP was surprisingly the far the most efficient and reduced the time to optimality by 87%. The complexity of the problem increases with the size of the test case, and the largest test case was only able to find a solution within 43,5% of the optimal solution within 5000 seconds. Evidently, the exact solution method is not well-suited for this problem.

Overall, the construction phase of the matheuristic managed to outperform the exact solution method on both quality of solution and time efficiency. The construction phase struggles to compete on solution quality with the exact solution method on the small-sized case. However, as the size of the test cases increases, the quality of the construction phase solutions constantly outperform the exact solution method. The same applies for the efficiency.

The greedy (G), valid (V) and hybrid (H) strategy settings were the strategies of handling violations that proved best during the testing of the construction phase. The greedy strategy managed to, on average, reduce the running time with 93% while also reducing the gap between feasible solution and exact solution method best bound with 21%. The valid strategy is superior on quality of the solutions and reduces the gap with 34%. This is however at the cost of efficiency, and time used by the valid strategy is only reduced with 56,5%.

When including the improvement phase, the solutions improve even more compared to the exact solution method. Three operators for improving the multi-product MIRP solution was proposed, Inter ship, Intra ship and Combination operator. The Combination operator proved to have the overall best performance on solution quality. It is the construction

solution produced with the greedy strategy that is most responsive to the improvement operators. As a result, the quality of the greedy strategy solutions are improved significantly, and it is overall superior to both the valid and hybrid strategies. The matheuristic is superior to the exact solution method independent of both the chosen violation handling strategy and improvement operator. However, the best combination is the greedy strategy combined with the Combination operator. With this combination, the matheuristic managed to reduce the average gap with 30% while also reducing the average time used with 88%.

7.2 Further research

In this thesis, we assume that washing of compartments between the switch of products in a compartment can be done without large costs compared to the other cost components or that the products are similar such that extensive cleaning is unnecessary. However, including washing to the formulation would make the problem more general. By including cleaning, the complexity of the model would increase even more than the model presented in this thesis. It would be interesting to add washing to the formulation in future work on multi-product MIRPs-UC as it would be a better representation of the reality. It would also be interesting to further explore how the proposed matheuristic responds to this increase in complexity.

As for all deterministic models, a suggestion for future research on this multi-product MIRP-UC is to take uncertainty into account. In the maritime transportation industry, there is no doubt that the ship route and scheduling planners are faced with a great deal of uncertainty. The weather, fuel prices and demand are just some of the uncertain factors relevant for this industry. In order to make this model suitable for practical use, uncertainty must likely be introduced to the model. The uncertain components of this problem have not been the focus of this thesis and we have assumed the problem to be deterministic. However, the inclusion of uncertainty to give a better representation of reality is highly interesting for future research on the multi-product MIRP-UC.

In Section 6.3, we discussed rolling horizon schemes in context of the exact solution method. Since MIRPs are ongoing planning problems without any predefined end to the planning horizon, it is important to consider what happens after the chosen length of planning horizon in the test cases used. An interesting direction for future work is to incorporate the proposed matheuristic into a rolling horizon scheme or other scheme for handling ongoing planning problems. However, we consider the concept of using a rolling horizon scheme as especially interesting.

Bibliography

- Aghezzaf, E.-H., Raa, B., & Van Landeghem, H. (2006). Modeling inventory routing problems in supply chains of high consumption products. *European Journal of Operational Research*, *169*(3), 1048–1063.
- Agra, A., Andersson, H., Christiansen, M., & 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., & 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., Ivarsøy, K. S., Solhaug, I. E., & Tomasgard, A. (2016). Combined ship routing and inventory management in the salmon farming industry. *Annals of Operations Research*, 1–25.
- Al-Khayyal, F., & Hwang, S. J. (2007). Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, Part I: Applications and model. *European Journal of Operational Research*, *176*(1), 106–130.
- Andersson, H. (2011). A maritime pulp distribution problem. *INFOR*, *49*(2), 125-138.
- Andersson, H., Christiansen, M., & Desaulniers, G. (2015). A new decomposition algorithm for a liquefied natural gas inventory routing problem. *International Journal of Production Research*, *54*(2), 564–578.
- Andersson, H., Hoff, A., Christiansen, M., Hasle, G., & Lokketangen, A. (2010). Industrial aspects and literature survey: Combined inventory management and routing. *Computers & Operations Research*, *37*(9), 1515–1536.
- AON. (2012). *Industry analysis: Maritime transportation* (Industry report). Advisen ltd. Retrieved from <http://www.advisenltd.com/wp-content/uploads/marine-transportation-industry-analysis-aon-2012-12.pdf>
- Archetti, C. (2014). *Matheuristics for routing problems*. Retrieved from http://www.sintef.no/projectweb/verolog2014/plenaries/matheuristics_routing_verolog2014_new.pdf (Conference presentation, VeRoLog 2014)
- Archetti, C., Bertazzi, L., Hertz, A., & Speranza, M. G. (2012). A hybrid heuristic for an inventory routing problem. *INFORMS Journal on Computing*, *24*(1), 101-116.
- Archetti, C., Doerner, K. F., & Tricoire, F. (2013). A heuristic algorithm for the free newspaper delivery problem. *European Journal of Operational Research*, *230*(2), 245-257.
- Archetti, C., & Speranza, M. (2014). A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, *245*(2), 223-246.

- Bertazzi, L., & Speranza, M. (2012). Matheuristics for inventory routing problems. In J. Montoya-Torres, A. Juan, L. H. Huatucó, J. Faulin, & G. Rodríguez-Verjan (Eds.), *Hybrid algorithms for service, computing and manufacturing systems: Routing and scheduling solutions* (pp. 1–14). IGI Global.
- Bilgen, B., & Ozkarahan, I. (2007). A mixed-integer linear programming model for bulk grain blending and shipping. *International Journal of Production Economics*, *107*(2), 555 - 571.
- Boschetti, M. A., Maniezzo, V., Roffilli, M., & Bolufé Röhler, A. (2009). Matheuristics: Optimization, simulation and control. In M. J. Blesa, C. Blum, L. Di Gaspero, A. Roli, M. Sampels, & A. Schaerf (Eds.), *Hybrid metaheuristics* (pp. 171–177). Springer Berlin Heidelberg.
- Bredstrom, D., Carlsson, D., & Ronnqvist, M. (2005). A hybrid algorithm for distribution problems. *IEEE Intelligent Systems*, *20*(4), 19-25.
- Campbell, A. M., & Savelsbergh, M. W. P. (2004). A decomposition approach for the inventory-routing problem. *Transportation Science*, *38*(4), 488-502.
- Christiansen, M. (1999). Decomposition of a combined inventory and time constrained ship routing problem. *Transportation Science*, *33*(1), 3-16.
- Christiansen, M., & Fagerholt, K. (2014). Chapter 13: Ship routing and scheduling in industrial and tramp shipping. In P. Toth & D. Vigo (Eds.), *Vehicle routing* (p. 381-408). MOS-SIAM.
- Christiansen, M., Fagerholt, K., Flatberg, T., Haugen, O., Kloster, O., & 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., & Ronen, D. (2007). Chapter 4 maritime transportation. In C. Barnhart & G. Laporte (Eds.), *Transportation* (Vol. 14, p. 189 - 284). Elsevier.
- Christiansen, M., Fagerholt, K., Nygreen, B., & Ronen, D. (2013). Ship routing and scheduling in the new millennium. *European Journal of Operational Research*, *228*(3), 467 - 483.
- Christiansen, M., Fagerholt, K., & Ronen, D. (2004). Ship routing and scheduling: Status and perspectives. *Transportation Science*, *38*(1), 1-18.
- Coelho, L. C., Cordeau, J.-F., & Laporte, G. (2011). The inventory-routing problem with transshipment. *Computer & Operations Research*, *39*(11), 2537-2548.
- Coelho, L. C., Cordeau, J.-F., & Laporte, G. (2012). Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, *24*, 270 - 287.
- Cóccola, M. E., & Méndez, C. (2015). An iterative milp-based approach for the maritime logistics and transportation of multi-parcel chemical tankers. *Computers Industrial Engineering*, *89*, 88 - 107. (Maritime logistics and transportation intelligence)
- Dauzère-Pérès, S., Nordli, A., Olstad, A., Haugen, K., Koester, U., Myrstad, P. O., Teistklub, G., & Reistad, A. (2007). Omya Hustadmarmor optimizes its supply chain for delivering calcium carbonate slurry to European paper manufacturers. *Interfaces*, *37*(1), 39–51.
- Foss, E., Myklebust, T. N., Andersson, H., & Christensen, M. (2016). *A multi-product mar-*

- itime inventory routing problem with undedicated compartments*. (Submitted revised to Lecture Notes in Computer Science, Springer Verlag)
- Guerrero, W., Prodhon, C., Velasco, N., & Amaya, C. (2013). Hybrid heuristic for the inventory location-routing problem with deterministic demand. *International Journal of Production Economics*, 146(1), 359 - 370.
- Halvorsen-Weare, E. E., & Fagerholt, K. (2013). Routing and scheduling in a liquefied natural gas shipping problem with inventory and berth constraints. *Annals of Operations Research*, 203(1), 167–186.
- Hemmati, A., Hvattum, L. M., Christiansen, M., & 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.
- Kimms, A. (1997). Rolling planning horizon. In *Multi-level lot sizing and scheduling* (p. 239-245). Physica-Verlag HD.
- Li, J., Karimi, I., & Srinivasan, R. (2010). Efficient bulk maritime logistics for the supply and delivery of multiple chemicals. *Computers & Chemical Engineering*, 34(12), 2118 - 2128.
- Maraš, V., Lazić, J., Davidović, T., & Mladenović, N. (2013). Routing of barge container ships by mixed-integer programming heuristics. *Applied Soft Computing*, 13(8), 3515 - 3528.
- Michel, K., & Noble, P. (2008). Technological advances in maritime transportation. *The Bridge*, 38(2), 33-40.
- Nygreen, B., Christiansen, M., Haugen, K., Bjørkvoll, T., & Kristiansen, (1998). Modeling norwegian petroleum production and transportation. *Annals of Operations Research*, 82, 251-268.
- Persson, J. a., & Göthe-Lundgren, M. (2005). Shipment planning at oil refineries using column generation and valid inequalities. *European Journal of Operational Research*, 163(3), 631–652.
- Raa, B., & Aghezzaf, E.-H. (2008). Designing distribution patterns for long-term inventory routing with constant demand rates. *International Journal of Production Economics*, 112(1), 255-263.
- Raa, B., & Aghezzaf, E.-H. (2009). A practical solution approach for the cyclic inventory routing problem. *European Journal of Operational Research*, 192(2), 429–441.
- Rakke, J. G., Andersson, H., Christiansen, M., & Desaulniers, G. (2014). A new formulation based on customer delivery patterns for a maritime inventory routing problem. *Transportation Science*, 49(2), 384-401.
- Rakke, J. G., Stålhane, M., Moe, C. R., Christiansen, M., Andersson, H., Fagerholt, K., & Norstad, I. (2011). A rolling horizon heuristic for creating a liquefied natural gas annual delivery program. *Transportation Research Part C: Emerging Technologies*, 19(5), 896 - 911.
- Ronen, D. (2002). Marine inventory routing: shipments planning. *Journal of the Operational Research Society*, 53(1), 108–114.
- Savelsbergh, M., & Song, J.-H. (2008). An optimization algorithm for the inventory routing problem with continuous moves. *Computers Operations Research*, 35(7), 2266 - 2282.

- Siswanto, N., Essam, D., & Sarker, R. (2011). Solving the ship inventory routing and scheduling problem with undedicated compartments. *Computers and Industrial Engineering*, *61*(2), 289–299.
- Song, J.-H., & Furman, K. C. (2013). A maritime inventory routing problem: Practical approach. *Computers Operations Research*, *40*(3), 657 - 665.
- Stålhane, M., Rakke, J. G., Moe, C. R., Andersson, H., Christiansen, M., & Fagerholt, K. (2012, February). A construction and improvement heuristic for a liquefied natural gas inventory routing problem. *Computers & Industrial Engineering*, *62*(1), 245–255.
- Stålhane, M. (2015a). *Matheuristics* [Lecture Notes]. (Department of Industrial Economics and Technology Management)
- Stålhane, M. (2015b). *Valid inequalities* [Lecture Notes]. (Department of Industrial Economics and Technology Management)
- UNCTAD. (2015). *Review of maritime transport 2015* (Industry report, UNCTAD/RMT/2015). Retrieved from http://unctad.org/en/PublicationsLibrary/rmt2015_en.pdf
- Yu, Y., Chen, H., & Chu, F. (2008). A new model and hybrid approach for large scale inventory routing problems. *European Journal of Operational Research*, *189*(3), 1022 - 1040.

Appendix A

ICCL paper, 2016

Forthcoming in Lecture Notes in Computer Science series of Springer Verlag.

A Multi-product Maritime Inventory Routing Problem with Undedicated Compartments

Elise Foss, Trine N. Myklebust, Henrik Andersson, and Marielle Christiansen

Norwegian University of Science and Technology, Department of Industrial Economics and Technology Management Trondheim, Norway

Abstract. This paper considers the problem of routing bulk tankers to minimize cost while managing the inventory in ports. Multiple non-mixable products are transported and the allocation of products to undedicated compartments on board the ships is an important aspect of the problem. An arc-flow formulation of the problem is proposed together with several valid inequalities. Computational results are reported for an evaluation of the model and the valid inequalities. Results are also reported for two simplified models where either the compartments are dedicated or the products are mixable.

1 Introduction

Maritime transportation has long taken a dominant role in global trade. According to AON (2012), 90% of all goods traded across borders are moved by the maritime shipping industry. Remarkable improvements in the efficiency of maritime transportation have been seen in the last 50 years, but still significant improvements can be made by improving the routing and scheduling of ships through the use of operations research.

A maritime inventory routing problem (MIRP) is a planning problem where the problem owner has the responsibility for both the inventory management at one or both ends of the maritime transportation legs and for the ship routing and scheduling. MIRPs are considered to belong to the industrial shipping segment where the mainstay is liquid or dry bulk cargoes that is shipped in large quantities. When transporting liquid bulk, the products are stored in large compartments on board the ship. As of 2014, tankers have the second largest market share with 29% of the number of vessels in the markets (UNCTAD, 2015).

The purpose of this paper is to give further attention to MIRPs handling multiple non-mixable products with allocation to undedicated compartments. Most of the literature on MIRPs with multiple products simplifies the allocation of products by assigning them to dedicated compartments. Our goal is to develop a model for a MIRP with allocation of products to undedicated compartments and to explore the behavior of this model. The trade-off between the increased realism of the model and the increase in complexity is evaluated by considering two simplified models with dedicated compartments and mixable products.

The remainder of this paper is organized as follows. In Sect. 2 related literature on MIRPs and the handling of multiple products are reviewed. In Sect.

3, the problem is described in detail. The mathematical formulation and valid inequalities are presented in Sect. 4. The computational results for the undedicated compartment model and the simplified models are reported and discussed in Sect. 5. Finally, concluding remarks are given in Sect. 6.

2 Related Literature

Here, we review relevant literature on MIRPs addressing the transportation of multiple products and the allocation of these. Two recent surveys on the area of maritime transport optimization are Christiansen et al. (2012) and Andersson et al. (2010).

Ronen (2002) was the first to study the transportation of multiple products rather than a single product. Multiple products introduce new challenges like the handling of different products in different ports and ship/product compatibility. Similarly to the problem proposed in this paper, Hemmati et al. (2015) have chosen not to enforce any restrictions on the combinations of products and ports, i.e. each product can be consumed or produced in any number of ports. Al-Khayyal and Hwang (2007), Siswanto et al. (2011), and Agra et al. (2014) have a set of production ports and a set of consumption ports for each product which in a greater degree limits the flow of products. In both Hemmati et al. (2015) and Al-Khayyal and Hwang (2007), a ship is allowed to (un)load different products at the same time, but a port cannot handle the same product by different ships simultaneously. Agra et al. (2014) have solved this issue by restricting the ports to only have one ship operating at a time.

In the context of multi-product MIRPs, the bulk products that are considered often need to be transported in different compartments due to their non-mixable nature. Up until now, little research has addressed the issue of how these products are to be loaded on board the ship. Ronen (2002), Persson and Göthe-Lundgren (2005), Dauzère-Pérès et al. (2007), and Hemmati et al. (2015) all have models with multiple products, but disregard the allocation of products into compartments onboard the ship. Agra et al. (2014), Al-Khayyal and Hwang (2007), and Li et al. (2010) assume the products to be non-mixable and are thus forced to address the problem of allocating products to compartments. They define each compartment to be dedicated to a specific product, introducing a limitation on which products that can be carried by each compartment of a ship. The use of dedicated compartments is the most used method of solving the problem of allocating products to compartments.

Siswanto et al. (2011) introduce undedicated compartments which they define to be a compartment that can take any product, however it can only store one product at a time. In the event of an empty compartment, any product can be loaded to that compartment. However, Siswanto et al. (2011) assume that only a ship with empty compartments returns to the production ports and thus the danger of mixing products in the same compartment during the shipment is removed.

3 Problem Description

The multi-product MIRP in this paper considers the transportation of multiple non-mixable products and the allocation of the products to undedicated compartments onboard the ship. For maritime transportation this is especially relevant for the shipping of liquid bulk products. Given the nature of the products that are carried, the compartments must be washed regularly, and often before they can be loaded with a different product. This is necessary to avoid pollution of the products, e.g. to avoid a deposition of crude oil in the tankers. Since the time used washing a compartment between switching products is insignificant compared with the time used in port, it is disregarded.

We consider a short-sea transportation problem with a planning horizon that spans a few weeks. It is solved with respect to the ship routing and scheduling, inventory management in ports, and the allocation of products to compartments on each ship. The objective is to minimize the costs consisting of four components; sailing costs between ports, waiting costs outside a port, operating costs in port, and (un)loading unit costs in port. The value chain owner is responsible for both the inventory in ports and the routing and scheduling of the ships, and for that reason the inventory holding costs are ignored.

The problem deals with the transportation of multiple products in a many-to-many distribution network. Each port has a berth capacity restricting the number of ships operating in the port simultaneously.

Each ship has an initial start position either at a port or a point at sea. The sailing time between all ports is known for all ships. A ship is not allowed to visit a port without operating in that port. When a ship arrives at a port, it may wait outside the port before starting to operate. Operate is the activity of (un)loading products during a ship's port visit. Waiting outside a port may be necessary if e.g. there is no available berth at the port, or to better time the start of operation with the inventory levels in the port. However, after a ship has started to operate in a port, the ship is not allowed to wait and then continue to operate. When a ship has finished all operating activities in a port, it must immediately sail to its next destination port without waiting.

Over the course of the planning period, a port either consumes or produces a set of products. All ports have one separate storage for each of the products it handles and fixed lower and upper inventory limits are specified for each product in each port. The initial inventory for each product in all ports is known. If a port neither produces or consumes a given product during the planning horizon, it does not handle that product and it does not have a storage of that product.

Each ship can carry a selection of products, possibly all. In addition, each ship has a given number of undedicated compartments where products can be allocated. The compartments can vary in size and each have a maximum capacity. The products that are transported cannot be mixed and thus a compartment can only contain one product at a time. The capacity of a compartment in a ship is often large compared with the quantity that is (un)loaded in a given port, and hence it is natural to allow partial (un)loading. If a compartment has available capacity it can be loaded with more of the same product which it currently

contains. However, if a compartment is emptied at a port, any product can now be loaded into the compartment. Allocation of products, (un)loading in port and the possibility of partial unloading are illustrated in Fig. 1.

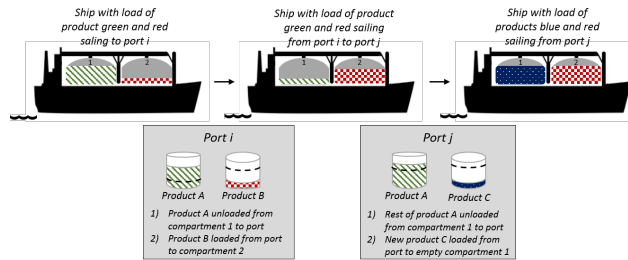


Fig. 1. An example of allocation of three products to a ship with two compartments

At the start of each schedule, the initial load of a product in every compartment in each ship is known. When a ship visits a port, the binding loading capacity under operation is the lowest of the ship's and the port's loading capacity. When a product is loaded into a compartment, it continues to stay in that compartment during sailing and waiting outside ports, until it is unloaded in a different port. Hence, no reallocation of products between compartments can take place between operating times in the ports. In a port, however, a product can be reallocated to a different compartment via the storage of that product in the port.

4 Model Description

The model is originally based on the work of Agra et al. (2013), but significant modifications have been made to account for multiple products and undedicated compartments. A time-discrete model is proposed to handle varying production and consumption rates.

4.1 Mathematical Formulation

The formulation of the problem is described in four parts: flow conservation, loading and unloading, inventory management, and objective function.

Flow Conservation Constraints. Let \mathcal{V} be the set of ships to be routed and scheduled. Each ship v has a starting position either in port or a point at sea, $o(v)$, and an artificial point of destination $d(v)$. The ships are routed to serve a set of ports \mathcal{N} and each ship will have one schedule over the planning horizon. \mathcal{T} defines the set of time periods and \bar{T} is the total number of time periods

in the planning horizon. The number of time periods needed for each ship to sail between two ports is assumed to be known, and the travel time for ship v between port i and j is defined as T_{ijv} .

B_{it} is the berth capacity in port i in time period t and limits the number of ships simultaneously operating in the port. Each ship is assumed to have three types of possible modes; sailing, waiting, and operating. To design the routing constraints with these modes, three binary variables are needed. x_{ijvt} equals 1 if ship v sails from port i directly to port j starting at the beginning of time period t , and 0 otherwise. o_{ivt} equals 1 if ship v operates in port i in time period t , and 0 otherwise. Finally, the waiting variable w_{ivt} equals 1 if ship v is waiting outside port i in time period t , and 0 otherwise. Figure (3) illustrates all the variables.

$$\sum_{j \in \mathcal{N} \cup d(v)} x_{o(v)jv1} + o_{o(v)v1} + w_{o(v)v1} = 1 \quad v \in \mathcal{V} \quad (1)$$

$$\sum_{i \in \mathcal{N} \cup o(v)} \sum_{t \in \mathcal{T}} x_{id(v)vt} = 1 \quad v \in \mathcal{V} \quad (2)$$

$$\sum_{j \in \mathcal{N} \cup o(v)} x_{jiv(t-T_{jiv})} + w_{iv(t-1)} + o_{iv(t-1)} = \sum_{j \in \mathcal{N} \cup d(v)} x_{ijvt} + w_{ivt} + o_{ivt} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (3)$$

$$o_{iv(t-1)} \leq \sum_{j \in \mathcal{N} \cup d(v)} x_{ijvt} + o_{ivt} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (4)$$

$$o_{iv(t-1)} \geq \sum_{j \in \mathcal{N} \cup d(v)} x_{ijvt} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (5)$$

$$\sum_{v \in \mathcal{V}} o_{ivt} \leq B_{it} \quad i \in \mathcal{N}, t \in \mathcal{T} \quad (6)$$

$$x_{ijvt} \in \{0, 1\} \quad i \in \mathcal{N} \cup o(v), j \in \mathcal{N} \cup d(v), v \in \mathcal{V}, t \in \mathcal{T} \quad (7)$$

$$w_{ivt}, o_{ivt} \in \{0, 1\} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (8)$$

Constraints (1) and (2) ensure that the ships' schedules have a beginning and an end. If a ship travels directly from $o(v)$ to $d(v)$, the ship is not used and is idle during the entire planning horizon. Constraints (3) are the ship flow conservation constraints. Constraints (4) restrict the ships to only be able to wait prior to operation. Constraints (5) enforce operations in a port, i.e. a ship cannot leave a port prior to operating, while constraints (6) are the berth capacity constraints. Constraints (7) and (8) are the binary restrictions. Figure 2 shows an example of a ship's route.

As can be seen, at time period one, $t = 1$, the ship sails directly from its origin node, o , and arrives in *Port 2* at $t = 5$. Then, the ship waits at $t = 6$, operates in two time periods and sails to *Port 1* at $t = 8$. The ship waits outside

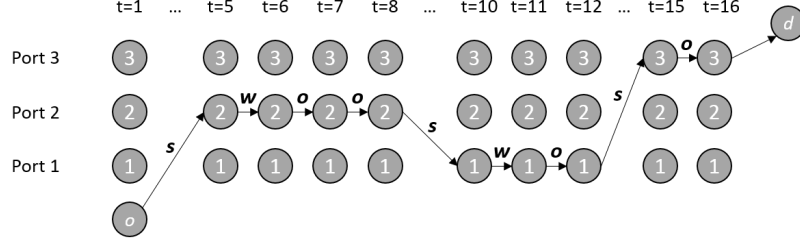


Fig. 2. Example of a ship route consisting of sailing s , waiting w , and operating o

Port 1 at $t = 11$ and operates at $t = 12$. Finally, the ship sails to *Port 3* and operates in one time period before the schedule ends at $t = 16$ when the ship sails to the destination node, d .

Loading and Unloading Constraints. Let \mathcal{K} be the set of all products, \mathcal{K}_v the set of products ship v can transport, and \mathcal{V}_k the set of ships that can transport product k . \bar{Q}_v^V and \bar{Q}_i^P define the upper (un)loading capacity of ship v and port i in each time period. Thus, each ship can (un)load as many products or as much of a product within one time period as long as it does not exceed the (un)loading capacity of the port or the ship. A ship v has a set of compartments, defined as \mathcal{C}_v , and each compartment c has a capacity \bar{K}_{vc} . Each ship v starts with an initial load in each compartment c of product k , defined as L_{vck}^0 . Variable l_{vckt} denotes the load onboard ship v of product k in compartment c at the end of time period t . Variables q_{ivckt}^L and q_{ivckt}^U represent the quantity loaded and unloaded of product k to/from compartment c by ship v from/to port i in time period t . Finally, to handle the allocation of products, variable y_{vckt} equals 1 if compartment c in ship v contains product k at the end of time period t , and 0 otherwise.

$$\sum_{k \in \mathcal{K}_v} \sum_{c \in \mathcal{C}_v} (q_{ivckt}^L + q_{ivckt}^U) \leq \min\{\bar{Q}_v^V, \bar{Q}_i^P\} o_{ivt} \quad i \in \mathcal{N}, v \in \mathcal{V}, t \in \mathcal{T} \quad (9)$$

$$l_{vck(t-1)} + \sum_{i \in \mathcal{N}} q_{ivckt}^L = \sum_{i \in \mathcal{N}} q_{ivckt}^U + l_{vckt} \quad v \in \mathcal{V}, c \in \mathcal{C}_v, k \in \mathcal{K}_v, t \in \mathcal{T} \quad (10)$$

$$l_{vck0} = L_{vck}^0 \quad v \in \mathcal{V}, c \in \mathcal{C}_v, k \in \mathcal{K}_v \quad (11)$$

$$\sum_{k \in \mathcal{K}_v} y_{vckt} \leq 1 \quad v \in \mathcal{V}, c \in \mathcal{C}_v, t \in \mathcal{T} \quad (12)$$

$$l_{vckt} \leq \bar{K}_{vc} y_{vckt} \quad v \in \mathcal{V}, c \in \mathcal{C}_v, k \in \mathcal{K}_v, t \in \mathcal{T} \quad (13)$$

$$q_{ivckt}^L, q_{ivckt}^U \geq 0 \quad i \in \mathcal{N}, v \in \mathcal{V}, c \in \mathcal{C}_v, k \in \mathcal{K}_v, t \in \mathcal{T} \quad (14)$$

$$l_{vckt} \geq 0 \quad v \in \mathcal{V}, c \in \mathcal{C}_v, k \in \mathcal{K}_v, t \in \mathcal{T} \quad (15)$$

$$y_{vckt} \in \{0, 1\} \quad v \in \mathcal{V}, c \in \mathcal{C}_v, k \in \mathcal{K}_v, t \in \mathcal{T} \quad (16)$$

Constraints (9) ensure that a ship can only (un)load when it is operating in a port and define the upper limit on the total quantity (un)loaded by a ship in a time period. Constraints (10) represent the load balance for each ship, while constraints (11) define the initial load of every product in every compartment for each ship. Constraints (12) ensure that only one product can be in each compartment at any time. The load capacity of each compartment is given in constraints (13), which also enforce the binary variable y_{vckt} to be 1 when there is a load in a compartment. Constraints (14) – (15) define the non-negativity constraints, while constraints (16) define the binary restrictions.

Inventory Management Constraints. The production/consumption quantities of a product k in port i in time period t are denoted P_{ikt} and D_{ikt} , respectively, and can vary over the planning horizon. Each port has a storage for each product it handles, and the initial inventory of product k in port i is called S_{ik}^0 . \bar{S}_{ik} and \underline{S}_{ik} define the upper and lower inventory limit in port i for product k , respectively. Variable s_{ikt} gives the inventory level in port i of product k at the end of time period t .

$$s_{ik(t-1)} + \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}_v} q_{ivckt}^U + P_{ikt} = D_{ikt} + \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}_v} q_{ivckt}^L + s_{ikt} \quad i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{T} \quad (17)$$

$$\underline{S}_{ik} \leq s_{ikt} \leq \bar{S}_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K}, t \in \mathcal{T} \quad (18)$$

$$s_{ik0} = S_{ik}^0 \quad i \in \mathcal{N}, k \in \mathcal{K} \quad (19)$$

Constraints (17) are the inventory balance for all ports and products. Constraints (18) state lower and upper inventory limits for each product in every port. Lastly, constraints (19) define the initial inventory of each product.

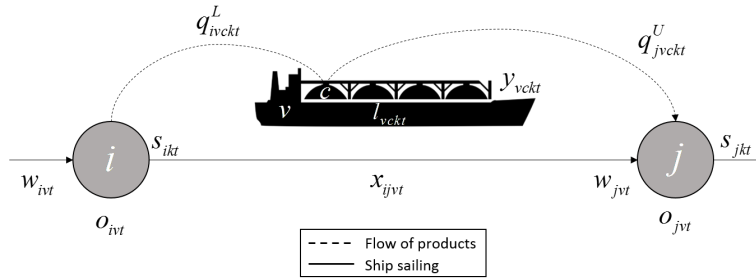


Fig. 3. Illustration of variables

Objective Function. The objective function, presented in (20), minimizes the sailing-, waiting-, and operation costs as well as the variable (un)loading costs. Sailing-, waiting- and operating costs are defined as a fixed unit cost per time period used on the activity. C_{ijv}^T is the cost of ship v sailing from port i to j . C_v^W is the cost of waiting outside a port for ship v . C_{iv}^O is the fixed cost of ship v operating in port i . There is also a variable component, C_{ivk}^Q , which is defined as the cost per unit of product k (un)loaded in port i by ship v . We assume that no costs are associated with switching between loading/unloading different products in one compartment because switching time can be considered insignificant compared with the length of a time period.

$$\begin{aligned} \min \sum_{v \in \mathcal{V}} \sum_{i \in \mathcal{N} \cup o(v)} \sum_{j \in \mathcal{N} \cup d(v)} \sum_{t \in \mathcal{T}} C_{ijv}^T x_{ijvt} + \sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} C_v^W w_{ivt} + \\ \sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} C_{iv}^O o_{ivt} + \sum_{i \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{C}_v} \sum_{k \in \mathcal{K}_v} \sum_{t \in \mathcal{T}} C_{ivk}^P (q_{ivckt}^L + q_{ivckt}^U) \end{aligned} \quad (20)$$

4.2 Valid Inequalities and Tightening Constraints

By exploiting the structure of the problem, valid inequalities have been developed to strengthen the LP-relaxation of the problem and in turn reduce the solution time. In this paper, only the most promising valid inequalities from our studies have been included.

Minimum Number of Visits with Ship Capacity Sequence (MV). MV is inspired by similar valid inequalities addressed by Andersson et al. (2015). Here, a ship capacity sequence is introduced to avoid the generalization done when the maximum ship capacity is used to calculate the lower bound for the entire planning horizon.

The ship capacity sequence is defined over a subinterval of the planning horizon and is built upon the maximum number of times each ship can visit a port. To be able to define a maximum number of visits of each ship, two assumptions are made, (1) each ship will only travel back and forth from port i to its nearest port after the initial visit to port i and (2) each ship will only operate one period in each port visit. With this, the maximum number of visits ship v can make to port i in time interval $\mathcal{T}' = \{\underline{T}', \dots, \overline{T}'\}$ is V_{iv}^{MAX} , as shown in (21).

$$V_{iv}^{MAX} = \left\lceil \frac{T' - T_{jiv}}{2 \cdot T_i^{MIN} + 2} \right\rceil \quad i \in \mathcal{N}, v \in \mathcal{V} \quad (21)$$

T' is the length of time interval \mathcal{T}' , and T_i^{MIN} is the sailing time for ship v from port i to its nearest port. j denotes the ship's position at the beginning of the time interval.

Ship Capacity Sequence. The ship capacity sequence, defined for each port i , gives the maximum amount of products that can be (un)loaded in a port in m visits during a time interval. First, the highest ship capacity is added cumulatively to the capacity sequence a number of times equal to the maximum number of visits defined in (21). The same follows for the rest of the ships, in descending order based on capacity. The length of the ship capacity sequence is equal to the total number of visits to port i in the time interval, from all ships. The ship capacity sequence of port i is denoted $\bar{\mathcal{K}}_i^V = \{\bar{K}_{i0}^V, \bar{K}_{i1}^V, \dots, \bar{K}_{im}^V\}$ for $i \in \mathcal{N}$.

Assume a fleet of two ships where the largest ship has a capacity of 200 and can visit port i at most three times in time interval \mathcal{T}' . The other ship has a capacity of 100 and can visit port i a maximum of two times. The ship capacity sequence of port i , with $\bar{K}_{i0}^V = 0$ for the case of no loading, is then equal to $\bar{\mathcal{K}}_i^V = \{0, 200, 400, 600, 700, 800\}$ for the given time interval.

Excess production and consumption. The excess production of product k in port i , $e_{ik\mathcal{T}'}^P$, and the excess consumption of product k in port i , $e_{ik\mathcal{T}'}^D$, during time interval \mathcal{T}' is defined in (22) and (23), respectively.

$$e_{ik\mathcal{T}'}^P = \sum_{t \in \mathcal{T}'} P_{ikt} + s_{ik(\underline{T}'-1)} - \bar{S}_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K} \quad (22)$$

$$e_{ik\mathcal{T}'}^D = \sum_{t \in \mathcal{T}'} D_{ikt} - s_{ik(\underline{T}'-1)} + \underline{S}_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K} \quad (23)$$

Since a ship can handle both excess consumption and production in the same visit, the lower bounds on visits for produced and consumed products cannot be added together. On this note, the maximum of $e_{ik\mathcal{T}'}^P$ and $e_{ik\mathcal{T}'}^D$ aggregated over product, $e_{i\mathcal{T}'} = \max\{\sum_{k \in \mathcal{K}} e_{ik\mathcal{T}'}^P, \sum_{k \in \mathcal{K}} e_{ik\mathcal{T}'}^D\}$, is used as the restricting quantity in the inequalities.

If time interval \mathcal{T}' starts at $t = 1$, then the incoming inventory level of product k , $s_{ik(\underline{T}'-1)}$ is equal to the initial inventory of that product, S_{ik}^0 . By this, the minimum number of visits needed to serve the excess level can be calculated a priori. Let p_i be the first position in the ship capacity sequence corresponding to a capacity high enough to cover $e_{i\mathcal{T}'}$. Hence, p_i corresponds to the minimum number of visits needed. In all other cases, the incoming inventory is a variable and this simplification is impossible. The valid inequalities for time interval \mathcal{T}' are defined by (24) or (25) depending on the starting period of the time interval.

$$\sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}'} x_{jivt} \geq p_i \quad i \in \mathcal{N} \quad (24)$$

$$\sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}'} x_{jivt} \geq \frac{e_{i\mathcal{T}'} + (m-1)\bar{K}_{im}^V - m\bar{K}_{i(m-1)}^V}{\bar{K}_{im}^V - \bar{K}_{i(m-1)}^V} \quad i \in \mathcal{N}, 1 < m < |\bar{\mathcal{K}}_i^V| \quad (25)$$

Valid inequalities (24) and (25) give a lower bound on the number of visits to port i in time interval T' .

Minimum Number of Compartments per Product with Compartment Capacity Sequence (MCP). MCP is an extension of a valid inequality presented by Andersson et al. (2015) adapted to account for both multiple products and a heterogeneous set of tanks on the ships. A compartment capacity sequence is designed equivalently to the ship capacity sequence, using compartment capacities. The sequence is created for all products k and ports i and is written as $\bar{C}_{ik}^V = \{\bar{C}_{ik0}^V, \bar{C}_{ik1}^V, \dots, \bar{C}_{ikm}^V\}$. The excess production, $e_{ik\mathcal{T}'}^P$, and consumption, $e_{ik\mathcal{T}'}^D$, are calculated in (22) and (23) respectively, and $e_{ik\mathcal{T}'}$ is the maximum of excess production and consumption. Let p_{ik} represent the first position in the ship compartment capacity sequence sufficient to cover $e_{ik\mathcal{T}'}$. p_{ik} is then the minimum number of compartments needed for each port and product combination. When N_v^C is the number of compartments in ship v , the valid inequalities MCP for time interval \mathcal{T}' are presented in (26) and (27).

$$\sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}_k} \sum_{t \in \mathcal{T}'} N_v^C x_{jivt} \geq p_{ik} \quad i \in \mathcal{N}, k \in \mathcal{K} \quad (26)$$

$$\sum_{j \in \mathcal{N}} \sum_{v \in \mathcal{V}_k} \sum_{t \in \mathcal{T}'} N_v^C x_{jivt} \geq \frac{e_{ik\mathcal{T}'} + (m-1)\bar{C}_{ikm}^V - m\bar{C}_{i(m-1)}^V}{\bar{C}_{ikm}^V - \bar{C}_{i(m-1)}^V} \quad i \in \mathcal{N}, k \in \mathcal{K}, 1 < m < |\bar{C}_i^V| \quad (27)$$

Minimum Number of Operation Periods (MO). The idea of imposing a lower bound on the minimum number of operation periods has been introduced by e.g. Agra et al. (2013) for a single-product inventory routing problem. Here, it is extended to account for multiple products. Excess production, e_{ik}^P , and consumption, e_{ik}^D are calculated by (22) and (23) respectively, but for the entire planning horizon and thus the initial stock level is S_{ik}^0 . Under the assumption that each product is either produced or consumed, only e_{ik}^P or e_{ik}^D is positive. The minimum number of operation periods required by each port is equivalent to the sum of operation periods required by each product. The valid inequalities (28) enforce a lower bound on the number of operation periods needed in each port.

$$\sum_{v \in \mathcal{V}} \sum_{t \in \mathcal{T}} o_{ivt} \geq \left\lceil \sum_{k \in \mathcal{K}} \frac{e_{ik}^P + e_{ik}^D}{\min\{\bar{Q}_i^P, \max\{\bar{Q}_v^V; v \in \mathcal{V}\}\}} \right\rceil \quad i \in \mathcal{N} \quad (28)$$

5 Computational Study

All instances of our mathematical programming models are solved using Mosel Xpress-MP. Mosel Xpress-MP is run on a Hewlett Packard 64-bit Windows 7 Enterprise PC with Intel(R) Core(TM) i7-3770 3.40 GHz processor and 16,0 GB (15.9 GB usable) RAM. Note that Xpress solves LP problems integrated with the IP solution procedure. Due to the use of Presolve in Xpress, the LP bounds that are reported in this chapter may be higher than if the LP relaxation of the IP problem was solved explicitly.

5.1 Instances and Data

The name of each instance is built up of two components; which case is used and which valid inequality that has been added. The small case consists of two ships with two compartments each, four ports, and three products and is denoted by S . The medium case have three ships, with two or three compartments each, six ports, and four products and is denoted M . Finally, the large case is equivalently denoted L and consists of four ships, with two or three compartments each, eight ports, and four products. To represent which of the three valid inequalities that has been added, the notation introduced in Sect. 4.2 is used, namely MV for valid inequalities defined in (24) and (25), MCP for valid inequalities in (26) and (27) and MO for valid inequalities in (28). UC is used to refer to the instance where no valid inequalities have been added.

MV and MCP use a time interval when deciding the binding capacity of a ship or a compartment in the ship/compartment capacity sequence. Preliminary testing showed that using the entire length of the time horizon as the length of the time interval gives the tightest formulation. Thus, all succeeding tests employ the full planning horizon as time interval. Andersson et al. (2015) present results that indicate that starting the time interval in the first time period is most beneficial. Thus, here the incoming inventory level of the time interval, $s_{ik}(\underline{T}'-1)$, is always equal to the initial inventory, S_{ik}^0 .

5.2 Exact Solution Method and Valid Inequalities

In this section, the results from the testing of the model and the valid inequalities are presented. We have tested the valid inequalities independently, as well as other interesting combinations. Table 1 shows the results from testing the model alone, and with the different valid inequalities. We use bold font to identify the best solution and best lower bounds in Table 1. Note that the LP bounds presented below corresponds to results obtained from Xpress when it solves the LP problem integrated with the IP solution procedure. This can lead to different results than if the LP relaxation was solved explicitly.

As can be seen, MV tightens the formulation and increases the LP bound, resulting in a more efficient branch-and-bound procedure. This is evident in S_{MV} as the time to optimality is improved from S_{UC} . MCP gives an even tighter formulation and higher LP bound than MV. S_{MCP} gives a significantly

Table 1. Results of small-, medium- and large-sized test cases with different valid inequalities. Running time 5000 seconds

Test case Info		UC	MV	MCP	MO	MCP_MV
S	LP bound	16 257	18 975	25 199	16 419	20 395
	Best solution	29 883	29 883	29 883	29 883	29 883
	Best bound	29 883	29 883	29 883	29 883	29 883
	Time to optimality	2 100 s	1 783 s	1 012 s	3 928s	3 463s
M	LP bound	19 016	21 589	25 198	19 547	19 779
	Best solution	35 633	42 415	36 673	35 233	35 153
	Best bound	26 065	25 742	27 582	26 183	24 518
	Gap	26.9%	39.3%	24.8%	25.7%	30.3%
L	LP bound	24 473	28 293	34 391	24 592	29 377
	Best solution	64 930	57 252	61 310	61 673	-
	Best bound	31 017	30 718	34 933	30 782	30 443
	Gap	52.2%	46.3%	43.0%	50.1%	-

better time to optimality and has the best performance over all test cases, in terms of both the highest bound and the lowest gap. MCP reduced the gap between the LP solution and the optimal integer solution from 45.6% to 15.9% in the small test case, and it is reasonable to believe that this reduction can explain the high efficiency of this inequality. MV and MCP are the two best performing valid inequalities, however, a combination of the two is not efficient. The combination of the two increases the complexity more than it manages to reduce the search space, and the time to optimality is higher than without any valid inequalities.

Even though the LP bound is improved, S_MO has the highest running time to optimality. While MO has one of the highest running times to optimality in the small test case, M_MO finds a good integer solution and thus achieves one of the best bounds. Even though MO showed a slight improvement from the small- to the medium-sized test case, it did not show any improvement in the large-sized test case.

5.3 Model Simplifications

To use undedicated compartments (UC) to model the handling of multiple non-mixable products is a highly realistic approach to real life applications. However, alternative approaches do exist, namely either employing dedicated compartments (DC), or assuming the products to be mixable and thus no separate compartments are needed (NC). Only minor changes are needed to the UC model introduced to employ either DC or NC, and explicit formulations for these models are not included. In this section, we report and compare the performance of the three models. To be able to compare the models, no valid inequalities have been added to the test instances. UC, DC, and NC are used to denote which model is tested.

The UC model has the freedom to change which products that are loaded in which compartments and can thus choose an optimal allocation of each product, while DC must always adhere the capacity constraints of each product's

dedicated compartment(s). This is a restriction of the UC model; less variables are needed and the complexity of the model is reduced. In NC models, the only capacity limit is the ship capacity, and NC is thus a relaxation of the UC model. NC is an even greater simplification than DC. Compared with NC, the number of variables and constraints in UC increases approximately 30% and 20% respectively. The results of the test instances can be found in Tab. 2.

Table 2. Model simplification results of the three test cases. Running time 5 000 seconds.

Test case	Info	NC	UC	DC
S	Best solution	29 883	29 883	30 303
	Best bound	29 883	29 883	30 303
	Time to optimality	737 s	2 100 s	1 519 s
M	Best solution	35 463	35 633	36 400
	Best bound	26 296	26 065	26 803
	Gap	25.8%	26.9%	26.4%
L	Best solution	57 713	64 930	52 500
	Best bound	32 905	31 017	32 957
	Gap	43.0%	52.2%	37.2%

S_NC is the first to prove optimality, followed by *S_DC* and last *S_UC*, as expected. The optimal objective value in *S_DC* is higher than those of *S_NC* and *S_UC*, which illustrates that NC and UC utilize a degree of freedom not applicable in DC. *S_NC* did not, however, find a better solution than the optimal solution of *S_UC*. Note that the underlying flexibility on quantity (un)loaded of all MIRP models in general, often makes it possible for a model with compartments to adapt and replicate the solution of a model without compartments. This would, however, not be possible in cases of tramp shipping where the ship only (un)load fixed sized cargoes. For the medium-sized test case, the same pattern is seen in the solutions and the size of the gaps as in the small test case. However, due to the large gaps the results in the large-sized test case deviate from the expected pattern.

5.4 Comparison of the Model and Model Simplifications

We have chosen to compare the solutions of the model and the model simplifications on the small-sized instances since the optimal solution is known for all models. The focus is on the comparison of models with UC and DC results, since the solutions of UC and NC proved to be equal for the tested case.

Figure 4 shows how the routing of the ships differ between UC and DC. The UC model manages to find a shorter feasible route compared with the DC model. Often the reason is that in UC the ships have the flexibility of loading a compartment with any product the ship can carry after it is emptied. Emptying a compartment of a product will thus free up capacity that can now be available

to all products. Another aspect of the flexibility contained in UC is the ability to reallocate products in order to obtain the optimal allocation of products to compartments. In DC, the dedication of products to compartments is not necessarily optimal, but it cannot be improved. For example, in the optimal solution of DC, *Product 3* is dedicated/fixed to the smallest compartment. In contrast, *Product 3* ends up using the largest compartment available in the fleet in UC. Thus, in DC the compartment capacity of *Product 3* is binding and the fact that *Product 3* is fixed to the smallest compartment prevents DC from finding the optimal solution found by UC. This lack of flexibility is reflected in the costs, and the possibility of saving economical values by using UC exists.

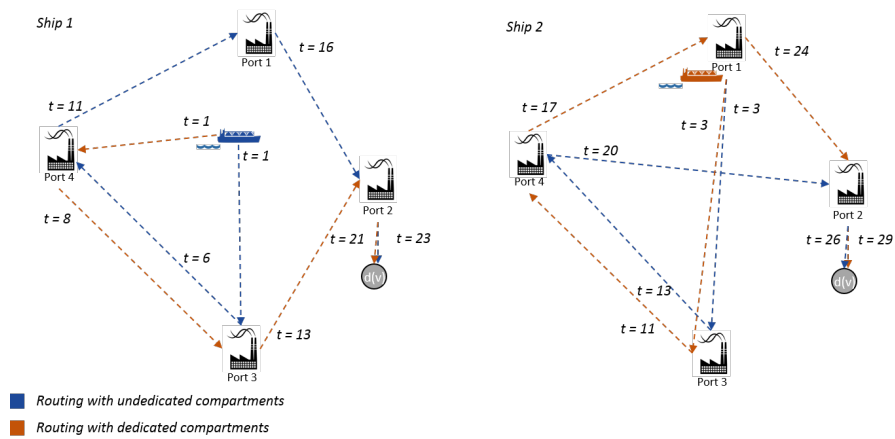


Fig. 4. Comparison of routing of ships with undedicated and dedicated compartments

When compartments are dedicated, the given capacities for each product must remain the same through the planning horizon. This implies that the DC model can be vulnerable to varying production rates while UC can more easily adapt to a high variation in supply and demand during the planning horizon by reallocating its products. UC handles the allocation of products in a more realistic way than both NC and DC, however using undedicated compartments come with the drawback of adding more complexity. The greater the number of compartments in an UC model, the less capacity is locked to a product at a time and the flexibility increases. As a result, the performance of a UC model moves toward the performance of the NC model. In a DC model, however, the performance is not dependent on the number of compartments due to the fact that it always has one fixed capacity per product.

6 Concluding Remarks

In this paper, we developed a mathematical formulation for a maritime inventory routing problem addressing the allocation of multiple products to undedicated compartments onboard the ships. Three different types of valid inequalities were developed and tested, the most promising using a capacity sequence to define the minimum number of compartment visits required in a port. Computational results were also given for two simplified models to compare different ways of handling the allocation of products. Employing undedicated compartments is the most realistic approach to real life applications but it comes with the drawback of added complexity. However, comparison with models employing dedicated compartments or mixable products indicate a potential for economical savings by using undedicated compartments.

References

- 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 Simonetti, L. (2014). Hybrid heuristics for a short sea inventory routing problem. *European Journal of Operational Research*, 236(3):924–935.
- Al-Khayyal, F. and Hwang, S. J. (2007). Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk, Part I: Applications and model. *European Journal of Operational Research*, 176(1):106–130.
- Andersson, H., Christiansen, M., and Desaulniers, G. (2015). A new decomposition algorithm for a liquefied natural gas inventory routing problem. *International Journal of Production Research*, pages 1–15.
- Andersson, H., Hoff, A., Christiansen, M., Hasle, G., and Lokketangen, A. (2010). Industrial aspects and literature survey: Combined inventory management and routing. *Computers & Operations Research*, 37(9):1515–1536.
- AON (2012). Industry analysis: Maritime transportation. White paper, Advisen ltd.
- Christiansen, M., Fagerholt, K., Nygreen, B., and Ronen, D. (2012). Ship routing and scheduling in the new millennium. *European Journal of Operational Research*, 228(3):467–483.
- Dauzère-Pérès, S., Nordli, A., Olstad, A., Haugen, K., Koester, U., Myrstad, P. O., Teistklub, G., and Reistad, A. (2007). Omya Hustadmarmor optimizes its supply chain for delivering calcium carbonate slurry to European paper manufacturers. *Interfaces*, 37(1):39–51.
- Hemmati, A., Hvattum, L. M., Christiansen, M., and Laporte, G. (2015). An iterative two-phase hybrid matheuristic for a multi-product short sea inventory-routing problem.
- Li, J., Karimi, I., and Srinivasan, R. (2010). Efficient bulk maritime logistics for the supply and delivery of multiple chemicals. *Computers & Chemical Engineering*, 34(12):2118 – 2128.
- Persson, J. a. and Göthe-Lundgren, M. (2005). Shipment planning at oil refineries using column generation and valid inequalities. *European Journal of Operational Research*, 163(3):631–652.

- Ronen, D. (2002). Marine inventory routing: shipments planning. *Journal of the Operational Research Society*, 53(1):108–114.
- Siswanto, N., Essam, D., and Sarker, R. (2011). Solving the ship inventory routing and scheduling problem with undedicated compartments. *Computers & Industrial Engineering*, 61(2):289–299.
- UNCTAD (2015). Review of maritime transport 2015. *UNCTAD/RMT/2015*.